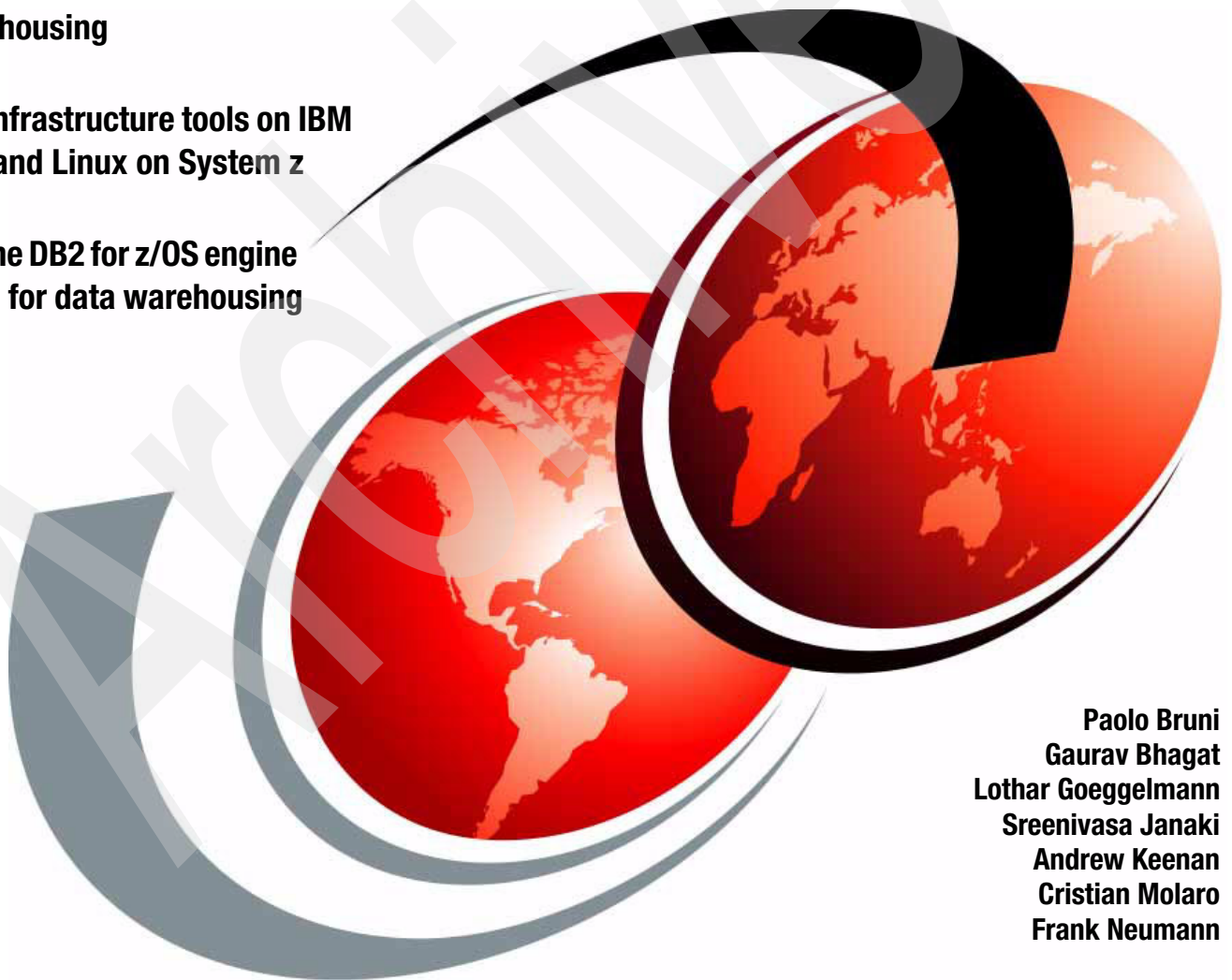


# Enterprise Data Warehousing with DB2 9 for z/OS

Understand the evolution of  
data warehousing

Evaluate infrastructure tools on IBM  
System z and Linux on System z

See how the DB2 for z/OS engine  
is suitable for data warehousing



Paolo Bruni  
Gaurav Bhagat  
Lothar Goeggelmann  
Sreenivasa Janaki  
Andrew Keenan  
Cristian Molaro  
Frank Neumann

**Redbooks**





International Technical Support Organization

**Enterprise Data Warehousing with DB2 9 for z/OS**

September 2008

Archived

**Note:** Before using this information and the product it supports, read the information in “Notices” on page xxi.

Archived

**First Edition (September 2008)**

This edition applies to IBM DB2 Version 9.1 for z/OS (program number 5635-DB2).

**© Copyright International Business Machines Corporation 2008. All rights reserved.**

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.



# Contents

<b>Figures</b> .....	ix
<b>Tables</b> .....	xv
<b>Examples</b> .....	xvii
<b>Notices</b> .....	xxi
Trademarks .....	xxii
<b>Preface</b> .....	xxiii
The team that wrote this book .....	xxiii
Become a published author .....	xxvi
Comments welcome .....	xxvi
<b>Part 1. Data warehouse today</b> .....	1
<b>Chapter 1. Definitions</b> .....	3
1.1 Introduction .....	4
1.2 The data warehouse environment .....	4
1.2.1 Data warehouse and data mart .....	6
1.3 Data warehouse data definitions .....	10
1.3.1 Data warehouse data modeling styles .....	10
1.3.2 Multidimensional data model .....	11
1.4 Data warehouse functional definitions .....	14
1.4.1 Replication techniques .....	14
1.4.2 Data transformations .....	15
1.4.3 Application techniques .....	16
<b>Chapter 2. Evolution of business intelligence</b> .....	17
2.1 Market drivers and challenges .....	18
2.2 BI technology and functionality evolution .....	19
2.2.1 Current BI trends .....	20
2.2.2 BI evolution and maturity .....	21
2.3 Types of users and their requirements .....	22
2.4 Information On Demand .....	24
2.4.1 Operational business intelligence .....	26
2.4.2 Applying Information On Demand with dynamic warehousing .....	29
<b>Chapter 3. Why implement a data warehouse on System z</b> .....	31
3.1 New challenges for data warehouse solutions .....	32
3.2 Data warehousing with System z .....	32
3.2.1 Availability and scalability .....	32
3.2.2 Workload management .....	35
3.2.3 Hardware data compression .....	36
3.2.4 Regulatory compliance .....	36
3.2.5 Disaster recovery .....	37
3.2.6 I/O connectivity .....	42
3.2.7 Parallel access volumes .....	43
3.2.8 Total cost of ownership .....	45
3.2.9 System z10 .....	46

3.2.10 Existing System z customer base . . . . .	48
3.2.11 DB2 for z/OS with additional data warehousing capabilities . . . . .	48
3.2.12 Extract, transform, and load on the same platform . . . . .	51
<b>Chapter 4. The architecture for the BI solution on System z . . . . .</b>	<b>59</b>
4.1 Business requirements for a data warehouse environment . . . . .	60
4.2 The business intelligence architecture with System z . . . . .	61
4.2.1 The components involved . . . . .	62
4.2.2 Configuration alternatives . . . . .	63
4.3 When DB2 for z/OS is a good fit . . . . .	67
4.4 Business requirements revisited . . . . .	68
<b>Part 2. Design and implementation of our warehouse scenario . . . . .</b>	<b>69</b>
<b>Chapter 5. The business scenario and data models . . . . .</b>	<b>71</b>
5.1 Background information . . . . .	72
5.2 Business requirements . . . . .	72
5.3 Solution overview . . . . .	75
5.4 The transactional data model . . . . .	78
5.4.1 The OLTP database model . . . . .	78
5.4.2 Creating the schema for the OLTP model . . . . .	80
5.4.3 Data from legacy data sources . . . . .	81
5.4.4 Simulating the transactional environment . . . . .	84
5.5 The operational and dimensional data model . . . . .	85
5.6 Referential integrity for a data warehouse . . . . .	87
5.7 Data modeling options: IBM Industry Models . . . . .	88
<b>Chapter 6. The system environment . . . . .</b>	<b>91</b>
6.1 Implemented architecture . . . . .	92
6.2 System configuration . . . . .	93
6.3 System parameters of DB2 subsystems . . . . .	94
6.4 Workload Manager configuration . . . . .	95
6.5 zIIP utilization . . . . .	96
<b>Chapter 7. Functions in DB2 for z/OS for a data warehouse . . . . .</b>	<b>99</b>
7.1 Index compression . . . . .	100
7.1.1 How index compression works . . . . .	100
7.1.2 Implementation guidelines . . . . .	102
7.1.3 Implementation examples . . . . .	107
7.1.4 Considerations on index compression . . . . .	118
7.2 Table space compression . . . . .	118
7.2.1 Considerations on data compression . . . . .	120
7.3 Index-use tracking by using real-time statistics . . . . .	121
7.4 Not logged table spaces . . . . .	121
7.5 Exploiting the DB2 STATEMENT CACHE . . . . .	123
7.6 Star schema processing . . . . .	124
7.6.1 Star schema access methods . . . . .	125
7.6.2 Star schema processing implementation example . . . . .	129
7.7 Index on expressions . . . . .	139
7.8 Working with the ADD CLONE SQL command . . . . .	143
7.8.1 Operating cloned objects . . . . .	144
7.9 Table space partitioning . . . . .	148
7.9.1 Universal table space . . . . .	149

7.10	Materialized query tables	152
7.10.1	When to consider an MQT	152
7.10.2	MQTs used in our scenario	153
7.11	OLAP functions	154
7.11.1	RANK and DENSE_RANK	154
7.11.2	ROW_NUMBER	157
<b>Chapter 8.</b>	<b>Q replication and event publishing</b>	<b>159</b>
8.1	Introduction to replication functions	160
8.1.1	Q replication	160
8.1.2	Event publishing	162
8.2	Implementation of Q replication and Event Publisher	162
8.2.1	Common infrastructure	163
8.2.2	Configuring Q replication subscriptions	171
8.2.3	Configuring event publishing using the DB2 Replication Center	183
8.3	Operating Q replication and event publishing	195
<b>Chapter 9.</b>	<b>Setting up ETL components for a data warehouse</b>	<b>201</b>
9.1	Overview of components for ETL on Linux on System z and z/OS	202
9.2	Configuring a HiperSocket connection to z/OS on Linux on System z	203
9.3	Setting up BatchPipes	206
9.4	Setting up WebSphere Classic Federation	210
9.4.1	Setting up the WebSphere Classic Federation server on z/OS	212
9.4.2	Defining and registering the flat file in Classic Data Architect	215
9.4.3	Installing and configuring the WebSphere Classic Federation client on Linux on System z	226
9.5	Installing IBM Information Server	228
9.5.1	Topology considerations: Product tiers	229
9.5.2	User ID and registry considerations before installing Information Server	231
9.5.3	Installing server components for Linux on System z	231
9.5.4	Installing DataStage and QualityStage designer clients on Windows	242
9.5.5	Configuring DataStage to access DB2 for z/OS, WebSphere MQ, and WebSphere Classic Federation	245
9.5.6	Cataloging the DB2 for z/OS subsystems	248
9.5.7	Setting up ODBC connections to DB2 for z/OS databases and WebSphere Classic Federation	248
9.5.8	Troubleshooting configuration problems	250
9.5.9	Granting user access and creating a new DataStage project	251
9.5.10	Defining multiple nodes for parallel execution	254
<b>Chapter 10.</b>	<b>Full load using DataStage</b>	<b>257</b>
10.1	ETL data in our scenario	258
10.2	Loading overview	260
10.2.1	Full load description	261
10.2.2	Incremental load description	261
10.2.3	Lookup tables	261
10.3	Load jobs for populating ODS from OLTP	262
10.3.1	DB2z stage: Reading DB2 data	262
10.3.2	DB2z stage: Writing DB2 data	262
10.3.3	Parallel jobs and BatchPipes	263
10.3.4	Sample load jobs from OLTP to ODS	264
10.4	Load jobs for populating a DDS from an ODS	266
10.4.1	Load of the Date dimension table	266
10.4.2	Surrogate key stage utilization	267

10.4.3 Load of the fact table . . . . .	269
10.5 Accessing WebSphere Classic Federation in DataStage jobs . . . . .	271
10.6 Running and monitoring an ETL job in DataStage Director . . . . .	273
10.6.1 Typical load challenges. . . . .	273
10.6.2 Hints for table organization while preparing for loading data . . . . .	274
10.7 Debugging load jobs: A brief look . . . . .	274
10.8 Performance considerations . . . . .	276
10.8.1 Performance statistics. . . . .	276
10.8.2 Parallel jobs versus server jobs . . . . .	277
10.8.3 Choice of stage for performance improvement. . . . .	278
10.9 Naming standards. . . . .	279
10.10 Data quality implementation during full load . . . . .	279
<b>Chapter 11. Incremental update with DataStage . . . . .</b>	<b>281</b>
11.1 Operational BI revisited. . . . .	282
11.2 Reduction of a batch window . . . . .	283
11.3 Usage of queues as sources of data. . . . .	283
11.4 Anatomy of a queue . . . . .	284
11.5 DataStage job to read a queue . . . . .	285
11.5.1 WebSphere MQ Connector stage. . . . .	285
11.5.2 ODBC Enterprise stage . . . . .	286
11.6 Automated reading of the queues and updating of the data warehouse. . . . .	286
11.6.1 Update of the DWHODS Customer table . . . . .	287
11.6.2 Update of the DWHDDS Customer table . . . . .	289
11.7 Concurrency considerations . . . . .	292
11.8 Summary. . . . .	293
<b>Chapter 12. An operational business intelligence implementation . . . . .</b>	<b>295</b>
12.1 OLTP application with embedded analytics . . . . .	296
12.2 The order processing Web application . . . . .	297
12.3 Implementation considerations . . . . .	299
12.3.1 Handling multiple data sources. . . . .	299
12.3.2 Leveraging tools to render business intelligence data . . . . .	300
12.4 Improving response times by using materialized query tables . . . . .	300
<b>Chapter 13. Reporting and analysis with Cognos 8 BI . . . . .</b>	<b>305</b>
13.1 Overview of Cognos 8 BI components . . . . .	306
13.2 Installing Cognos 8 BI Server on Linux on System z . . . . .	308
13.2.1 Topology overview . . . . .	308
13.2.2 System considerations and version checks . . . . .	309
13.2.3 Preparation for installation . . . . .	311
13.2.4 Installing Cognos BI Server components . . . . .	313
13.2.5 Configuring the IBM HTTP Server . . . . .	316
13.2.6 Configuring and starting the Cognos BI Server. . . . .	317
13.2.7 Defining DB2 for z/OS data sources. . . . .	323
13.2.8 Configuring Cognos with WebSphere Application Server. . . . .	326
13.3 Building Cognos data models and packages . . . . .	335
13.3.1 Framework Manager packages. . . . .	337
13.3.2 Defining Transformer models for PowerCubes. . . . .	352
13.3.3 Creating and publishing OLAP PowerCubes . . . . .	363

13.4 Reports with Report Studio . . . . .	367
13.5 Ad hoc queries with Query Studio . . . . .	374
13.6 Multidimensional analysis with Analysis Studio . . . . .	385
<b>Chapter 14. Reporting with DataQuant, QMF, and AlphaBlox . . . . .</b>	<b>397</b>
14.1 DataQuant . . . . .	398
14.1.1 When to consider DataQuant . . . . .	398
14.1.2 DataQuant functions . . . . .	398
14.1.3 A small DataQuant report for RedParts Distribution . . . . .	400
14.2 QMF . . . . .	409
14.3 AlphaBlox . . . . .	411
14.3.1 When to consider AlphaBlox . . . . .	411
14.3.2 AlphaBlox functions . . . . .	411
14.3.3 A small AlphaBlox Web application for RedParts Distribution . . . . .	412
14.3.4 Recommendations to configure AlphaBlox on Linux on System z . . . . .	421
<b>Part 3. Appendixes . . . . .</b>	<b>423</b>
<b>Appendix A. Index compression jobs . . . . .</b>	<b>425</b>
<b>Appendix B. Schema definitions . . . . .</b>	<b>431</b>
B.1 Schema definition for the transactional database . . . . .	432
B.2 Schema definition for the data warehouse database . . . . .	437
<b>Appendix C. Additional material . . . . .</b>	<b>447</b>
C.1 Locating the Web material . . . . .	447
C.2 Using the Web material . . . . .	447
C.2.1 System requirements for downloading the Web material . . . . .	448
C.2.2 How to use the Web material . . . . .	448
<b>Related publications . . . . .</b>	<b>449</b>
IBM Redbooks . . . . .	449
Other publications . . . . .	449
Online resources . . . . .	450
How to get Redbooks . . . . .	451
Help from IBM . . . . .	451
<b>Index . . . . .</b>	<b>453</b>

Archived

# Figures

1-1 Data warehouse and BI environment . . . . .	5
1-2 Example of a BI architecture. . . . .	8
1-3 Sample incremental process. . . . .	14
2-1 Operational intelligence introduction and user population . . . . .	22
2-2 BI tools - User associations . . . . .	23
2-3 Information On Demand logical architecture. . . . .	25
2-4 Performance management cycle . . . . .	27
2-5 Example components for a DB2 for z/OS operational BI solution. . . . .	29
3-1 Parallel Sysplex architecture. . . . .	33
3-2 DB2 data sharing architecture . . . . .	34
3-3 GDPS/PPRC. . . . .	39
3-4 Parallel access volumes . . . . .	44
3-5 HyperPAV . . . . .	45
3-6 The specialty engines . . . . .	46
3-7 Evolution of the System z platform . . . . .	47
3-8 Data sharing and mixed workload. . . . .	52
3-9 WebSphere Classic Federation . . . . .	54
3-10 WebSphere Classic Data Event Publisher . . . . .	55
3-11 Event publishing and Q Capture . . . . .	56
3-12 Information Server for System z . . . . .	57
4-1 Data flow across components. . . . .	61
4-2 Data warehouse architecture on System z with one data sharing groups . . . . .	64
4-3 SYSPLEX with two LPARs and two data sharing groups . . . . .	66
5-1 Conceptual overview for scenario. . . . .	76
5-2 Transactional schema (OLTP_B and OLTP_W). . . . .	78
5-3 Transactional schema (OLTP) based on the TPC-H data model . . . . .	79
5-4 The transactional data model . . . . .	80
5-5 Web application simulating OLTP workload for the transactional environment . . . . .	84
5-6 Model for the operational data store, based on TPC-H . . . . .	85
5-7 Dimensional data model . . . . .	86
6-1 Implemented architecture . . . . .	92
6-2 Data warehouse hardware . . . . .	94
6-3 RMF Workload Activity Report . . . . .	96
7-1 B-tree schematic representation . . . . .	101
7-2 Data and index compression compared at run time . . . . .	102
7-3 Maximum savings on disk in function of the buffer pool, assuming ideal index compression . . . . .	103
7-4 Impact of buffer pool selection for a compression ratio of 4:1 . . . . .	105
7-5 Access path involving a compressed index . . . . .	114
7-6 Index only scan on an uncompressed index. . . . .	117
7-7 Exceptional processing and the NOT LOGGED option . . . . .	122
7-8 Star schema representation . . . . .	125
7-9 Star schema implementation sample . . . . .	129
7-10 Query access path before activating star schema processing . . . . .	132
7-11 Access path changes after enabling a star join . . . . .	135
7-12 Star join layout in Optimization Service Center. . . . .	135
7-13 Optimization Service Center star join dimension node type . . . . .	136
7-14 ADD CLONE process schematic representation . . . . .	144

7-15 Results for the top four customers for each quarter . . . . .	155
7-16 Results from DENSE_RANK expression . . . . .	156
7-17 Results from ROW_NUMBER expression . . . . .	157
8-1 Q replication overview. . . . .	161
8-2 Event publishing overview . . . . .	162
8-3 DB2 Replication Center launchpad. . . . .	170
8-4 DB2 Replication Center main panel . . . . .	171
8-5 Replication Center Launchpad for Q replication . . . . .	172
8-6 Specifying the type of replication . . . . .	173
8-7 Defining the source and target servers . . . . .	174
8-8 Creation of a Q replication queue map . . . . .	175
8-9 Options for the Q replication queue map . . . . .	176
8-10 Defined source and target servers . . . . .	177
8-11 Selecting the source tables for Q replication . . . . .	177
8-12 Profile settings for target tables . . . . .	178
8-13 Manage Target Object Profiles window . . . . .	179
8-14 Mapping source to target columns . . . . .	180
8-15 Q Apply management of unexpected conditions . . . . .	181
8-16 Q replication options for initial load . . . . .	182
8-17 Review and complete subscription definition panel . . . . .	182
8-18 Event publishing - Replication Center Launchpad . . . . .	183
8-19 Creating publications for event publishing . . . . .	184
8-20 Sample of a publishing queue map creation. . . . .	185
8-21 Event publishing subscription creation . . . . .	186
8-22 Selecting source tables for event publishing. . . . .	187
8-23 Selecting rows for event publishing . . . . .	188
8-24 Event publishing - Review and complete publications panel . . . . .	189
8-25 Updating Publishing Queue Maps . . . . .	191
8-26 Changing queue map properties. . . . .	192
8-27 Starting an inactive event publishing subscription . . . . .	196
9-1 Components setup for ETL . . . . .	202
9-2 YaST Control Center. . . . .	204
9-3 Network Card Configuration Overview . . . . .	205
9-4 Assigning the static IP address. . . . .	205
9-5 HiperSocket connection displayed . . . . .	206
9-6 Interacting with batch pipes . . . . .	208
9-7 Data server configuration in Classic Architect . . . . .	215
9-8 Supplier copybook . . . . .	216
9-9 Mapping a sequential table. . . . .	217
9-10 Parameters for the sequential table . . . . .	218
9-11 Sequential table mapping - Table name and dataset name . . . . .	219
9-12 Summary of fields . . . . .	220
9-13 Generate DDL . . . . .	221
9-14 Updated SUPPLIER sequential table DDL . . . . .	222
9-15 Running the DDL on the data server . . . . .	223
9-16 Running the DDL on the data server . . . . .	224
9-17 Supplier sequential table contents . . . . .	225
9-18 Installation wizard for WebSphere Classic Federation client installation . . . . .	226
9-19 Selecting installation options for the WebSphere Classic Federation client . . . . .	227
9-20 Installation directory for the WebSphere Classic Federation client. . . . .	227
9-21 Confirmation message for a successful client installation. . . . .	227
9-22 Information Server 8.0.1 tiers . . . . .	230
9-23 Welcome panel for the Information Server installation . . . . .	232



9-24	Installation option to create a response file . . . . .	232
9-25	Choosing the installation directory . . . . .	233
9-26	Selecting the server tiers for installation . . . . .	233
9-27	Selecting the product components to install . . . . .	234
9-28	Selecting the installation type . . . . .	234
9-29	Message about DB2 installation . . . . .	235
9-30	Specifying the user ID to create and access the repository . . . . .	236
9-31	Installation option for WebSphere Application Server . . . . .	237
9-32	User registry selection and WebSphere Application Server user ID . . . . .	238
9-33	Selecting the admin user ID for the Information Server Web console . . . . .	238
9-34	DataStage project list during installation . . . . .	239
9-35	Defining a new DataStage project during installation . . . . .	239
9-36	Installation location for DB2 for Linux on System z . . . . .	240
9-37	Defining the InfoSphere DataStage administrator user . . . . .	240
9-38	Name and port number for the DataStage instance . . . . .	241
9-39	Summary before installation of DataStage and QualityStage . . . . .	242
9-40	Starting the installation of DataStage and QualityStage client on Windows . . . . .	243
9-41	Installation option to create a response file . . . . .	243
9-42	Choosing the installation directory . . . . .	244
9-43	Installation options . . . . .	244
9-44	Pre-installation summary . . . . .	245
9-45	Editing the LD_LIBRARY_PATH for a DataStage project . . . . .	247
9-46	Granting use of DataStage . . . . .	251
9-47	Assigning roles for DataStage and QualityStage . . . . .	252
9-48	Logging in to DataStage . . . . .	253
9-49	Adding a new project . . . . .	253
9-50	Adding a user of a group . . . . .	254
9-51	Selecting the configuration file environment variable for the job properties . . . . .	256
9-52	Job properties with the configuration file as a parameter . . . . .	256
9-53	Selecting a different configuration for running a DataStage job . . . . .	256
10-1	Transforming customer data from OLTP to ODS . . . . .	258
10-2	Transforming order and line-item data from OLTP to ODS . . . . .	259
10-3	Transforming data from the ODS to the DDS . . . . .	260
10-4	DB2z stage properties for reading data from DB2 for z/OS . . . . .	262
10-5	DB2z stage properties for writing data to DB2 for z/OS . . . . .	263
10-6	BatchPipe utility . . . . .	264
10-7	Sample simple job from OLTP to ODS . . . . .	264
10-8	Sample simple job from OLTP to ODS successfully completed . . . . .	265
10-9	Population of ODS Customer table from the OLTP Customer table . . . . .	265
10-10	Load of DWHODS.ORDERS table from the necessary OLTP tables . . . . .	266
10-11	Date dimension load . . . . .	267
10-12	Properties of the Surrogate key generator stage . . . . .	268
10-13	Use of the surrogate key generator stage in loading data . . . . .	268
10-14	Sample surrogate generator stage values . . . . .	269
10-15	Order transaction fact table load . . . . .	270
10-16	Completed Order transaction fact table load . . . . .	270
10-17	Job log report of reject files during Order transaction fact table load . . . . .	270
10-18	Accessing legacy data in a DataStage job . . . . .	271
10-19	Table definition for supplier data . . . . .	272
10-20	Debugging job example . . . . .	275
10-21	Peek output for debugging - An example . . . . .	275
10-22	DataStage job performance statistics . . . . .	276
10-23	DataStage job CPU utilization . . . . .	277

10-24	Combining pipeline and partitioning parallelism . . . . .	278
10-25	DataStage job parameters for database name and credentials . . . . .	279
11-1	MQ message definition link. . . . .	284
11-2	Sample WebSphere MQ Connector stage entries . . . . .	285
11-3	ODBC Enterprise stage settings . . . . .	286
11-4	Read message queue to update ODS Customer table. . . . .	287
11-5	Message column Payload split into consisting columns . . . . .	288
11-6	Job log from update of the DWHODS customer table . . . . .	288
11-7	SCD implementation example . . . . .	289
11-8	SCD Lookup of dimension table . . . . .	290
11-9	SCD Surrogate Key settings. . . . .	290
11-10	SCD dimensional update mapping . . . . .	291
11-11	SCD output to file . . . . .	291
11-12	SCD result with the effective date. . . . .	292
11-13	SCD result continued with the end date and recent flag. . . . .	292
11-14	Workload Manager helping to prioritize for operational BI . . . . .	292
12-1	Order processing application that accesses both OLTP and data warehouse data . . . . .	296
12-2	OLTP application for order processing . . . . .	297
12-3	OLTP application with BI information . . . . .	299
12-4	Access plan for data warehouse query in order processing application without MQT. . . . .	301
12-5	Costs of the data warehouse query without MQT. . . . .	301
12-6	Access plan for data warehouse query in order processing application using MQT. . . . .	304
12-7	Costs of the data warehouse query using MQT . . . . .	304
13-1	Example Cognos 8 BI dashboard portlets . . . . .	307
13-2	Topology overview for Cognos installation . . . . .	308
13-3	Creating a dedicated user ID for Cognos installation and administration . . . . .	312
13-4	Welcome panel of the installation wizard for Cognos BI Server . . . . .	313
13-5	License agreement for Cognos BI Server. . . . .	313
13-6	Installation location for Cognos BI Server. . . . .	314
13-7	Component selection for Cognos BI Server . . . . .	314
13-8	Summary before installation of Cognos BI Server . . . . .	315
13-9	Completion notice after Cognos BI Server installation . . . . .	315
13-10	Installing Cognos 8 Transformer. . . . .	316
13-11	Error message with incompatible JRE . . . . .	318
13-12	Deleting an existing setting for the content store . . . . .	320
13-13	Creating a new database for the content store . . . . .	320
13-14	Configuring and testing the DB2 database for the Cognos content store. . . . .	321
13-15	Operations on first touch for the Cognos content store . . . . .	321
13-16	Setting port numbers for the Web server gateway . . . . .	322
13-17	Starting the Cognos BI Server . . . . .	322
13-18	Data source connections in Cognos Administration . . . . .	323
13-19	Naming a new data source . . . . .	324
13-20	Specifying the type for the new data source . . . . .	324
13-21	Database name and signon parameters for the data source . . . . .	325
13-22	Testing the connection to the new data source . . . . .	325
13-23	Data source list with DWHD911 and DWHD912. . . . .	325
13-24	Navigating to JVM settings for the application server . . . . .	328
13-25	Adding LD_LIBRARY_PATH environment settings . . . . .	329
13-26	Starting to build application files for deployment in cogconfig . . . . .	329
13-27	Selecting the applications to build. . . . .	330
13-28	Settings for the p2pd EAR file. . . . .	330
13-29	Report about successful creation of the p2pd EAR file. . . . .	331
13-30	Changing the dispatcher port number. . . . .	331

13-31	Specifying the remote file system . . . . .	332
13-32	Installation options for the Cognos 8 application . . . . .	333
13-33	Cognos configuration . . . . .	336
13-34	Cognos configuration test . . . . .	337
13-35	Creating a new Framework Manager project . . . . .	339
13-36	Manually running the Run Metadata Wizard . . . . .	340
13-37	Cognos 8 defined data source list . . . . .	341
13-38	Metadata wizard - Selecting objects . . . . .	342
13-39	Database View - Imported metadata . . . . .	344
13-40	Example query object within the Business view namespace . . . . .	345
13-41	Example of a Dimensional view . . . . .	346
13-42	Defining relationships between query subjects . . . . .	347
13-43	Single determinant definition . . . . .	348
13-44	Multiple determinant definitions . . . . .	349
13-45	Defining a business filter . . . . .	350
13-46	Defining a calculated column . . . . .	351
13-47	Defining packages to publish . . . . .	352
13-48	Defining Transformer model data source . . . . .	353
13-49	Query definition in Transformer . . . . .	354
13-50	Data class definition for source fields . . . . .	355
13-51	Insert dimensions . . . . .	356
13-52	Defining the general properties for the time dimension . . . . .	357
13-53	Defining the time properties for the time dimension . . . . .	358
13-54	Adding the Branch Name for further role associations . . . . .	359
13-55	Completed role associations . . . . .	360
13-56	Defining an alternate drill-down path . . . . .	361
13-57	Defining a calculated measure . . . . .	362
13-58	Completed Transformer model . . . . .	363
13-59	Starting to create a new data source for the Power Cube . . . . .	365
13-60	Selecting the data source type . . . . .	365
13-61	Specifying the path to the mdc file for the PowerCube definition . . . . .	366
13-62	Creating a new package for analysis . . . . .	367
13-63	Starting the Report Studio from the My Actions panel . . . . .	369
13-64	Package selection for the new report . . . . .	369
13-65	Report Studio welcome panel . . . . .	369
13-66	Report template selection in Report Studio . . . . .	370
13-67	Defining a new SQL-based query object . . . . .	370
13-68	Properties for the SQL query in Report Studio . . . . .	371
13-69	Setting query hints . . . . .	371
13-70	Selecting data items in Report Studio . . . . .	372
13-71	Adding a text item to the report . . . . .	372
13-72	formatting the table heading . . . . .	373
13-73	Running the defined report . . . . .	373
13-74	Result of the Returned Item report . . . . .	374
13-75	Navigating to Query Studio . . . . .	375
13-76	Query - Selecting the source package . . . . .	376
13-77	Query - Dragging the order query items . . . . .	377
13-78	Query - Dragging the customer nation . . . . .	378
13-79	Query - Grouping by nation . . . . .	379
13-80	Query - Sorting on descending order . . . . .	380
13-81	Query - Adding a query filter . . . . .	381
13-82	Query - Adding a column filter . . . . .	382
13-83	Query - Modifying the column filter . . . . .	383

13-84	Query - Adding the conditional formatting. . . . .	384
13-85	Query - Viewing the report as a PDF . . . . .	385
13-86	Launch analysis studio . . . . .	386
13-87	Analysis - Blank analysis . . . . .	386
13-88	Analysis - Selecting rows . . . . .	387
13-89	Analysis - Selecting columns . . . . .	388
13-90	Analysis - Selecting a measure. . . . .	389
13-91	Analysis - Selecting a filter . . . . .	390
13-92	Analysis - Including charts . . . . .	391
13-93	Analysis - Adding a title. . . . .	392
13-94	Analysis - Saving the layout . . . . .	393
13-95	Analysis - Completed RedParts sample 1 . . . . .	394
13-96	Analysis - Building RedParts sample 2. . . . .	395
13-97	Analysis - Removing the subtotals . . . . .	396
14-1	DataQuant dashboard. . . . .	399
14-2	DataQuant - Defining the source data . . . . .	401
14-3	DataQuant - Displaying the table . . . . .	402
14-4	DataQuant Diagram Query Builder . . . . .	403
14-5	DataQuant Prompted Query Builder. . . . .	404
14-6	DataQuant SQL query editor . . . . .	405
14-7	DataQuant tabular result . . . . .	406
14-8	DataQuant - Visual report design view . . . . .	407
14-9	DataQuant - Design view, adding the bar chart . . . . .	408
14-10	DataQuant - RedParts report . . . . .	409
14-11	AlphaBlox - Defining the data source . . . . .	413
14-12	AlphaBlox - Creating the application. . . . .	414
14-13	WebSphere Administrative Console - Deploying the application . . . . .	415
14-14	WebSphere Administrative Console - Defining user roles for the application. . . . .	416
14-15	AlphaBlox list of applications . . . . .	417
14-16	AlphaBlox Query Builder. . . . .	418
14-17	AlphaBlox Blox builder output. . . . .	419
14-18	AlphaBlox - Output of RedParts JSP . . . . .	421
14-19	WebSphere Administrative Console - Java environment for RedParts application . . . . .	422

# Tables

2-1	Examples of current BI trends. . . . .	20
2-2	BI evolution examples. . . . .	21
2-3	Information On Demand architectural principles. . . . .	25
3-1	Impact of workload management to a mixed workload environment . . . . .	35
5-1	Scenario data sources . . . . .	72
5-2	Sample scenario deliverables. . . . .	73
5-3	Demonstrated DB2 functionality . . . . .	77
5-4	PART table format . . . . .	82
5-5	Supplier table format. . . . .	83
5-6	PART SUPPLIER table format . . . . .	83
6-1	DSNZPARM definitions for a data warehouse . . . . .	94
7-1	Comparing index and data compression . . . . .	101
7-2	Index compression performance compared for index scan operations. . . . .	118
7-3	Sample PLAN_TABLE extract before activating star schema processing . . . . .	132
7-4	Sample PLAN_TABLE extract after activating star schema processing. . . . .	135
8-1	DEL message format - Message header . . . . .	190
8-2	DEL message format - Data from the source table. . . . .	190
9-1	Required APARs for BatchPipes support . . . . .	209
9-2	The Information Server user IDs. . . . .	231
10-1	WebSphere Client Federation properties . . . . .	272
12-1	Handling application requirements when using multiple DB2 for z/OS subsystems . . . . .	300
13-1	Product components . . . . .	309
13-2	Models and packages. . . . .	338

Archived

# Examples

5-1 Table space DDL .....	80
5-2 Table DDL.....	80
5-3 PART data .....	81
5-4 Supplier data.....	82
5-5 PART SUPPLIER data .....	83
6-1 Service class definitions for the OLTP DB2 subsystem .....	95
6-2 Service class definitions for the data warehouse DB2 subsystem .....	96
7-1 DSN1COMP JCL sample .....	106
7-2 DSN1COMP output sample .....	106
7-3 SQL extract - Creating and altering an index with the COMPRESS option .....	107
7-4 Creating a compressed index is not supported on 4K buffer pools .....	107
7-5 Altering an index defined on a 4K buffer pool to COMPRESS YES .....	108
7-6 Index in REBUILD PENDING after alter to COMPRESS YES or COMPRESS NO ..	108
7-7 Sample catalog query for index compression analysis.....	108
7-8 STOSPACE JCL sample .....	109
7-9 Sample index for compression impact analysis .....	109
7-10 Index compression query output sample .....	109
7-11 Customer table, TPC-H model data sample .....	110
7-12 Sample index creation for analysis of column position influence .....	111
7-13 Index comparison - Influence on compression ratio of column order .....	111
7-14 DSN1PRNT JCL sample.....	111
7-15 Print of first leaf page for index NAME - KEY .....	112
7-16 Print of first leaf page for index KEY - NAME .....	112
7-17 How data is stored with index compression .....	112
7-18 TPC-H order priority checking query.....	113
7-19 Index for compression test - Physical properties .....	113
7-20 JCL sample for issuing a MODIFY TRACE command .....	115
7-21 OMEGAMON XE for DB2 Performance Expert on z/OS command report sample. .	115
7-22 Performance Expert report sample using a non-compressed index.....	116
7-23 Performance Expert report sample using a compressed index .....	116
7-24 Sample query index only scan .....	117
7-25 Index creation for index only scan of compressed index test .....	117
7-26 Details for indexes involved in index only scan.....	118
7-27 DSN1COMP compression report .....	119
7-28 Altering a table space for using compression .....	119
7-29 Compression Report of REORG utility .....	120
7-30 Alter table space NOT LOGGED syntax.....	122
7-31 Sample query against a not logged table space.....	122
7-32 ICOPY status of a not logged table space after an update operation.....	122
7-33 Index space must be copied with not logged table spaces.....	123
7-34 Not logged table space and load nocopypend .....	123
7-35 Populating the DSN_STATEMENT_CACHE_TABLE.....	124
7-36 Sample query candidate for star join .....	126
7-37 Optimization Service Center report - Default values for STARJOIN and SJTABLES	127
7-38 Sample query candidate for a star join .....	130
7-39 Creating a multicolumn index in the fact table for star join processing .....	130
7-40 Defining indexes for each dimension key column in the fact table .....	131
7-41 System parameters implementation for start join .....	133

7-42	Online activation of star schema processing parameters . . . . .	133
7-43	Deactivating star schema processing parameters by reloading startup zparms. . . .	134
7-44	Optimization Service Center report on changed values for STARJOIN, SJTABLES	134
7-45	Compression on a star schema index. . . . .	137
7-46	DB2 OMEGAMON/PE Record trace - IFCID 002 - STARJOIN pool statistics . . . .	138
7-47	Index creation DDL - Index on expression . . . . .	139
7-48	Stage 2 predicate query example . . . . .	140
7-49	Index on expression creation example . . . . .	140
7-50	Index on expression cannot contain aggregate functions. . . . .	141
7-51	Creating an index on expression . . . . .	141
7-52	Runstats output for an index on expression . . . . .	141
7-53	Sample query from TCP-H for index on expression implementation . . . . .	142
7-54	Creating an index on expression . . . . .	142
7-55	SQL sample of creating a CLONE table of CUSTOMER . . . . .	145
7-56	Sample query for inquiring clone tables information from the DB2 catalog . . . . .	145
7-57	Sample DB2 catalog information about cloned tables . . . . .	145
7-58	Display of a database that contains cloned objects . . . . .	146
7-59	View of VSAM data sets underlying a cloned object. . . . .	146
7-60	Exchange data execution sample . . . . .	146
7-61	Rollback of an exchange data operation. . . . .	147
7-62	Exchange data and operations involving several clones . . . . .	147
7-63	Drop clone. . . . .	148
7-64	DDL for partition-by-growth table space . . . . .	150
7-65	DDL for creating a partition-by-range table space . . . . .	151
7-66	Partition-by-range table definition . . . . .	151
7-67	Adding a partition . . . . .	152
7-68	Creating MQT . . . . .	153
7-69	Dropping MQT . . . . .	154
7-70	RANK expression . . . . .	155
7-71	DENSE_RANK expression . . . . .	156
7-72	ROW_NUMBER . . . . .	157
8-1	Defining a user and started tasks for replication server . . . . .	163
8-2	Q Apply DB2 authority failure report . . . . .	164
8-3	Procedure Sample for Q Capture . . . . .	164
8-4	Procedure sample for Q Capture . . . . .	165
8-5	Started tasks for Q replication. . . . .	165
8-6	MQSeries queues used in our implementation . . . . .	166
8-7	Creating common MQSeries objects . . . . .	166
8-8	MQ queue creation output sample . . . . .	167
8-9	JCL sample for the creation of the Q replication queue . . . . .	168
8-10	Creating MQSeries queues for Q replication and event publishing . . . . .	168
8-11	WHERE clause on table rows for event publishing. . . . .	188
8-12	Message format . . . . .	190
8-13	Event publishing message browse REXX code sample . . . . .	193
8-14	JCL sample for event publishing message browse. . . . .	194
8-15	Browsing a delimited event publishing message . . . . .	194
8-16	Browsing a XML event publishing message . . . . .	194
8-17	Sample Q Capture output for event publishing. . . . .	195
8-18	Q Capture output - A subscription being started. . . . .	196
8-19	REXX sample for formatting the contents of the IBMQREP_APPLYTRACE table. .	196
8-20	Output sample of formatted information in the IBMQREP_APPLYTRACE table . .	198
8-21	Getting information from USS. . . . .	198
8-22	Browsing Q Capture log using OEDIT . . . . .	199



9-1	IOCDs definition for HiperSocket channels	203
9-2	Using the ping command to verify the HiperSocket connection	206
9-3	Log output for the FTP command issued by a DataStage DB2z stage	207
9-4	Log output for a load utility call though DSNUTILS.	207
9-5	BatchPipes subsystem status	208
9-6	Listing of currently opened batch pipes in the system	209
9-7	Error message in the log if PK54242 is missing	210
9-8	Data server procedure	213
9-9	Query processor definition	213
9-10	TCP/IP connection handler configuration	214
9-11	SAF Exit	214
9-12	Data server job log	214
9-13	The cac.ini file	228
9-14	Modifications in the dsenv file	246
9-15	Exception in DataStage if MQ libraries are not accessible	247
9-16	Cataloging the DB2 subsystems	248
9-17	ODBC data source definitions in file .odbc.ini	248
9-18	The uvodbc.config file with settings for available ODBC data sources	249
9-19	DB2 commands to bind DataStage packages and grant access to public	250
9-20	Output of ddtestlib	250
9-21	Using dssh to verify the ODBC connection to the Classic Federation Server	251
9-22	Defining an additional node	255
9-23	Configuration with an additional node	255
10-1	Error message with DB2z stage and FTP to a data set	273
10-2	BatchPipe related error messages	273
12-1	Query for most sold parts in the country of a given customer	298
12-2	Query for returned items per customer	298
12-3	MQT definition that supports popular parts query	302
12-4	Rewritten query using the defined MQT	302
12-5	JDBC code fragment to set the current refresh age value	303
13-1	Using netstat to list used ports	310
13-2	Using db2level to display DB2 for Linux, UNIX, and Windows version information	310
13-3	Using versionInfo.sh to check the WebSphere Application server version	310
13-4	Output of Java version check for the supported 31-bit version	311
13-5	Output of Java version check for the unsupported 64-bit version	311
13-6	Additional .profile settings for the Cognos user	312
13-7	Sample CLASSPATH setting for including DB2 JDBC driver JAR files	312
13-8	Adding Cognos directories to LD_LIBRARY_PATH	316
13-9	Modifications in httpd.conf	316
13-10	HTTP daemons for the Cognos BI Web server gateway	317
13-11	Running the C8DB2.sh script to create a content store database	318
13-12	Setting the JRE in .profile	319
13-13	Cataloguing remote DB2 for z/OS databases	323
13-14	Profile settings for user cognos	326
13-15	Java version of the JDK that ships with WebSphere Application Server	326
13-16	Running the create_profile script to create a new WebSphere Application Server profile	327
13-17	Starting the server instance	327
13-18	Installation report of p2pd.ear	333
13-19	Output in SystemOut.log indicating that dispatcher is ready to process requests	334
13-20	SystemOut.log messages during servlet startup	334
13-21	Creating the PowerCube on the Server	364
13-22	Returned item query	368

14-1 Sample query for DataQuant .....	403
14-2 RedParts home page .....	417
14-3 AlphaBlox sample query .....	419
14-4 RedParts JSP .....	420
A-1 ASM program used for performance evaluation .....	425
A-2 JCL sample for assembly, bind and execution of performance model program .....	428
B-1 Schema definition for the transactional system .....	432
B-2 ODS schema definitions .....	437
B-3 DDS schema definitions .....	441

# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.*

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.


## COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

# Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®	FlashCopy®	Redbooks®
AlphaBlox®	GDPS®	Redbooks (logo)  ®
BatchPipes®	Geographically Dispersed Parallel Sysplex™	REXX™
Blox®	HiperSockets™	RMF™
CICS®	HyperSwap™	Service Director™
DataStage®	IBM®	SupportPac™
DB2 Connect™	IMST™	Sysplex Timer®
DB2®	Informix®	System i™
DFSMS™	InfoSphere™	System Storage™
DFSMS/MVS™	MQSeries®	System z™
DFSMSHsm™	MVS™	System z10™
DFSORT™	MVS/ESA™	System z9®
Distributed Relational Database Architecture™	OMEGAMON®	Tivoli®
DRDA®	OS/390®	VTAM®
DS8000™	Parallel Sysplex®	WebSphere®
Enterprise Storage Server®	pSeries®	z/OS®
ESCON®	pureXML™	z/VM®
eServer™	QMFTM	z10™
FICON®	RACF®	z9™
		zSeries®

The following terms are trademarks of other companies:

Cognos, PowerCube, and the Cognos logo are trademarks or registered trademarks of Cognos Incorporated, an IBM Company, in the United States and/or other countries.

SUSE, the Novell logo, and the N logo are registered trademarks of Novell, Inc. in the United States and other countries.

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

J2EE, Java, JavaScript, JavaServer, JDBC, JDK, JRE, JSP, JVM, Solaris, Sun, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Excel, Internet Explorer, Microsoft, SQL Server, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

# Preface

Enterprises look more and more to business intelligence (BI) to gain a competitive edge. Today's BI systems incorporate large data warehouses that are consolidated with near real-time operational data stores (ODS) and continuously updated from multiple sources. An increasing number of users in the enterprise want to access the data warehouse with BI applications with real-time needs.

There is a renewed interest in the ability to implement a data warehouse solution on DB2® for z/OS® and System z™. This is due to the inherent characteristics of security, availability, performance, mixed workload management, and the growing portfolio of data warehousing tools and functions provided by IBM®.

In this IBM Redbooks® publication, we focus on today's software components on System z and show how you can use them to realize the infrastructure for a full data warehouse solution. By using a retail business scenario loosely based on the TPC-H benchmark, we guide you through the warehouse implementation steps. In addition, we highlight the available methods, techniques, and technologies for the deployment of this solution:

- ▶ System z and Linux® on System z
- ▶ DB2 9 for z/OS
- ▶ InfoSphere™ DataStage® and InfoSphere QualityStage
- ▶ WebSphere® Q Replication and WebSphere Data Event Publisher
- ▶ Cognos® 8 BI

This book provides an opportunity for you to look at satisfying the operational needs of corporate users in addition to the longer term needs. In addition, business decision makers, architects, implementors, DBAs, and data acquisition specialists can use this book to gain a better understanding of how a data warehouse can be designed, implemented, and used.

This book is dedicated only to the first step of implementing a BI solution and concentrates on the setup of the infrastructure. It does not intend to cover all the aspects of a BI solution. For example, specific considerations on data sharing, very large databases, data mining, performance, and capacity planning are intentionally excluded and will be the topics of future documents.

## The team that wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO), San Jose Center.

**Paolo Bruni** is a DB2 Information Management Project Leader at the ITSO, San Jose Center. He has authored several books about DB2 for z/OS and related tools, and has conducted workshops and seminars worldwide. During Paolo's many years with IBM, in development, and in the field, his work has been mostly related to database systems.

**Gaurav Bhagat** is a System z Consultant in India Software Lab Services. He has six years of experience in System z system programming, software deployment, and migration on z/OS. Over the years, Gaurav has worked as a system programmer and consultant on WebSphere, DB2, and OMEGAMON® for z/OS. He holds a Bachelor of Engineering degree in Electronics and Communication from Delhi University.

**Lothar Goeggelmann** is a Software Development Engineer for data warehouse on System z at the IBM Development Lab in Boeblingen, Germany. He has 12 years of experience with software development on System z and porting of software products to System z. Over the years, he has worked in various projects such as DB2 for z/OS, WebSphere Portal on z/OS, and WebSphere Portal on Linux on System z. He has worked at IBM for 24 years. Lothar has a degree in computer science from the Hochschule Ulm.

**Sreenivasa Janaki** is a Data Architect working for National City Corporation in the United States. He has more than 10 years of experience developing data warehouse and BI systems, many of which are implemented on DB2. His areas of interest include the extract, transform, and load (ETL) processes of data warehouses, database design, and database administration. He holds two Master of Science degrees in engineering from the University of Alaska at Fairbanks.

**Andrew Keenan** is a Managing Consultant in Australia with IBM Global Business Services. He has 13 years experience in BI, management reporting, and data migration projects. He has acquired skills in using multiple vendor products within the Analytics & Reporting, Information Management, and Single Customer View solution areas. Over the years, Andrew has worked on a number of projects in implementing Cognos BI solutions.

**Cristian Molaro** is an independent consultant based in Belgium working in Europe for customers with some of the largest DB2 installations. His main activity is linked to DB2 for z/OS administration and performance. He has presented papers at several international conferences and local user groups in Europe. He is an IBM Certified DBA and Application Developer for DB2 for z/OS V7, V8, and V9. He holds a Chemical Engineer degree and a Master in Management Sciences.

**Frank Neumann** is a Software Architect for Data Warehouse on System z in the IBM Lab in Boeblingen, Germany. During his career with IBM, Frank worked as a developer, team lead, and architect for software components and solutions. He holds a Master of Computer Science degree from the University of Karlsruhe, Germany.

Figure 1 shows a photo of the team.



Figure 1 Left to right: Sreeni, Paolo, Cristian, Andrew, Lothar, Frank, and Gaurav

Special thanks to the following people for their contributions to this book:

- ▶ Claudia Imhoff, Ph.D., President and Founder of Intelligence Solutions, Inc. for allowing us to use some of her definitions
- ▶ Larry Gosselin for allowing us to use charts from his presentation “IBM Information On Demand 2007 - Reference Architecture”
- ▶ Khadija Souissi, a System z Specialist at the Technical Marketing Competence Center Europe in the IBM Lab in Boeblingen, Germany, for her contributions throughout this project

Thanks to the following people for their contributions to this project:

Emma Jacobs  
Deanna Polm  
Sangam Racherla  
ITSO

Oliver Draese  
Ingo Hotz  
Helmut Schilling  
Juergen Schimpf  
Emil Wolf  
IBM Germany

Angelo Sironi  
IBM Italy

Nigel Campbell  
Colin Moden  
IBM Canada

Paul Christensen  
Sean Crowley  
Gary Crupi  
Tony Curcio  
Larry Gosselin  
Richard Harken  
IBM USA

Jaime Anaya  
Aarti Borkar  
Willie Favero  
Beth Hamel  
Terrie Jacopi  
Bob Lyle  
Roger Miller  
Terry Purcell  
Jonathan Sloan  
Guogen Zhang  
IBM Silicon Valley Lab

## Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

[ibm.com/redbooks](http://ibm.com/redbooks)

- Send your comments in an e-mail to:

[redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)

- Mail your comments to:

IBM Corporation, International Technical Support Organization  
Dept. HYTD Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400





# Part 1

## Data warehouse today

In this part, we provide a brief overview of data warehouse and business intelligence (BI) definitions. We highlight the main reasons for a return of interest on System z as a platform that provides the characteristics needed for a successful implementation of a BI solution. We also describe the infrastructure that may be used for such a solution.

The objective of this part is to present terminology, explain the reasons for using DB2 and System z for building a data warehouse, and review the methods and tools that are available to build a solid data warehouse solution.

This part contains the following chapters:

- ▶ Chapter 1, “Definitions” on page 3
- ▶ Chapter 2, “Evolution of business intelligence” on page 17
- ▶ Chapter 3, “Why implement a data warehouse on System z” on page 31
- ▶ Chapter 4, “The architecture for the BI solution on System z” on page 59

Archived

# Definitions

Where does a data warehouse begin and end? Is a data mart part of a data warehouse? When is a project for a company, which was started as a data warehouse, really a data mart? Where does operational reporting belong? What do you call a data warehouse that provides detailed operational information? How does business intelligence (BI) differentiate from a data warehouse?

In this chapter, we provide a broad overview of the concepts of data warehousing and some operating definitions. This chapter contains the following sections:

- ▶ 1.1, "Introduction" on page 4
- ▶ 1.2, "The data warehouse environment" on page 4
- ▶ 1.3, "Data warehouse data definitions" on page 10
- ▶ 1.4, "Data warehouse functional definitions" on page 14

## 1.1 Introduction

Data warehousing has evolved from the needs of corporate decision makers. The large amount of data gathered from the 1970s and 1980s by large corporations in their operational systems was somewhat harnessed and put together for making informed business decisions in data warehouses.

In the late 1980s and early 1990s, personal computers, fourth generation languages (4GLs), and online analytical processing (OLAP) tools became readily available. Users wanted more direct control over their data. Each department began requesting extra views of the enterprise data or different extracts for different purposes from the same source.

Today many companies are looking at cross-functional business processes to provide better customer-centric services. There is a need to understand and react faster to the immediate concerns of a customer by improving on activities such as customer service, campaign management, and real-time fraud detection. These services require the ability to access and update an integrated, consolidated, and current view of detailed operational data. Therefore, the enterprises are turning to “near” real-time transactional middleware as a way of moving data between applications and other source and target configurations.

## 1.2 The data warehouse environment

A *data warehouse* is an organization’s data with a corporate-wide scope for use in decision support and informational applications. Therefore, a data warehouse is designed to serve all possible decision support processes for an organization. Two types of data subsystems make up the “data warehouse environment”:

- ▶ The *enterprise data warehouse*, which contains all data with a global scope
- ▶ The *data marts*, which contain data for a specific business

Another subsystem, called the *BI application layer*, incorporates BI user applications and templates. In general, it provides all the services to select, extract, and manipulate the data.

The data warehouse layer and the BI application layer together become the data warehouse and BI or decision support system of the enterprise, which is illustrated in Figure 1-1 on page 5. This data warehouse and BI system provides executives and line managers with information to help support their decisions and to sustain and improve their business. These decisions can be about the future developments or operational in nature.

The ETL box is the data warehouse function that manages the extract, transform, and load (ETL) process for the data warehouse and the ETL elements that populate the data marts. This function generally entails the extraction, transformation, and loading of your data from the data source into the data warehouse by using a *staging area*.

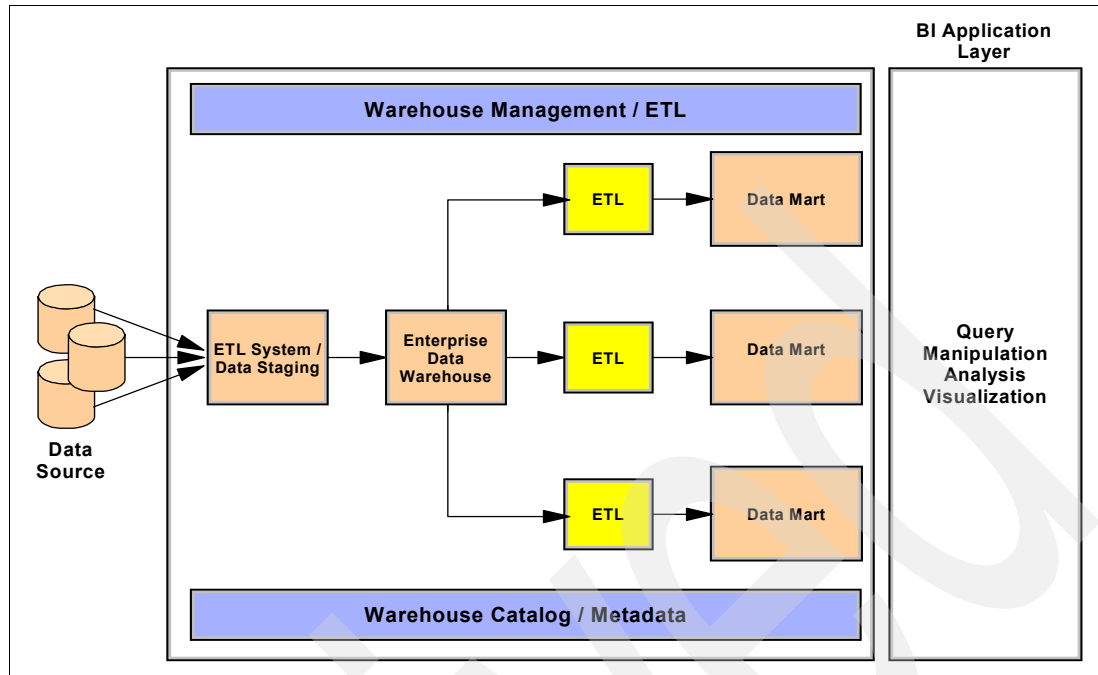


Figure 1-1 Data warehouse and BI environment

Let us see some other definitions.

According to Bill Inmon, the Corporate Information Factory contains components, such as an applications environment, from which the data flows into an integration and transformation layer (I & T layer) and then into an operational data store (ODS).<sup>1</sup> The data warehouse is fed by either the ODS or directly from the I & T layer, which consists of current and historical detailed data. This then can flow into data marts that are derived and dependent on the data warehouse primarily, except for external data and historical reference data. Therefore, Inmon's definition does not put the data marts within the data warehouse.

Traditionally, the Kimball Group has referred to the process of providing information to support business decision making as data warehousing.<sup>2</sup> Now they use the phrase *data warehouse/business intelligence* to mean the complete end-to-end system. They refer to the data that can be queried in a warehouse/BI system as the enterprise's data warehouse, and value-add analytics as BI applications.

Barry Devlin states, "A data warehouse is simply a single, complete, and consistent store of data obtained from a variety of sources and made available to users in a way that they can understand and use in a business context".<sup>3</sup>

As it is often the case, for a given problem, different approaches can yield the right solution. You may choose to build a data mart directly from the source system data. You can also use a *federated system* application or an MIS approach and query all the source systems directly for decision support processing.

In this chapter, we focus on the definitions of the subsystems that are the components of the software architecture, especially for the data warehouse environment (shown in Figure 1-1 on page 5). In Chapter 2, "Evolution of business intelligence" on page 17, we expand these

<sup>1</sup> <http://www.inmoncif.com/home/>

<sup>2</sup> <http://www.rkimball.com/>

<sup>3</sup> Barry Devlin. *Data Warehouse: From Architecture to Implementation*. Addison-Wesley Professional, 1996.

concepts to include more recent additions to data warehousing. We describe the solution that is pertinent to our scenario in 5.3, “Solution overview” on page 75.

We start by defining the *enterprise data warehouse*, the *departmental data warehouse* or *data mart*, the *operational data store*, and *business intelligence*. Then we mention other commonly used data warehouse definitions.

### 1.2.1 Data warehouse and data mart

You know you have built an enterprise data warehouse when it has the following characteristics:

- ▶ Is corporate-wide in scope
- ▶ Presents a subject-oriented data organization
- ▶ Contains a non-volatile store of data that is integrated, cleansed, and derived as needed from various source systems
- ▶ Is capable of holding historical data

The data mart, or business process dimensional model, usually has a more restrictive scope in the sense that it is not corporate wide. It is more restrictive in terms of size and usability. It can have primitive, summarized, and derived data that answers a particular business process or problem domain, or it helps a particular business group within an organization. A data mart supports looking at important measures by a variety of contextual dimensions.

You can ask: Is a data mart just a smaller data warehouse? The answer is “not quite,” for several reasons. We said earlier that the scope of the data mart is smaller. Also the data warehouse usually is created with many different users in mind and is designed to answer several questions that are not yet asked by the users.

For example, in a risk-oriented data mart for banking, you may have a data mart that has a metric called “Count of loans with balances having been past due for greater than 90 days”. It may span across dimensions of Branch, General Ledger Code, Time (as of the date that the 90-day threshold was crossed), Loan Type of Commercial versus Consumer, and so on. In the data warehouse, you save the loan number along with the exact due date so that you can calculate whether the loan is past due for greater than 30 days or 60 days and so forth.

Therefore, we store detailed granular data that is helpful in answering unanticipated questions within context. Also business rules differ from one department to another in classifying a loan as past due for a date period. In a data mart, you might apply the rule for one department or process and ignore the other departments. From a data warehouse, you apply the rule on its way to a derived data mart.

A data mart has been defined as “An implementation of a data warehouse with a small and more tightly restricted scope of data and data warehouse functions, serving a single department or part of an organization” by H. S. Gill and P. C. Rao in *The Official Client/Server Computing Guide to Data Warehousing*.<sup>4</sup> A data mart is created when the business users have precise questions that they require to be consistently answered over a period of time. A data mart commonly supports multidimensional analysis.

#### Warehouse catalog

The *warehouse catalog* is the subsystem that stores and manages all the *metadata*. The metadata refers to such information as data element mapping from source to target, data element meaning information for information systems and business users, the data models

<sup>4</sup> H. S. Gill and P. C. Rao. *The Official Client/Server Computing Guide to Data Warehousing*. Que 1996, p. 353.

(both logical and physical), a description of the use of the data, and temporal information. In a banking environment, for example, if you have a field, such as Chargeoff Amount, then it is worthwhile to the information systems and business users if this field is the net of Recovery Amount.

An example of the use of metadata is to inform users how they can distinguish a self-serve safe deposit box from a non-self serve safe deposit box or distinguish business or commercial loans from consumer loans. Temporal information is about maintaining knowledge of temporal intervals for business data. This can include information over time ranges in regard to loan rates, borrower credit ratings, and collateral ratings.

### **Warehouse management or ETL subsystem**

The warehouse or ETL management subsystem is the data warehouse function that manages the ETL process for the data warehouse and the ETL elements that populate the data marts. It can include job scheduler, backup systems, recovery and restart, version control, lineage, and other areas.

For example, in a banking scenario, someone might want to know whether the load from the General Ledger system finished successfully and the time at which it completed. In this case, you go to this subsystem to learn that information. You can even have a report that details this information. Information in this subsystem, such as the backup and recovery related information, can be useful to the application development team and the DBAs. Having such a system can help reduce the burden on key development team resources. It can also help in any transition efforts as new team members are brought into maintain the decision support system of the enterprise.

### **Operational data store**

In the data warehousing arena, there is another construct known as the *operational data store*. The ODS can also be used to feed the enterprise data warehouse and hence dependent data marts. An ODS is integrated and subject-oriented similar to the data warehouse. However, the update frequency, direct update paths from applications that make it volatile, and current valued information distinguish it from the data warehouse. For general characteristics and benefits of an ODS, refer to *Building the Operational Data Store on DB2 UDB Using IBM Data Replication, WebSphere MQ Family, and DB2 Warehouse Manager*, SG24-6513.

### **Business intelligence**

The term *business intelligence* is interspersed in this book. Therefore, it is worthwhile to define it. The book *From Multiplatform Operational Data to Data Warehousing and Business Intelligence*, SG24-5174, defines BI as a general term that covers all processes, techniques, and tools that support business decision making based on information technology. The approaches can range from a simple spreadsheet to a major competitive intelligence undertaking. Figure 1-2 on page 8 shows an example of a BI architecture.

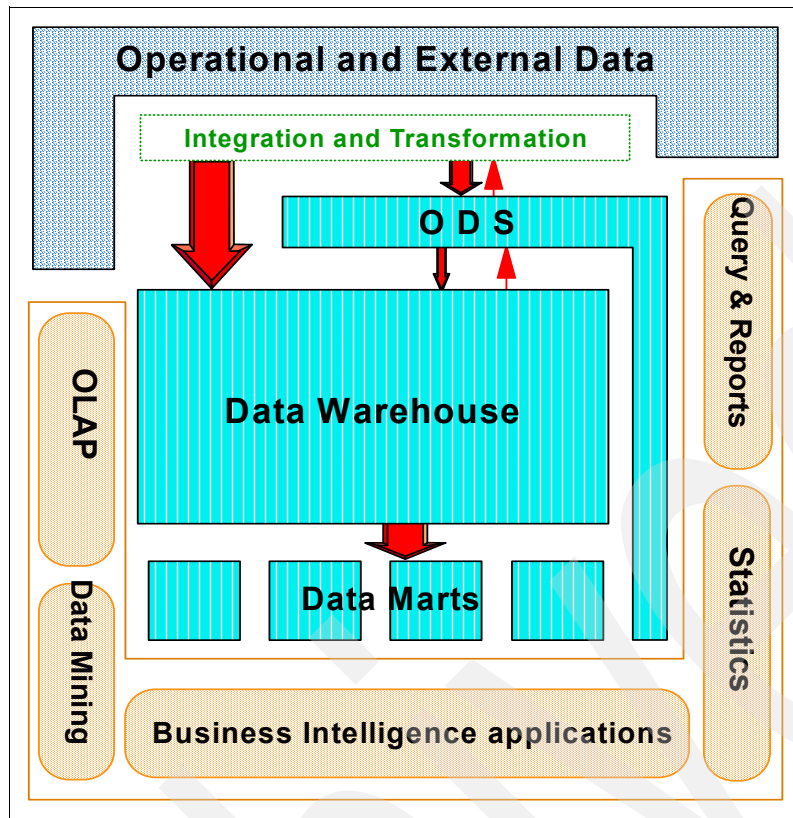


Figure 1-2 Example of a BI architecture

For more information, see “Business Intelligence and Performance Management” on the Web at the following address:

<http://www.ibm.com/BI>

In the following sections, we look at the major components of BI as illustrated in Figure 1-2.

### **Query and reporting**

A *query* is defined as a business question that is translated into a computer query. In a query and reporting process, the business question is defined, the data is accessed, the answer set is returned, and the report is presented in a chart or a graph to the user. The reports can be *canned*, in which they represent a stable workload, or they can be *ad hoc*, where they represent a variable workload.

### **Online analytical processing**

OLAP is made possible with a multidimensional view of data. OLAP usually involves a multidimensional data store from which tools, such as those provided by Cognos software, can display measures, use its calculation component along several dimensions, and present data to users. Therefore, a user should know the kind of queries in advance to use OLAP. In Relational OLAP (ROLAP) tools, the data is sourced from relational structures as opposed to the source being from a prebuilt OLAP cube.

### **Data mining**

Data mining is the exploration and analysis of large quantities of data in order to discover unknown or hidden, comprehensible and actionable information and using it to make business decisions and to support their implementation, including formulating tactical and



strategic initiatives and measuring their success. You use techniques such as decision trees, scoring, association analysis, and neural networks.

## Operational business intelligence

One of the prime prerequisites for competitive advantage in the marketplace is information. Business intelligence is the domain that is responsible for gathering that information and making it available to decision makers. For improved decision making and to enable a competitive advantage, the need for more current information continues to grow. Most companies are expending effort to satisfy this need.

There are two primary and different functional areas of any company where BI is required:

- Business strategy and planning

This area is the traditional BI. It is concerned with more strategic issues such as which products to make, where to sell them, how and where to distribute them, whether to use mergers and acquisitions, and how to maximize profit. To do this, a company needs what is termed *strategic, or informational, business intelligence*.

- Business operations

This area focuses on the tactical day-to-day operations of the company. It is typically focused heavily on developing and executing efficient business processes. The goal is to produce and deliver products quickly and at the lowest cost possible, and to maintain client satisfaction. To do this, a company needs what is termed *tactical, or operational business intelligence*.

At the IBM Information On Demand 2007 Conference in Las Vegas Nevada, Claudia Imhoff and Mike Schroeck defined operational BI as: “a set of services, applications and technologies for monitoring, reporting on, analyzing, and managing the business performance of an organization’s daily business operations.”<sup>5</sup>

Such help in decision making can be useful for front-line managers. For this to happen, operational data must be available with little latency to the front-line operational managers with history. An example of this information in a bank is for front-line managers to be better prepared in cross selling other products to a customer. It can also be to take the lifetime value of a customer and provide free overdraft protection up to a certain dollar value.

The best feature of operational BI enablement is that more users in an organization can use the data. This means that the IT managers do not have to constantly justify the value of the data warehouse environment to the organization. These front-line managers can help extol the value of this technology product as well. This product helps the front-line manager in understanding a customer better when they need it most, in real time, and provides service to the satisfaction of the customer.

It is difficult to argue against a technology that keeps a customer happy, which is the proven way to keep an organization stable and grow it. In large banks, for example, it is mandatory in the United States to look for suspicious money laundering activities by looking at transaction-level data. This rich repository of transaction data can be potentially used for operational BI enablement.

In general, the data warehouse environment can be broken down into data, functional, and technical aspects. The definitions that are commonly used are described in 1.3, “Data warehouse data definitions” on page 10. The definitions portray an understanding of the need

<sup>5</sup> Provided by Claudia Imhoff, PH.D., President and Founder of Intelligence Solutions, Inc., and Mike Schroeck, Partner, IBM Global Business Services, “Operational Intelligence: Business Process Management Meets Business Intelligence” presentation at the Information On Demand conference in Las Vegas 14-19 October 2007

for the perfect alignment of the three aspects to satisfy a business need for an enterprise-wide decision support system.

## 1.3 Data warehouse data definitions

To look at the data aspect, you need a data model. A good example is the widely used entity relationship model, which is perhaps the most traditional approach, with subject areas and data granularity and metadata. Predefined data models can be purchased to use as a starting point.

For example, for Banking, IBM offers the Financial Markets Industry Models, which provide an overall analysis framework for handling risk management, regulatory compliance, and so on. The IBM Financial Markets Process Models are designed for financial markets organizations that are looking to re-engineer, broaden, and standardize their core enterprise-wide business processes. The IBM Financial Markets Services Models are for organizations that are implementing new business process architectures. They are also for companies that are focused on their service-oriented architecture (SOA) strategy. For more details about IBM Industry Models, see 5.7, “Data modeling options: IBM Industry Models” on page 88.

Within the data models for Financial Markets is the Financial Markets data warehouse. This data warehouse is a data management toolkit that is designed to assist financial institutions in building warehousing solutions, from both an analytical and operational perspective.

As any data warehouse data model should, the Financial Markets data warehouse provides the blueprint for a single consistent enterprise view of the data. Such a prepackaged industry model can accelerate your goal for your own data warehouse. If you have already built a data warehouse, then chances are that you may be bringing in more source systems with a wider variety of data. For example, you may have some financial loan information but not lease information in the data warehouse. You may not have marketing campaign analysis information. You can use such models for accelerating the next releases of your data warehouse.

The data model in a warehouse context can be challenging because of the involvement of the time component that makes it a temporal-entity relationship model. Let us look at the different data modeling styles that are available.

### 1.3.1 Data warehouse data modeling styles

When time is introduced, relationships between data can change. For example, if you want only the latest information for a person, then you can enter their name, marital status, and so on in that entity. However, if you want a history of that person, then you have to enter a time component and call it *person history*. It might have one or more occurrences of name, marital status, and so on over time. Also, an added time component must be defined, which can be the time at which it is captured in the online transaction system or when it is known to the data warehouse at load time.

There are trade-offs because it is unrealistic to capture time stamps for each attribute used in the data warehouse. Input from the business requirements can help to reduce the span of the temporal model. The model does not go back to update or delete any inserted data. It inserts more rows that inform us of the end of the use of the older information. Therefore, in this case, it behaves in insert-only mode.

## Cumulative snapshot

When the cumulative snapshot data modeling style is employed, the data is a series of snapshots of the source data. The frequency of the snapshots is usually dependent on the business requirements. It is common to see a data warehouse that consists of monthly snapshots of information. For example, a state government might be interested in the types of assistance, such as food stamps, Medicaid, or cash assistance, that are available to a recipient at the end of each month and then compare the metrics month over month for several years. Another example is a bank that is interested in metrics such as the outstanding loan amount at the end of every month.

## Cumulative snapshot with rolling summarization

If volume is an issue because of the frequency of taking snapshots, such as daily, then older snapshot information can be consolidated to a lesser level of detail such as monthly.

## Continuous temporal model

In the continuous temporal model type of data modeling approach, all the data changes that are important to the business are captured without losing the changes. Contrast this with the cumulative snapshot approach, in a situation where a monthly snapshot was chosen. In the cumulative snapshot approach, all the changes that happened within the month are unknown to the data warehouse. Then we only know the information at the end of the month.

In the continuous temporal model, we take a snapshot of the data regardless of whether it is changed. In the continuous temporal model, we know all the changes that happened within the month.

The ODS data model used in this book is loosely based on the TPC-H benchmark model. We have the latest information in the ODS and keep the history in the dimensional data store (DDS) with the date dimension that provides the time slicing of the metrics in the fact table.

The data warehouse usually contains fact, event, or state information. *Events* or *facts* happen at a specific time, while a *state* represents a situation that is stable over a period of time. Events are associated with the time when they occur. The time can be represented by a time stamp or date based on business needs. States are associated with timestamp (or date) intervals. For example, consider the interval information, such as “01 October 2007 through 31 October 2007”, in your bank statement. This interval record can be tied to instant records that designate when the event took place such as each ATM transaction in your account.

Let us now see how the design for query and extract performance leads to summary tables and the reduction of join complexity that leads us into *multidimensional modeling*.

## 1.3.2 Multidimensional data model

To describe multidimensional data modeling, we must first define the elements that are presented in the following sections. The model can be construed as the resolution of a many-to-many relationship across all of the contextual dimensions for which the study of the measure is of interest.

### Measure

A measure usually indicates quantities, amounts, durations, and so on. A measure is a data item that information analysts can use in their queries to measure the performance of their business, such as the number of policies that are sold by an insurance company.

## Dimension

*Dimensions* provide the contextual background for the measures, for example, the number of auto (line of business) insurance policies sold in Ohio in March 2008. In this example, the line of business dimension distinguishes auto insurance from homeowners insurance. The mention of the state of Ohio shows the geography dimension, and March 2008 relates to the time dimension.

## Fact

A *fact* is a collection of related measures that are taken together with their associated dimensions and represented by the dimension keys. A fact should be of interest to an information analyst. For example, a general ledger reconciliation fact is useful in knowing if the measures used in reporting and analysis match the general ledger.

## Grain

The *grain* of a dimension is the lowest level of detail available within that dimension.

## Granularity

The *granularity* of a measure is determined by the combination of the grains of all of its dimensions.

## Dimension hierarchy

Dimensions can consist of one or more alternate hierarchies. An example shown later in this book is a hierarchy that corresponds to branches, nations, and regions. Each dimension hierarchy can include several aggregation levels.

## Multidimensional data modeling: Snowflake model

The *snowflake model* is the representation of a multidimensional model in which the dimension hierarchies are structured and normalized. For example, a time hierarchy is structured separately as year, month, week, and day.

## Multidimensional data modeling: Star model

The *star model* is the representation of a multidimensional model in which the dimensions are collapsed in dimension tables. For example, you might have a date dimension in which the week number, month, and year are all contained. Therefore, for a year, you can generally have 365 days. If you have measures that you must look at for 10 years, then you have approximately 3650 rows with one date for each day in this date dimension table. Each day has additional details indicating whether it is a holiday and which day of the week it is.

## Slowly changing dimensions

Most dimensions change slowly. For example, if you have a customer dimension that contains the name, gender, marital status, and so on, then it will change slowly. When the dimension changes, the Kimball Group considers the following design choices and implementation techniques:<sup>6</sup>

- Slowly changing dimension (SCD) Type 1

You use this type when you can choose to overwrite the existing information in the dimension with the new information. For example, if the marital status changes, then you overwrite the marital status, losing the older status. There are no key changes and so forth to worry about.

<sup>6</sup> "Kimball Design Tip #15: Combining SCD Techniques" by Margy Ross  
<http://www.kimballuniversity.com/html/designtipsPDF/DesignTips2000%20/KimballDT15CombiningSCD.pdf>

- **SCD Type 2**

You use this type when you intend to keep the older customer record for use with data until the change day and create the newer information in a new customer record for use in tying it to the fact table from that day onward. For this reason, you need surrogate or artificial keys that can be generated for these two same customer records in the customer dimension table.

**Note:** In this book, you can see the application of the SCD Type 2 technique in the customer dimension as implemented by the InfoSphere DataStage Slowly Changing Dimension stage.

- **SCD Type 3**

In this type, we keep the original attribute information and the most current information, while losing anything that happens in between. For example, if the marital status changes, then we have the first marital status that we knew of in the Original Marital Status field and the most recent status in the Current Marital Status field. We never overwrite the Original Marital Status field with any future changes. We only overwrite the Current Marital Status value.

## **Data analysis processing**

Consider the following examples of general types of data analysis processing for a bank:

- **Trend Analysis:** How many new accounts have opened every month for the past 12 months?
- **Statistical Analysis:** How much is the average fee per commercial loan per month by standard industry classification code?
- **Ranking:** What are the top 20 customers in terms of performing loans and most loan amount outstanding with the highest collateral rating and excellent credit rating?

## **Multidimensional data analysis techniques**

Multidimensional data analysis techniques are possible, especially with tools such as those provided by Cognos software, which can help in analyzing OLAP data sources across many dimensions. They are slice-and-dice, drill-down, roll-up, and pivoting analysis. You can see these methods in action in Chapter 13, “Reporting and analysis with Cognos 8 BI” on page 305. They are defined as follows:

- **Slice-and-dice analysis**

Slice-and-dice analysis occurs across several dimensions such as in insurance underwriting company, channel, line of business, geography, and time to look for typical business behavior and exceptional events.

- **Drill-down analysis**

Drill-down analysis is an iterative analysis where you explore the metrics at more detailed levels of the dimension hierarchies. As an example, you can look at the metrics by country, state, ZIP 3 code (in U.S. the first three digits of the ZIP Code) and ZIP 5 code. Roll-up is the reverse way of this going up the dimensional hierarchy.

- **Roll-up analysis**

Roll-up analysis enables you to explore the metrics at a higher level of summarization.

- **Pivoting analysis**

Pivoting analysis is when the result set from an OLAP source is rotated to another to be given priority in the demonstration of the result.

## 1.4 Data warehouse functional definitions

The functional aspect contains the data transport and transformation area. The subsystems to populate the enterprise data warehouse and the subsystem to move the data into the data marts are part of this aspect. Each such subsystem is called an *ETL task* as shown in Figure 1-1 on page 5. Therefore, the subsystem also populates the ODS and moves the data from the ODS to the dimensional model, which is the scenario presented in this book.

The data that is populated in the data warehouse environment typically comes from the source online transaction systems and possibly external sources. It can include foreign exchange rates, a valid list of ZIP Codes, and so forth. You can use a variety of replication techniques along with data transformations, data cleansing, and information derivation.

### 1.4.1 Replication techniques

Typically if you are starting to build your data warehouse project or are bringing in a new source system information into the data warehouse, you do an initial or full load of the history of the source system or system and then incrementally update the data with a frequency needed to satisfy the business requirements. Replication management in this context refers to techniques in the replication of data between different data management sites.

In Chapter 10, “Full load using DataStage” on page 257, and Chapter 11, “Incremental update with DataStage” on page 281, you see the full and incremental loads that were accomplished for the tables that were used for the scenario in this book. Such tools as DataStage are used to transform and apply the data from the source systems to the target database.

For the incremental load, a source capture technique can read from the recovery log of the online transaction system by the Q Capture tool as shown in Figure 1-3. Applying it to the target system with Event Publisher and DataStage is an efficient way of enabling source system updates to be piped into the data warehouse environment. For data warehouses to be useful for the information consumers, they must be nimble. With a quicker implementation of the ETL in a data warehouse with the use of tools, such as DataStage, this becomes more plausible. Therefore, source capture techniques, target apply techniques, and tools play an important role in data warehousing.

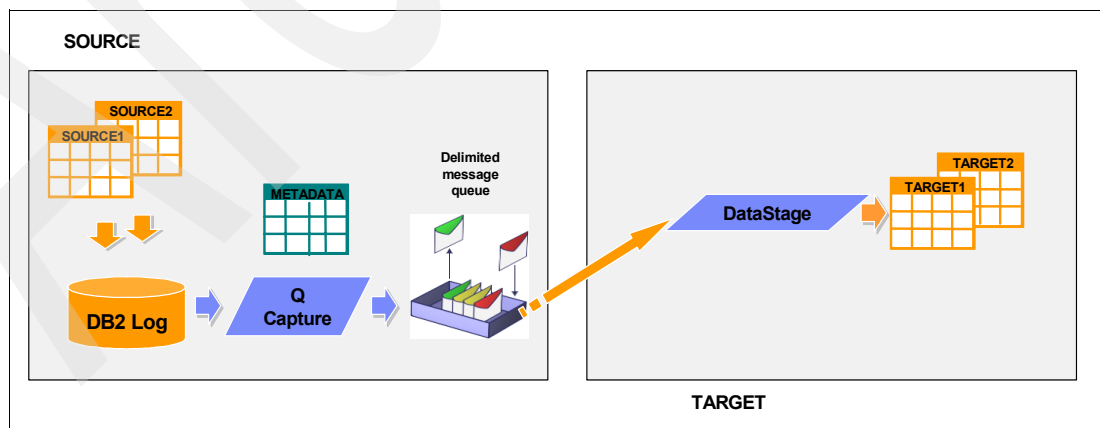


Figure 1-3 Sample incremental process

## 1.4.2 Data transformations

You can have either record-level, attribute-level, or both levels of transformation.

### Record-level transformation

Record-level transformation happens, for example, when you split one record into multiple records or push multiple records into one record. You see in Chapter 10, “Full load using DataStage” on page 257, that we used the DataStage tool and pushed the OLTP shipment table information into the orders and line-item tables in the ODS.

### Attribute-level transformation

Attribute-level transformation involves change at a single attribute-level and can be as simple as a data lookup. Again in Chapter 10, “Full load using DataStage” on page 257, you see how we used the country name that came in the OLTP customer table and supplanted it with a nation key in the ODS customer table using the ODS nation\_lookup table. You can have other attribute-level transformations such as aggregation and enrichment.

### Data cleansing

Data cleansing is a complicated process. It consists of statistical and other data manipulation and transformation techniques, and is used to eliminate variations and inconsistencies in data content. In general, data cleansing is used to improve the quality, consistency, and usability of the base data. As an example, you can have address data that may not have proper street addresses, ZIP Codes, and so forth. It is difficult if the marketing managers in an organization are doing campaign analysis by ZIP Code and the ZIP Codes in many records of the transaction system are empty. Therefore, you can use address cleansing tools that can use U.S. Postal Service information and make sure that the address information and ZIP Codes are cleansed.

### Business metadata

Business metadata is the enterprise knowledge of how the data is used for business purposes. Business people can refer to the business metadata when trying to understand and follow business policies and procedures.

### Technical metadata

Technical metadata can be static or dynamic.

*Static technical metadata* can be source-to-target mappings for all the source systems that are being extracted into the data warehouse environment. Examples of static technical metadata are the mapping of source system file inventory information to the target system database's subject areas, transformation and derivation rules, and so on.

*Dynamic metadata* includes information from job executions. Load schedules, load statistics, lookup rejects and, in general, load rejects, and so on are part of this grouping of metadata.

### 1.4.3 Application techniques

In application of the full or initial load, we used “load replace,” and in the incremental load, we used “load append”. This can be controlled by target table properties for the DB2z stage in DataStage jobs. (A DataStage process step is described in 10.3, “Load jobs for populating ODS from OLTP” on page 262).

You can also have a constructive merge in which case you update the end time for any record that is going to be superseded by new information. You can see this in Chapter 11, “Incremental update with DataStage” on page 281, where we explain the implementation of the slowly changing dimension stage.



# Evolution of business intelligence

In this chapter, we discuss the fast changing evolution of the business intelligence (BI) arena and the methods and technology that support it. As corporate demands for using vast information explode, technology and functionality have evolved, and the line of separation between data warehousing and online transaction processing (OLTP) systems has become dotted. Evolution within BI is discussed in the areas of technology and functionality maturity, users, and requirements.

We also discuss some of the market drivers and challenges that organizations are facing. We explain how recent trends, such as *Information On Demand* and *operational business intelligence*, provide innovative ways to address these challenges.

This chapter contains the following sections:

- ▶ 2.1, “Market drivers and challenges” on page 18
- ▶ 2.2, “BI technology and functionality evolution” on page 19
- ▶ 2.3, “Types of users and their requirements” on page 22
- ▶ 2.4, “Information On Demand” on page 24

## 2.1 Market drivers and challenges

Organizations are continuing to rate data warehouse and BI initiatives high within their strategic plans and continue to increase their spending. One of the key areas that is driving this is the need to invest in performance management and link corporate strategy and initiatives with metrics, queries, and reporting.

Corporate performance management implementations, along with the right data governance controls, allow an organization to take advantage of the following actions:

- ▶ Have realistic goals based on real information
- ▶ Adapt goals intelligently when required and focus on the outcome
- ▶ Align individuals' goals to the strategic goals of the organization
- ▶ Provide communication and accountability
- ▶ Measure progress against key performance indicators and publish results in a central location where it is available for everyone to provide feedback
- ▶ Gain an understanding of the business and what drives it

Market growth is also being driven by the push to have BI available to “the masses”. That is to give a greater number of users within the organization, from executives down to the operational staff, the ability to use corporate information or embedded analytics within applications to accelerate decision making in their daily tasks. Figure 2-1 on page 22 gives an indication of the user population and type of users that have been introduced to BI capability over time.

New capabilities continue to be added to BI implementations. An example of this is the addition of real-time updates of operational data within an operational data store (ODS) or similar, further allowing users to use current information.

Growth within the BI industry is also linked to the challenges that organizations are facing, such as the following examples:

- ▶ Increase employee productivity
- ▶ Improve business processes
- ▶ Better understand and meet customer expectations
- ▶ Manage risk and compliance
- ▶ Improve operational efficiency
- ▶ Manage ongoing cost pressures
- ▶ Promote the use of existing information in the decision making process

In addition to the business challenges, technical challenges also arise:

- ▶ Too much information and not knowing which is most important  
The amount of information is growing, and new types are being introduced.
- ▶ Lack of data integration  
Information is scattered throughout the organization, sometimes in separate silos.
- ▶ Lack of appropriate technical skills available
- ▶ Real-time access to information
- ▶ Query performance optimization
- ▶ The need to integrate structured and unstructured data sources
- ▶ Lack of trust in data sources

- ▶ The need for rapid deployment of new systems
- ▶ Scalability of systems
- ▶ The need to provide extended search capabilities across all enterprise data
- ▶ The need to create master data sets of information important to the business, for example, customers, products, and suppliers
- ▶ Lack of agility in regard to inflexible IT systems
 

Employees are spending too much time on managing change in these systems rather than doing more strategic tasks.
- ▶ Lack of self-service BI query, reports, and analysis
- ▶ Time too long for operational data updates to flow from an ODS to scorecards and dashboards in a traditional BI implementation

## 2.2 BI technology and functionality evolution

The message or concepts of BI have not changed over time. By applying processes and methodologies against corporate data and turning it into useful information organizations can gain insight and support their decision making process. The maturity of the technology and functionality that can be used to help apply the necessary processes and methodologies has changed, as has the output that BI implementations can deliver to organizations. In 2.2.2, “BI evolution and maturity” on page 21, we look at how BI has evolved.

Traditionally, data warehouses extract data from the operational storage through batch processes executed within maintenance windows. Today BI systems incorporate large data warehouses that are consolidated with near real-time operational data and support message-based integration of data. Real-time data feeds mean that a database is being updated in real time while it is being queried. This is similar to the transactional database. The new generation of data warehouse can be called *dynamic warehousing*.

Traditional and new breeds of BI applications access the dynamic warehouse, which contains both historical and operational data. They provide information to the user at the time they require while performing their tasks. The time available to provide this information has become shorter, and many organizations need to monitor their operational data and react to certain events as soon as it is triggered. In addition the number of users requiring information from the data warehouse environment has increased. This is demonstrated in Figure 2-1 on page 22.

Traditionally BI and data warehousing were for the executive officers of an organization. This is no longer true with the technology and functionality available now ensuring that BI can be applied at all levels within an organization. This has created pressure on conflicting workloads, and solutions now need to ensure the right balance is found between performance and latency.

Data warehouses must now address the following workloads:

- ▶ Increasing number of reports being developed throughout an organization
- ▶ Increasing number of users wanting to perform ad hoc queries
- ▶ Near real-time data loads
- ▶ New functionality being added to operational applications that provide BI capability against OLTP information

The core characteristics of mainframes are well positioned for processing the heavy workload of BI and performing the role of the key database server. For many organizations that hold their corporate information in VSAM, IMS™, or DB2 for z/OS data stores, deciding to put their data warehouse on System z has benefits.

Enterprises that have an existing DB2 for OLTP in place and already use System z as their data serving platform are well positioned to consolidate business-relevant data marts on the System z platform. DB2 for z/OS exploits the System z platform strengths that have been there from day one. In addition, DB2 for z/OS continues to be enhanced, specifically for data warehousing and BI functionality. Some of this is discussed in Chapter 3, “Why implement a data warehouse on System z” on page 31, and Chapter 7, “Functions in DB2 for z/OS for a data warehouse” on page 99.

Many enterprises are also turning to message-oriented, near real-time transactional middleware as a way of moving data between applications and other source and target configurations. The IBM MQSeries® family of products is such an example of messaging software that enables enterprises to integrate their business processes.

## 2.2.1 Current BI trends

New innovative functionality and technology is being introduced into the BI arena. We have discussed real-time data feeds as one example. Table 2-1 shows other examples of some of the current trends in BI.

*Table 2-1 Examples of current BI trends*

Trend	Description
Enterprise search	New search tools that allow access to information everywhere providing enterprise search capability.
BI Web services	Web services that allow collaborative and analytic services for incremental application development in a BI environment
Embedded analytics	Pre-built service-oriented components that can be integrated into operational applications and Web pages to become part of existing business processes. They combine current operational information with historical information from the data warehouse.
Event driven alerts	Allows monitoring of key business events and uses push technology to notify of a change in these events.
Analysis of unstructured information Text mining	Organizations have discovered that their unstructured data can contain useful information. Unstructured formats can include voice-mail, images, text, e-mails, and documents. New requirements include integrating structured and unstructured data.
Virtual BI	Virtual data marts <sup>a</sup> and operational data stores that provide real-time information across both the current operational data and the historic information held in the data warehouse.
Dynamic warehousing and real-time information	Traditional warehousing involved query and reporting of what happened in the past. Dynamic warehousing is an approach to implementing the Information On Demand story in regard to real-time information and leveraging information.
Data warehouse industry models	Provide for a faster implementation of data warehouse and BI solutions. IBM provide solutions in areas such as banking, insurance, telecommunications, retail and health.

- a. *Virtual data marts* are logical views of data warehouse data. The data mart is not physically implemented, aggregation and summarization logic is contained within the views, and business users access their data mart directly through these views.

## 2.2.2 BI evolution and maturity

In many organizations, the maturity path towards developing an enterprise BI solution may have included a number of stages over time with various outcomes occurring along the way.

Early data warehouse efforts focussed on querying and reporting of financial and sales information. The next wave introduced technologies, such as online analytical processing (OLAP) and data mining. Additional examples are listed in Table 2-2. These implementations were useful for analysis done in the past. As the maturity of BI has increased over time, the value that is returned to an organization, and the efficiency and effectiveness of the solutions, have also increased. Implementations have become useful for analyzing the present and performing predictive analysis.

Table 2-2 BI evolution examples

Example	Description
Static reports and spreadsheets	The number of reports throughout an organization increased dramatically. Sometimes the same reports were developed by different people. They may have involved manual effort and had no central controlled definition of business rules or metadata, potentially resulting in multiple versions of the truth.
Parameter reports	Parameter reports can provide multiple reports in one, reducing the number of reports required to be developed. Traditionally further analysis was limited to selected analysts.
Implementation of OLAP	The concept of providing slice-and-dice functionality and summary information became popular. OLAP cubes allowed multidimensional analysis of data within a predetermined cube data format. Cubes were created as required and were sometimes uncontrolled causing a lack of integration. Data quality within the source data may not have been addressed sufficiently when building cubes, and data quality issues caused problems with maintaining such cubes, such as different versions of hierarchies.
Traditional warehousing and data integration	Data integration exercises became important. Traditional data warehouses and data marts were ways to ensure corporate information could be integrated into a single reporting environment. The data warehouse environment proved useful for historical analysis but was unable to provide operational reporting requirements due to the time it took for data to travel from the OLTP source systems to the warehouse environment. <b>Note:</b> The term <i>traditional</i> has been used to distinguish from the term <i>dynamic</i> warehousing.
Ad hoc queries	Self service technology allowed executives to perform some of their own ad hoc queries against data within data marts or operational repositories.
Corporate performance management (CPM)	Performance management and scorecard initiatives become popular. These allowed the monitoring and management of an organization's performance by linking key indicators to the organization's strategies and goals.
Data mining and analytics	Complex data mining initiatives required specialists to build and train appropriate models.

## 2.3 Types of users and their requirements

The types of users who require a BI solution have evolved from traditionally being executives to everyone throughout the organization. The requirements of these users also differ depending upon their role and the maturity of BI within the organization.

Figure 2-1 shows that, as the population base for BI users has increased, the type of users has evolved from the executives down to those people at the front line of the business. Even now, it has evolved to include those customers to whom an organization is providing services. Traditionally the executive type users were making strategic decisions and looking at the organization as a whole. Query and reporting were primarily for the analyst level and above. Front line employees are now using BI in their day-to-day decision making and impacting at the transaction data level, for example the customer order of parts.

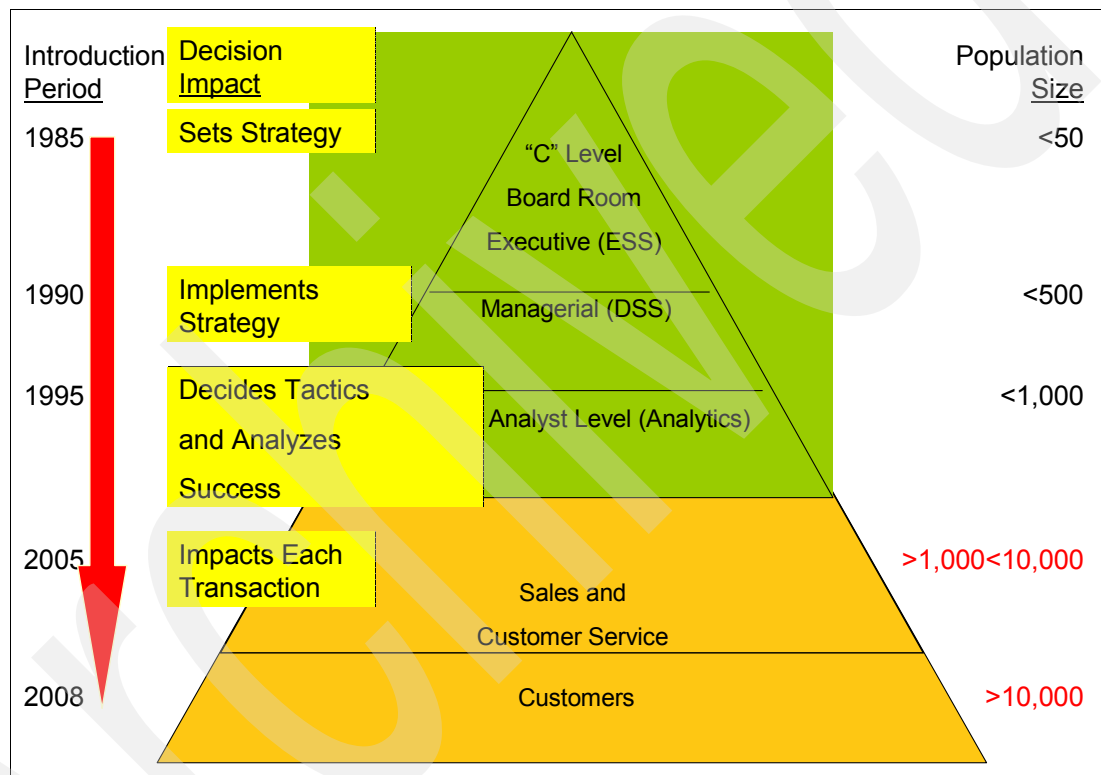


Figure 2-1 Operational intelligence introduction and user population

To implement a successful BI solution, it helps to understand the types of reports that exist and the different styles of reporting that can be used.

Reporting can for the most part be grouped into the following two categories:

- ▶ Production reports
- ▶ Business reports

*Production reports* can also be classified as *operational reports* that usually come from an ODS or operational system, such as a general ledger system. They are mostly used by the front line, customer facing staff of an organization. These reports are required for the day-to-day operation of the organization and can include financial statements, invoices, and customer orders. The trend is for these reports to be provided in more near real time.

Production reports are used for operational reporting. Operational reporting relates to a specific task or day to day business process and is a reaction to a specific event, such as taking a customer order.

*Business reports* are more targeted for the executives of middle management of an organization. These users understand the performance of the organization and are more responsible for implementing initiatives for strategic goals. Business reports combine information from more than one operational system and are usually sourced from an integrated environment, such as a data warehouse implementation.

Business reports are used in strategic and tactical reporting. Tactical reporting relates to a specific business process but is forward looking and does not just say what happened. Strategic reporting is also forward looking but addresses multiple business processes and looks at an organization as a whole, from a strategic and performance level.

BI users can be classified as follows:

- ▶ Ad hoc users
- ▶ Power users
- ▶ Publishers
- ▶ Viewers

Figure 2-2 shows how the different types of users require different technologies or capabilities to provide the interaction they require.

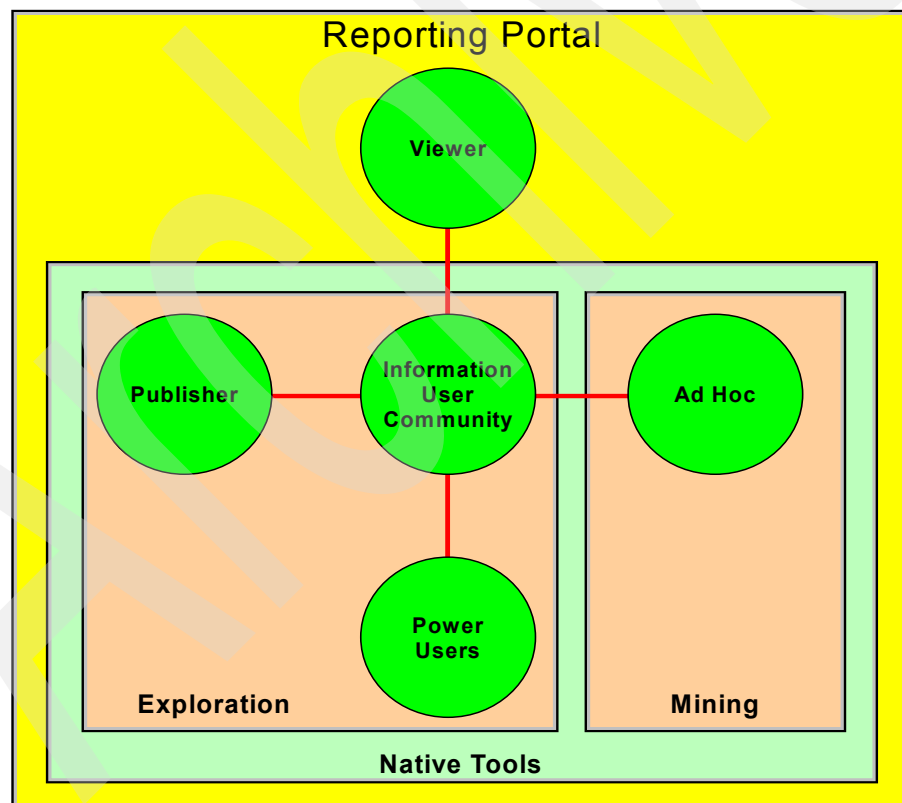


Figure 2-2 BI tools - User associations

*Ad hoc* users require traditional BI capability to perform their ad hoc analysis and extra mining functionality if performing specialist mining tasks. Many modern implementations of querying tools provide this functionality through a Web browser or BI portal.

*Power users* may perform complex analysis tasks against relational or multidimensional data. They require strong analytic and exploration capabilities.

*Publishers* build and deploy BI models, allowing others to perform their query and reporting tasks. They are also capable of performing analysis and exploration tasks because they need to understand the business information and requirements in order to deploy successful business-oriented solutions. Publishers require native BI tools to define models that meet business requirements.

*Viewers* can use a Web browser or BI portal to view published business and production reports.

## 2.4 Information On Demand

Information management has traditionally referred to centralized data repositories with accurate information that can support high performance and high availability. Although this requirement remains true, for information consumers, the contents of these repositories have become harder to access. Some of the reasons why this has occurred are mentioned in 2.1, “Market drivers and challenges” on page 18.

Innovative applications that allow Information On Demand and Information as a Service capabilities can be applied against selective business processes. In doing so, we can move beyond the traditional approaches of integrating and accessing information. We can allow information to become a strategic asset that can be used to make better decisions, by using the following methods:

- ▶ Integrating, analyzing, and standardizing the content and types of data sources available into master data sets based on business needs and not the underlying applications or repositories that contain this information

This process is known as *master data management*. These master data sets are identified as the information that an organization uses repeatedly across business processes such as customers, locations, and products. By following this approach, an organization can proactively address data quality sooner rather than later such as in the traditional data warehouse environment. The data warehouse can now use these master sets of information.

- ▶ Applying new technologies and architectures such as Web services and service-oriented architecture (SOA) along with a master data management strategy as mentioned previously

SOA without a data strategy may not be successful due to the possible high degree of data inconsistency and inaccuracy.

- ▶ Empowering users to make good use of this information by providing new capabilities to use, analyze, and maintain this information within their usual business processes
- ▶ Ensuring that information integrity and data governance methodologies are in place for people, process, and technology

These methodologies assist in providing compliance and providing trust, accountability, and stewardship around the data.

Organizing master data about critical information domains, such as customers or products, leads directly to better business decisions and business operations. Analyzing historical and real-time information helps unearth insightful relationships, again leading to better business decisions and business operations.



To successfully provide Information On Demand capabilities, a common reference architecture or roadmap that supports the business strategy and requirements is required. This roadmap should describe the major components of an end-to-end solution and provide a framework to identify the scope, assess the risk, and investigate gaps with an organization's existing capabilities.

The Information on Demand logical architecture diagram in Figure 2-3 offers an introduction to the common components that the Information On Demand architectures can share.

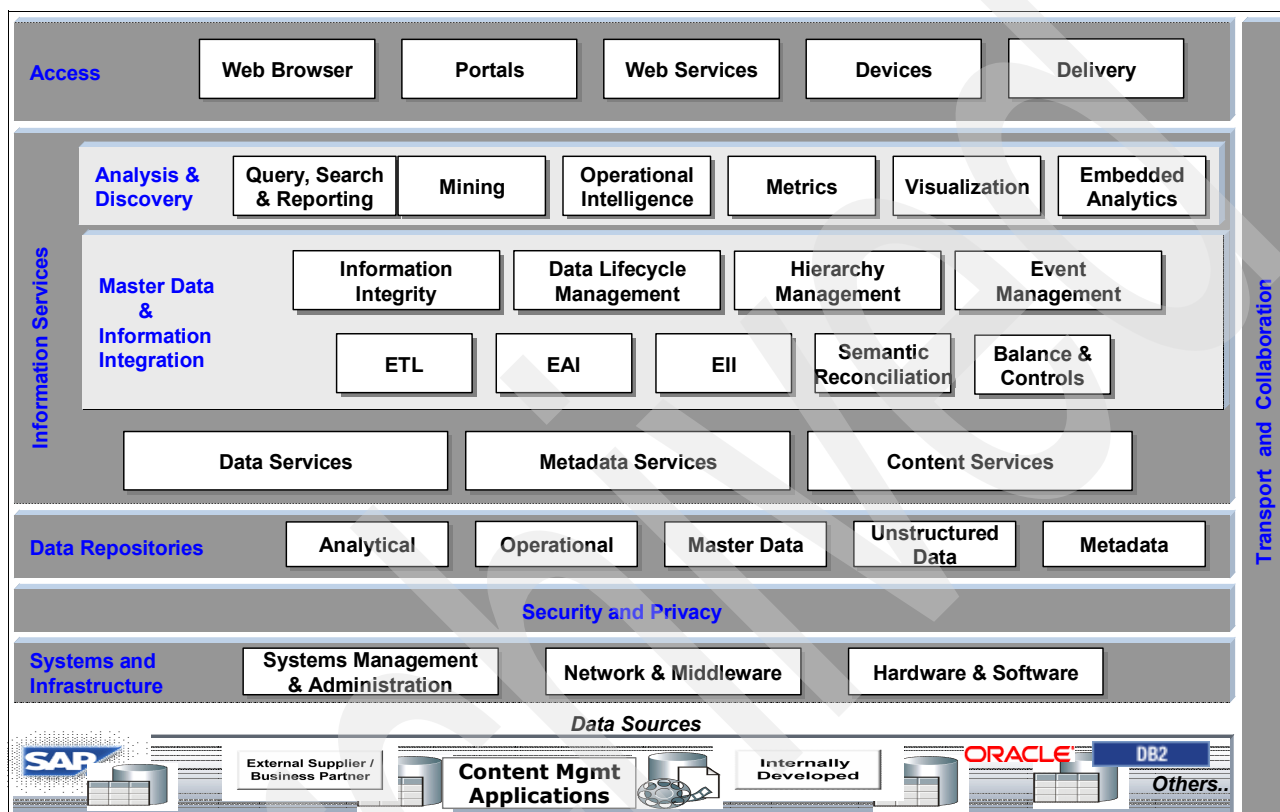


Figure 2-3 Information On Demand logical architecture

In addition, a set of guiding architectural principles that support the Information On Demand implementation are needed. Table 2-3 provides examples of these principles.<sup>1</sup>

Table 2-3 Information On Demand architectural principles

Principle	Description
Information as a service	In an SOA, information services enable business processes to gain access to information that they need in a timely fashion and in conformance with open industry standards to make decisions, recognize exceptions, and understand events.
Virtualized data	New composite applications that need data from homogeneous or heterogeneous sources in distributed or mainframe environments. The data may be structured or unstructured and available to applications via integrated or federated access.

<sup>1</sup> Presentation "IBM Information On Demand 2007 - Reference Architecture" by Larry Gosselin

Principle	Description
Unconstrained access to data (structured and unstructured)	Access and analysis of unstructured information is done to discover, organize, and deliver relevant knowledge via Unstructured Information Management Architecture framework (UIMA). Combined with structured data analysis, it provides enhanced analytics and encourages data reuse or sharing.
Single version of the truth	This principle entails consolidation of one master enterprise copy that describes core business entities: customers, suppliers, partners, products, materials, bill of materials, chart of account, location, and employees.
Metadata management	This principle refers to the management of data about data (metadata), which defines the meaning of data in the repository. It ensures consistency, completeness, and context of data.
Deep analytics	Identity resolution, relationship resolution, and anonymous resolution offer analytics capabilities to relate two entities without the need for a common attribute key.
Advanced search	Uncovering inherent meaning, user intent and application context allows for the delivery of meaningful information that is easily actionable.
Optimized data movement and placement	Data synchronization and data replication across systems provide for right-time availability. The placement of data facilitates information integration.
Data quality	Quality data is the product of an organization or program that builds information integrity into its technology, processes, and people.

### 2.4.1 Operational business intelligence

Operational BI refers to the methodologies, processes, and applications that are used to report on and analyze operational business performance.

Traditional BI mostly entailed executives analyzing historical data and performing strategic analysis on general questions, such as how an organization has performed against certain criteria over a given time period. Operational intelligence or operational BI has moved BI into all levels of an organization. It allows executives and everyone within the organization to incorporate proactive analysis into their operational processes and applications. This analysis is more specific and common than traditional BI analysis. It covers questions and decisions that are part of the organization's day-to-day business processes. An example includes: "What discount can I offer my customers based on their past ordering history?"

**Operational intelligence scenario:** In this scenario, BI is used to inform the management of the RedParts Distribution company about the number of products that are ordered or returned by customers on a near real-time basis. This information is used to determine product stock levels and return rates. This can then be combined with current promotions to determine which discounts to apply to customer orders in the next reporting period. When a customer calls to organize the return of unwanted products, this information can be displayed within an analytic application via business event management and analytic software such as Cognos 8 BI. This software shows the ordering and return history of the customer. Depending on the customer's history, a decision can be made to offer different incentives to the customer.

Traditional BI implementations usually use portals within Web browsers to display results to information consumers. Although this is still valid, for operational BI, newer methods, such as e-mail, corporate applications, metric display panels, and wireless devices, can be used.

Operational BI is about optimizing an organization's performance in the present, not the past. It provides the organization with a near real-time view by using dashboards, metrics, and reports.

Traditional BI and operational BI are different but complement each other. The traditional data warehouse can become a source of information that is used to provide historic data to operational BI applications.

An operational BI implementation should include the following components:

- ▶ **Historical data**  
This includes self-service dashboards with links to traditional analysis and reports.
- ▶ **Real-time data**  
Information should also include near real-time data that is also provided to the dashboards and reports.
- ▶ **Business activity monitoring for complex event and data processing from both operational data sources and the data warehouse environment**  
Key events are distributed to those required by using push technology, for example, e-mail notification.
- ▶ **Information On Demand capabilities**  
Refer to 2.4, "Information On Demand" on page 24, for an explanation of these capabilities.

Operational BI helps to link high level organization objectives with the day-to-day tasks that are performed by operational employees. This is achieved by following a performance management framework that can track how an organization is currently performing and not just how they did perform.

*Performance management* is a term that describes the methodologies, metrics, processes, and systems that are used to align organization resources with business objectives and goals. Performance management is used to monitor and manage the overall performance of an organization. The performance management cycle (Figure 2-4) shows an ongoing cycle by which organizations can set goals, analyze their progress using operational BI applications, gain insight, take action, and measure their success. This process is ongoing and is supported by the operational BI implementation.

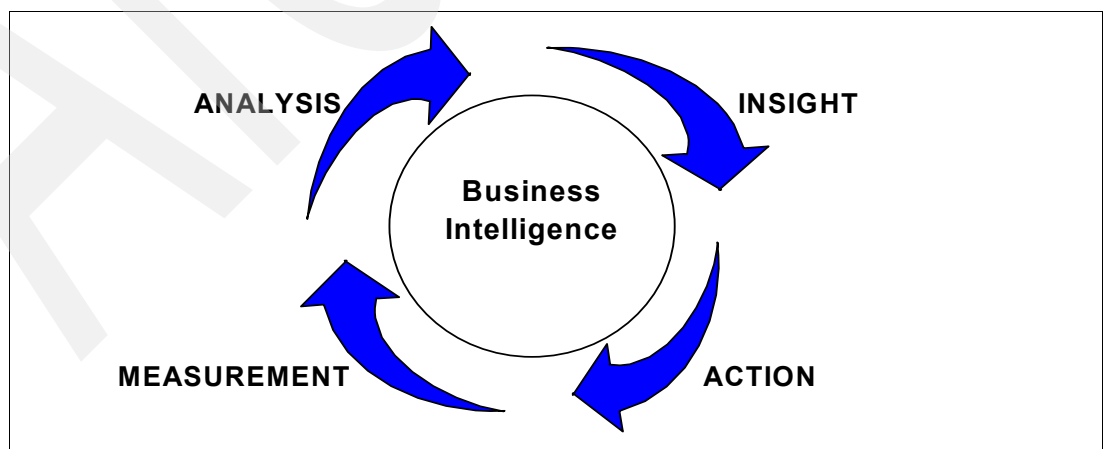


Figure 2-4 Performance management cycle

Data from many operational sources can be analyzed by using embedded analytics that define, measure, and track details regarding an organization's strategies and goals. Having operational BI systems that support this analysis can help in determining what is really important. It also helps those who make decisions to understand the information that is required and to alter their decisions at the point of interaction.

Analysis should lead to insight. These insights can be displayed with the help of scorecards and reports in near real-time by using applications such as Cognos 8 BI and IBM AlphaBlox®. Through an organization's decision making process, these insights can suggest actions that will improve the organization.

Insights and actions can also be measured to ensure that they are working. This can further lead to more insights and data becoming available for further analysis. The cycle can then be repeated, allowing an organization to refine its strategies.

Embedded analytics in operational BI provide the following features and characteristics:

- ▶ Solution-based operational BI delivery platform
- ▶ Customized analytic components
- ▶ Embedded into existing processes and Web applications
- ▶ Pre-built components ("objects")
- ▶ Integrates multidimensional and relational views
- ▶ Combination of operational information (that is, current point in time) with data warehouse (historic) information
- ▶ Service-oriented
- ▶ Real-time in nature

**Operational intelligence best practices and lessons learned:** Consider the following sample best practices and lessons learned when starting an operational intelligence initiative:

- ▶ Establish clear and continued executive sponsorship.
- ▶ Define and develop a shared "point of view" organizational perspective on operational intelligence.
- ▶ Start with a project that is "strategically" important.
- ▶ Focus on data governance and stewardship across the organization.
- ▶ Provide trusted data sources and solution flexibility.
- ▶ Leverage common process models and reference architecture.
- ▶ Apply integrated methods, and a common taxonomy and language.
- ▶ Deliver phased, value-driven implementations.
- ▶ Deploy a scalable, flexible, and technical infrastructure.
- ▶ Capture and share best practices and lessons learned.

Data warehouses have been implemented on DB2 for System z servers largely because of the traditional strengths that the System z platform provides, such as unparalleled performance, continuous availability, and unlimited scalability. Figure 2-5 shows example components for an operational BI solution by using DB2 for z/OS.

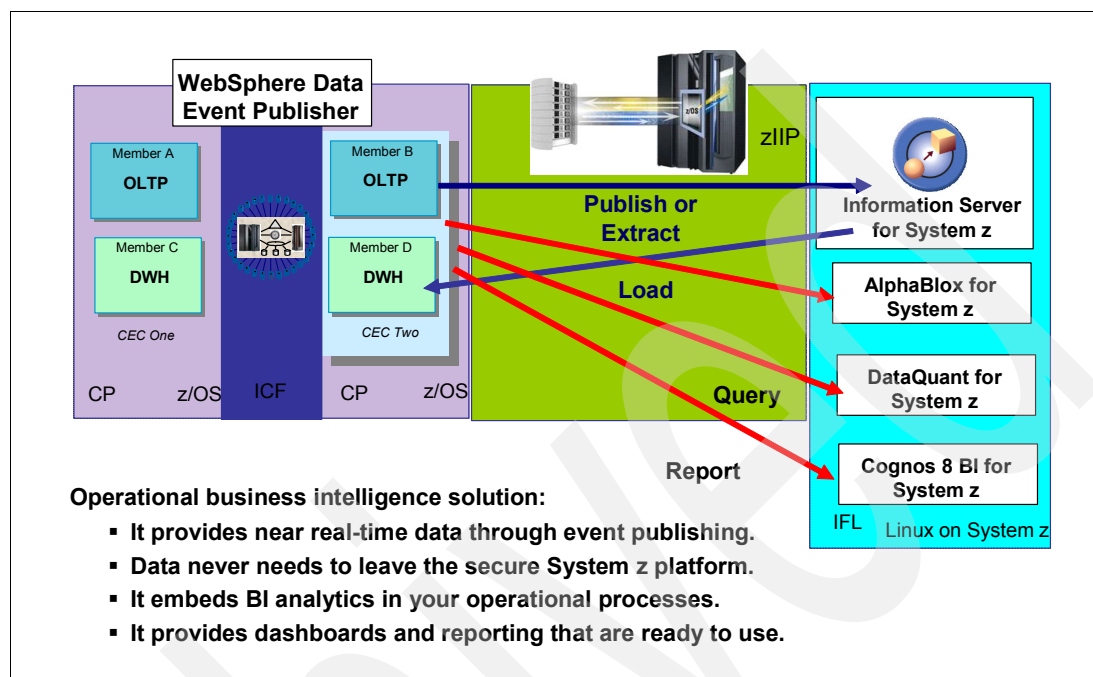


Figure 2-5 Example components for a DB2 for z/OS operational BI solution

## 2.4.2 Applying Information On Demand with dynamic warehousing

Dynamic warehousing is a new approach that can be used to implement Information On Demand initiatives. It addresses the primary business challenges that organizations face today. These challenges require the ability to deliver the right information to the right people at the right time to more effectively leverage information and enable more effective business decisions. It is about information on demand to optimize real-time processes.

Where traditional warehousing involved query and reporting of what happened in the past, dynamic warehousing is an approach to implement the Information On Demand story in regard to real-time information and leverage information.

Dynamic warehousing requires the following criteria:

- ▶ Support for real-time access to aggregated, cleansed information that can be delivered in the context of the activities and processes being performed
- ▶ Analytics that can be leveraged as part of a business process
- ▶ The ability to incorporate knowledge from unstructured information
- ▶ A complete set of integrated capabilities that extend beyond the warehouse to enable Information on Demand

The distinction between data warehousing and OLTP is blurring. Data warehousing and analytic applications access operational or near-real-time data. Transactions have become more complex to provide better interaction and productivity for people. Dynamic warehousing has capabilities and strengths on all IBM platforms. The traditional mainframe strengths for consistency with operational data, high security, and continuous availability match well with dynamic warehousing.

**Dynamic warehousing scenario:** In this scenario, we illustrate how to transform sale effectiveness. By using and understanding the relevant unstructured customer information to which the RedParts Distribution company has access, they can use this to identify other product cross-sell opportunities. They can also improve their negotiating position while engaged with customers at the point at which a customer is making an order over the Internet or in person at a branch.

## Why implement a data warehouse on System z

In this chapter, we explain why and when to implement a data warehouse on System z is the right choice. Customers who are already running their transactional system on DB2 for z/OS have multiple options if they want to build their warehouse environment.

Originally, DB2 for z/OS was designed to be a business intelligence (BI) platform, but then in response to customer demand that focused more on online transactional processing (OLTP). In the past couple of years, however, customers have encouraged IBM to expand business intelligence functions on DB2 for z/OS to use System z advantages for their data warehouse.

Additionally, new requirements for modern applications, such as using historical data from a data warehouse for tactical decision making in day-to-day transactions, make the System z platform an interesting and competitive hosting platform for data warehouses. Refer to *IBM System z Strengths and Values*, SG24-7333, for details.

This chapter contains the following sections:

- ▶ 3.1, “New challenges for data warehouse solutions”
- ▶ 3.2, “Data warehousing with System z” on page 32

## 3.1 New challenges for data warehouse solutions

As discussed in Chapter 2, “Evolution of business intelligence” on page 17, enterprises are increasingly harnessing the power of business intelligence to gain a competitive edge. Today’s BI systems incorporate large data warehouses that are consolidated with near-real-time operational data stores (ODS) and are continuously updated from multiple sources, such as financial, marketing, and inventory databases. Often, thousands of users access a data warehouse with various business intelligence applications. These applications analyze and synthesize data into real-time information that supports fast and informed decisions.

These applications must be capable of leveraging all types of information, including unstructured content such as call center logs, technical notes, contracts, or call logs. Data warehouses must serve the expanding needs of different types of applications. As a result, data warehouses must also support the varying service level demands of these different applications.

A combination of critical operational applications and analytics insights that require real-time responsiveness is essential. Also the current traditional back-office reporting and analysis for strategic and tactical planning purposes are still needed. Together these elements will lead to increasingly mixed workload environments. This is further complicated by rising data volumes, continuously expanding amounts of historical data, and the growing number of users, which causes requests for information to become more numerous and sophisticated. Current business intelligence solution proposals must consider the challenges that are arising from the mix of the following four workload types:

- ▶ Continuous, near-real-time data loading, similar to an OLTP workload
- ▶ A large number of standard reports
- ▶ An increasing number of true ad hoc query users
- ▶ An increasing level of embedded analytics and business intelligence-oriented functionality in OLTP applications

These four workload types are increasingly top challenges for organizations beyond the growing size of the data warehouse. Mixed workload performance is well on its way to becoming the single most important differentiator issue in data warehousing.

## 3.2 Data warehousing with System z

The long standing success record of workload management capabilities makes System z an ideal platform to run a data warehouse workload. In this section, we outline the main reasons that make System z an ideal platform to run data warehouse workloads and handle emerging BI and dynamic warehousing requirements.

### 3.2.1 Availability and scalability

Access to the data warehouse in a BI solution is moving from the back office to the front office, making the BI solution a business-critical application that requires the same kind of availability and security as OLTP systems. Frequently the data warehouse is large and requires careful management to ensure that business information is available when needed and that access to that data is responsive but safe and protected. The System z platform and DB2 9 for z/OS provide an excellent infrastructure for these requirements making System z the platform of choice for critical processing.



The System z hardware, z/OS operating system, and DB2 for z/OS are designed with reliability characteristics such as self monitoring, redundancy, self healing, and dynamic configuration and management. For example, in DB2, you can make database changes, such as adding a partition, without an outage.

As businesses grow, their data processing needs also grow. Business ventures, such as mergers, acquisitions, new services, or new government regulations, can accelerate how quickly the data processing needs of the business grow. As rapid growth occurs, companies need a way to scale their business successfully.

Parallel Sysplex® clustering technology and DB2 data sharing are the answer to availability and scalability. A *sysplex* is a group of z/OS systems that communicate and cooperate with one another by using specialized hardware and software. The systems are connected and synchronized through a Sysplex Timer® or System z Server Time Protocol (STP), and Enterprise Systems Connection (ESCON®) or Fiber Channel Connection (FICON®). A *Parallel Sysplex* is a sysplex that uses one or more coupling facilities (CFs), which provide high-speed caching, list processing, and lock processing for any applications on the sysplex. For information about Parallel Sysplex technology and benefits, refer to the Business resiliency page at the following address:

<http://www.ibm.com/systems/z/resiliency/parsys.html>

Figure 3-1 illustrates a Parallel Sysplex. A Parallel Sysplex can include central processor complexes (CPCs) of different generations (for example, an IBM eServer™ zSeries® z890 or z990 and a System z9™ BC, or z9 EC, or a System z10™).

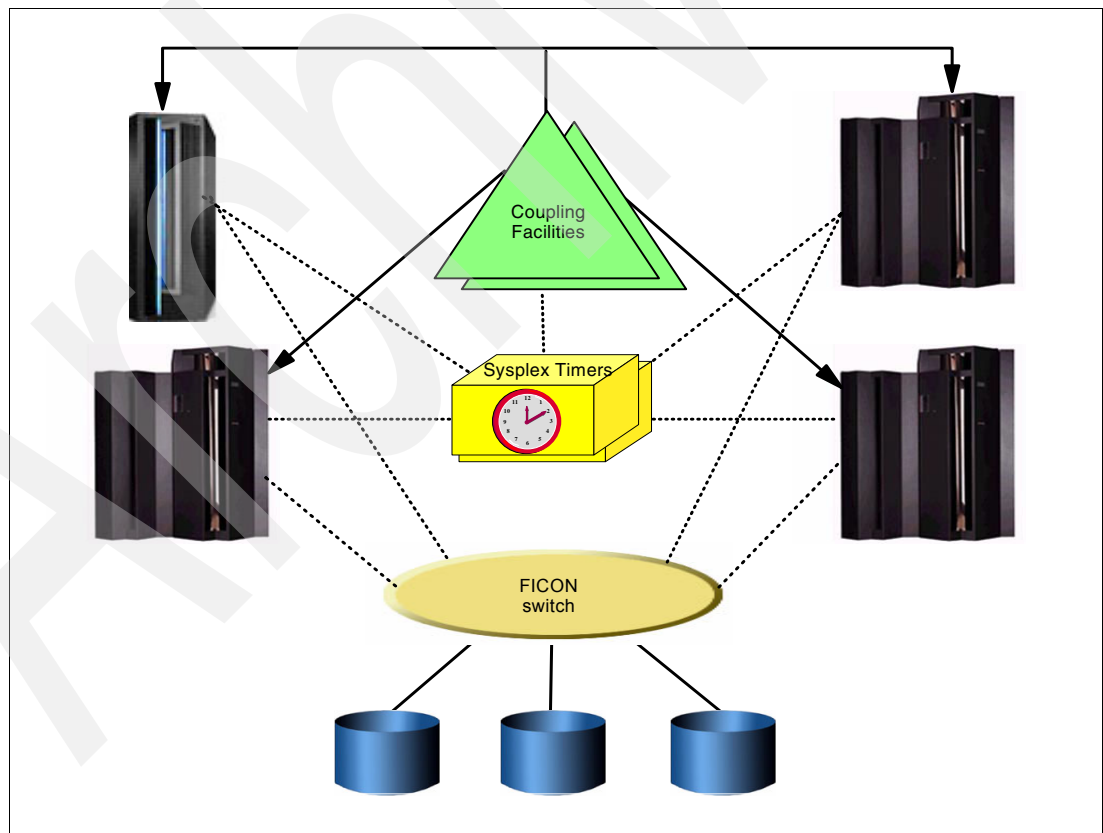


Figure 3-1 Parallel Sysplex architecture

The Parallel Sysplex can grow incrementally without sacrificing performance. The Parallel Sysplex architecture is designed to integrate up to 32 systems in one cluster. In a shared-disk cluster, each system is a member of the cluster and has access to shared data.

A collection of one or more DB2 subsystems that share DB2 data is called a *data sharing group*. DB2 subsystems that access shared DB2 data must belong to a data sharing group. A DB2 subsystem that belongs to a data sharing group is a member of that group. Each member can belong to one, and only one, data sharing group.

All members of a data sharing group share the same DB2 catalog and directory, and all members must reside in the same Parallel Sysplex. Figure 3-2 illustrates the DB2 data sharing architecture. Currently, the maximum number of members in a data sharing group is 32. The DB2 data sharing design gives businesses the ability to add new DB2 subsystems into a data sharing group, or cluster, as the need arises and without disruption. It provides the ability to do rolling upgrades of service or versions of the software stack without any application outage. For information about DB2 data sharing, refer to *DB2 for z/OS: Data Sharing in a Nutshell*, SG24-7322.

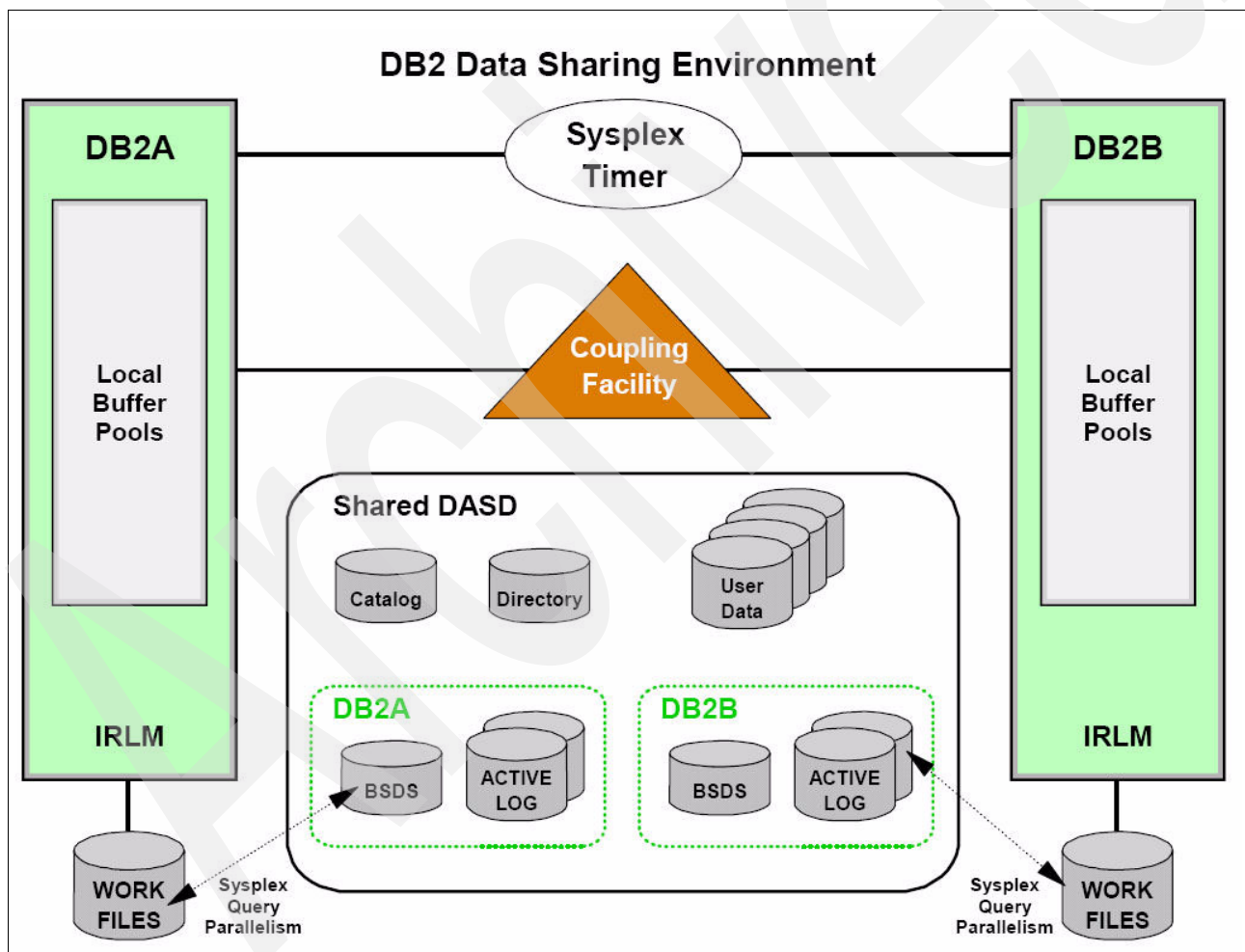


Figure 3-2 DB2 data sharing architecture

Automatic restart manager (ARM) is a component of z/OS that enables fast recovery of subsystems that might hold critical resources at the time of failure. If other instances of the subsystem in the Parallel Sysplex need any of these critical resources, fast recovery makes these resources available more quickly. Even though automation packages are used today to

restart the subsystem to resolve such deadlocks, ARM can be activated closer to the time of failure.

The implementation of Parallel Sysplex and DB2 data sharing caters to growing OLTP or BI needs, again with high availability and scalability.

### 3.2.2 Workload management

The Workload Manager (WLM) component of z/OS has proven its worth in many studies, demonstrations, and every-day work being done in systems around the world in its ability to maximize use of available resources. WLM has the ability to manage widely varying workloads efficiently and effectively. In doing so, you can fully use the systems resources that you have available. This means that you can run your data warehouse workload together with the transactional (OLTP) workload on the same DB2 subsystem or different DB2 subsystems on the same system.

With WLM, a particular element of work can be given an initial priority based on business needs. More importantly, over time, the priority of a given element of work can be altered based on changes in business needs as expressed in the WLM policy. DB2 for z/OS works hand-in-hand with WLM to ensure that these priority alterations take affect immediately with respect to query processing, regardless of how that query has been parallelized. Capacity can be put to use whenever it becomes available. For more information about data sharing and WLM on z/OS, refer to the following publications:

- ▶ *Workload Management for DB2 Data Warehouse*, REDP-3927
- ▶ *System Programmer's Guide to: Workload Manager*, SG24-6472

Table 3-1 illustrates the impact of workload management in a mixed workload environment. The first row, as a baseline, shows the times of short running and long running queries that are run sequentially. Without impact to each other, the short running query takes 1.5 seconds, while the long running takes 147 seconds.

In the second line, both query types are run in parallel. Because both queries now compete against each other and try to get resources from the system, the query response time suffers. However, it is important to note that the time for short running queries increases significantly by the concurrently run long running queries. In the example, query times go up from 1.5 seconds to 6.7 seconds. This might become an issue if the short running queries are run as part of a transactional application, where users experience a response time to their requests that is more than four times slower.

The third line illustrates changes in response time with WLM on z/OS. The concept of *period aging* automatically lowers the priority of long running queries and thereby keeps short running queries running at a higher priority. Consequently, long running queries do not impact short running queries too much. We see just a minor performance decrease for these.

Table 3-1 *Impact of workload management to a mixed workload environment*

	Short running queries	Long running queries
Sequential processing	1.5	147
Parallel processing with limited workload management	6.7	159
Parallel with z/OS WLM	1.9	158

### 3.2.3 Hardware data compression

DB2 has provided a hardware assisted option to compress table spaces so that they occupy less disk space since Version 3. Compression and decompression of data are accomplished on the System z environment via a hardware instruction. This instruction makes it faster than the software-based compression algorithms that are used by other databases systems on other platforms. As each generation of the System z processor gets faster, the compression feature also get faster.

Counter-intuitively, compressing data can reduce elapsed time of most data warehouse-type queries. DB2 for z/OS compresses the rows on a page, in order that each data page is full of compressed rows. It uses the hardware instruction along with a data dictionary to give the most efficient compression available. The compressed data can also be encrypted, thereby saving space and implementing security requirements at the same time.

With a rule of thumb of a 50% compression rate, a compressed page contains twice the rows that an uncompressed page contains. This means that each I/O retrieves twice as much compressed data as it retrieves if the data is uncompressed. The data remains compressed in the buffer pool. This means that DB2 for z/OS can cache twice as much compressed data in its buffer pool as it retrieves if the data is uncompressed. Finally, when data is modified in a row that is compressed, the information logged about that data change is also compressed, thus reducing log volume.

Not all data on a compressed page is decompressed, just the row or rows that are needed by the application are decompressed. Combined with the use of a hardware instruction to perform the decompression, this decompression serves to limit the amount of additional CPU that is needed to access compressed data.

The larger amount of data retrieved in each I/O is compounded with the DB2 9 for z/OS increased prefetch quantities. This provides significant elapsed time reductions for all types of sequential processes, including the typical business intelligence queries that use table scans and index range scans. This includes sequential processes for utility access, providing benefits in terms of faster reorganizations, faster unloads, and faster recovery.

### 3.2.4 Regulatory compliance

Regulations, such as the Sarbanes-Oxley Act (SOX), Basel II, Data Protection Act (UK), and the U.S. Patriot Act, were created to protect investors' interests, to avoid fraud, and to improve financial reporting. Companies must comply with these regulations. Database administrators must ensure that data is secure, access is controlled, changes are audited, and disaster recovery is in place. Regulations also emphasize the growing need to reproduce versions of data, applications, and entire business states, which challenges companies to keep a long record of activities.

The System z platform meets the highest industry security certifications. Encryption support is built in, even at the hardware level. Authorization functionality is an integral part of the operating system. Detection services prevent intrusions and record intrusion attempts. Network communication encryption follows the highest standards.

Consolidating warehouse data and operational data on one platform, such as the System z platform, eases efforts to comply with regulation requirements. Audit management efforts are reduced because of fewer data sources.

### 3.2.5 Disaster recovery

With the emergence of BI and dynamic warehousing, the disaster recovery requirements for a data warehouse environment are similar to that of OLTP. Therefore, it is important to consider disaster recovery scenarios before implementing a data warehouse solution.

The following System z and DB2 for z/OS technologies help in providing some of the best disaster recover solutions in the industry:

- ▶ Copy Services
- ▶ GDPS/XRC
- ▶ GDPS/PPRC
- ▶ GDPS/PPRC HyperSwap™
- ▶ BACKUP and RESTORE SYSTEM utilities of DB2 for z/OS

#### Copy Services

Copy Services is a collection of functions that provide disaster recovery, data migration, and data duplication functions. With the Copy Services functions, for example, you can create backup data with little or no disruption to your application. You can also back up your application data to the remote site for the disaster recovery. Customers who use the System z platform, along with IBM System Storage™ DS8000™ storage can have following optional licensed copy services:

- ▶ IBM System Storage FlashCopy®, which is a point-in-time copy function
- ▶ Remote mirror and copy functions, which include the following tools:
  - IBM System Storage Metro Mirror, previously known as synchronous PPRC
  - IBM System Storage Global Copy, previously known as PPRC Extended Distance
  - IBM System Storage Global Mirror, previously known as asynchronous PPRC
  - IBM System Storage Metro/Global Mirror, a three-site solution to meet the most rigorous business resiliency needs
- ▶ z/OS Global Mirror, previously known as Extended Remote Copy (XRC)
- ▶ z/OS Metro/Global Mirror, which is a three-site solution that combines z/OS Global Mirror and Metro Mirror

Many design characteristics of the DS8000 and its data copy and mirror capabilities and features contribute to the protection of your data, 24 hours a day and seven days a week.

#### FlashCopy

FlashCopy provides a point-in-time (PIT) copy of logical volumes with almost instant availability for the application of both the source and target volumes. Only minimal interruption is required for the FlashCopy relationship to be established to allow the copy operation to be initiated. The copy is then created “under the covers” by the IBM System Storage Enterprise Storage Server® (ESS), with minimal impact on other ESS activities.

Currently two FlashCopy versions are available: Version1 and Version 2. FlashCopy Version 2 supports all the previous FlashCopy Version 1 functions plus several enhancements.

#### Metro Mirror

Metro Mirror, which was previously known as *Synchronous Peer-to-Peer Remote Copy (PPRC)*, provides real-time mirroring of logical volumes between two DS8000 servers that can be located up to 300 km from each other. It is a synchronous copy solution where write operations are completed on both copies (local and remote site) before they are considered complete.

### **Global Copy**

Global Copy, which was previously known as *Peer-to-Peer Remote Copy Extended Distance (PPRC-XD)*, copies data non-synchronously and over longer distances than is possible with Metro Mirror. When operating in Global Copy mode, the source volume sends a periodic, incremental copy of updated tracks to the target volume, instead of sending a constant stream of updates. This causes less impact to application writes for source volumes and less demand for bandwidth resources, while allowing a more flexible use of the available bandwidth.

### **Global Mirror**

Global Mirror, which was previously known as *Asynchronous PPRC*, is a two-site long distance asynchronous remote copy technology. This solution integrates the Global Copy and FlashCopy technologies. With Global Mirror, the data that the host writes to the storage unit at the local site is asynchronously mirrored to the storage unit at the remote site. This way, a consistent copy of the data is automatically maintained on the storage unit at the remote site.

### **Metro/Global Mirror**

Metro/Global Mirror is a three-site, multipurpose, replication solution for both System z and open systems data. Local site (site A) to intermediate site (site B) provides high availability replication using Metro Mirror. Intermediate site (site B) to remote site (site C) supports long distance disaster recovery replication with Global Mirror. Both Metro Mirror and Global Mirror are well established replication solutions. Metro/Global Mirror combines Metro Mirror and Global Mirror to incorporate the best features of the two solutions.

### **z/OS Global Mirror**

z/OS Global Mirror, which was previously known as *Extended Remote Copy (XRC)*, is a copy function that is available for the z/OS and OS/390® operating systems. It involves a System Data Mover (SDM) that is found only in OS/390 and z/OS. z/OS Global Mirror maintains a consistent copy of the data asynchronously at a remote location that can be implemented over unlimited distances. It is a combined hardware and software solution that offers data integrity and data availability. It can be used as part of business continuance solutions for workload movement and data migration. z/OS Global Mirror function is an optional licensed function.

### **z/OS Metro/Global Mirror**

The z/OS Metro/Global Mirror mirroring capability implements z/OS Global Mirror to mirror primary site data to a location that is a long distance away. It also uses Metro Mirror to mirror primary site data to a location within the metropolitan area. This enables a z/OS three-site high availability and disaster recovery solution for even greater protection from unplanned outages of the DS8000.

### **GDPS/XRC**

GDPS/XRC is an industry leading e-business availability, multisite solution that is provided through IBM Global Services. It provides the capability to manage the remote copy configuration and storage subsystems, automate Parallel Sysplex operational tasks, and perform failure recovery from a single point of control, thereby helping to improve application availability.

Geographically Dispersed Parallel Sysplex™ (GDPS®) is an integrated solution offering that manages all aspects of switching computer operation from one site to another, planned or unplanned. In a GDPS/XRC configuration, the SDM is placed outside the production sysplex. Normally it is located at the recovery site.

One subset of the GDPS solution is the Remote Copy Management Facility (RCMF) offering. RCMF is an automated disk subsystem and RCMF, with a high level user interface. This interface is implemented in the form of ISPF-type displays and virtually eliminates the tedious and time consuming work with Time Sharing Option (TSO) commands. Managing XRC with RCMF can be the first step of a full GDPS implementation.

## GDPS/PPRC

GDPS/PPRC is designed to manage and protect IT services by handling planned and unplanned exception conditions, and maintain full data integrity across multiple volumes and storage subsystems. By managing both planned and unplanned exception conditions, GDPS/PPRC can help maximizing application availability and provide business continuity. GDPS is capable of providing the following functions:

- ▶ Continuous Availability solution
- ▶ Near transparent disaster recovery solution
- ▶ Recovery time objective (RTO) less than one hour for GDPS/PPRC
- ▶ RTO less than two hours for GDPS/XRC
- ▶ Recovery point objective (RPO) of zero (optional)
- ▶ Protecting against metropolitan area disasters

Figure 3-3 shows a simplified illustration of the physical topology of a GDPS/PPRC implementation, which consists of a Parallel Sysplex spread across two sites (site 1 and site 2) separated by up to 100 km of fiber with one or more z/OS systems at each site. The multisite Parallel Sysplex must be configured with redundant hardware, for example, a CF, and a Sysplex Timer in each site, and alternate cross-site connections.

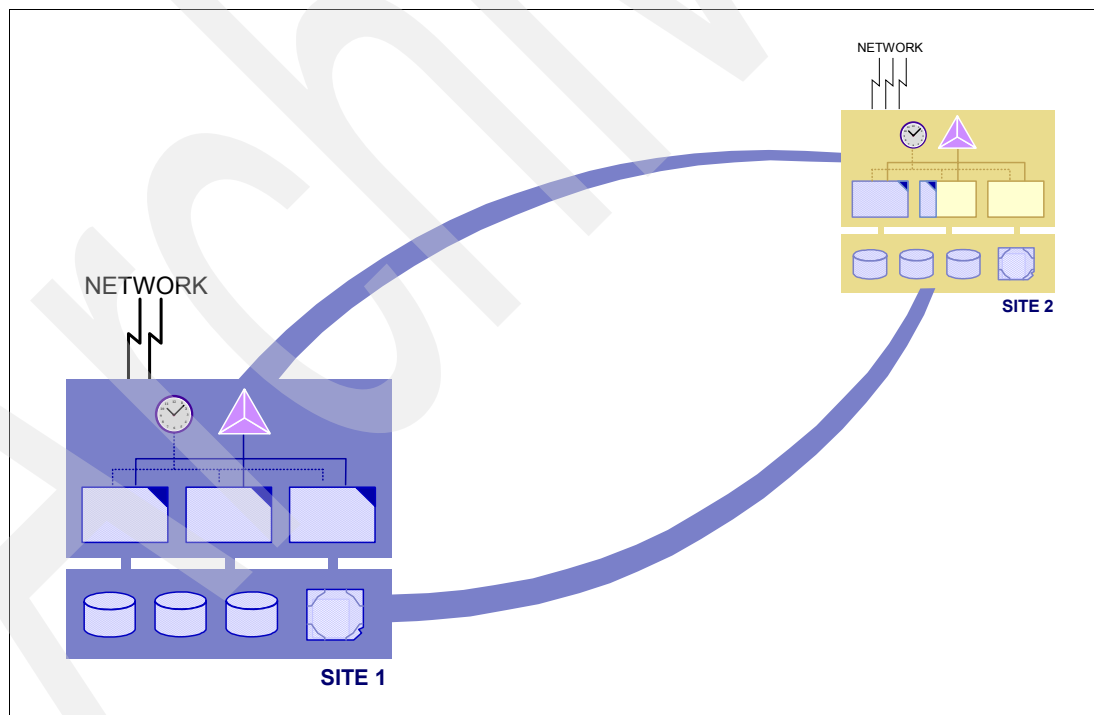


Figure 3-3 GDPS/PPRC

All critical data residing on disk storage subsystems in site 1 (the primary copy of data) are mirrored to the disk storage subsystem in site 2 (the secondary copy of data) through Synchronous PPRC.

GDPS/PPRC provides the Parallel Sysplex cluster continuous availability benefits, and it significantly enhances the capability of an enterprise to recover from disasters and other failures, as well as managing planned actions.

### **GDPS/PPRC HyperSwap**

The GDPS/PPRC HyperSwap function provides the ability to transparently switch the applications I/O activity to the secondary PPRC volumes for both planned and unplanned reconfiguration. Large configurations can be supported, because HyperSwap has been designed to swap a large number of volumes quickly. The important ability to resynchronize incremental disk data changes in both directions between primary and secondary PPRC disks is provided as part of this function.

The GDPS/PPRC HyperSwap function is designed to broaden the continuous availability attributes of GDPS/PPRC. This function can help significantly increase the speed of switching sites and switching disks between sites. The HyperSwap function is designed to be controlled by complete automation, allowing all aspects of the site switch to be controlled through GDPS.

For more information about GDPS, refer to the book *GDPS Family: An Introduction to Concepts and Facilities*, SG24-6374, and to the following Web addresses:

- ▶ Geographically Dispersed Parallel Sysplex: The e-business Availability Solution  
<http://www.ibm.com/servers/eserver/zseries/library/whitepapers/gf225114.html>
- ▶ Implementation Services for GDPS  
<http://www.ibm.com/services/us/index.wss/so/its/a1000189>

### **BACKUP and RESTORE SYSTEM utilities of DB2 for z/OS**

The BACKUP and RESTORE SYSTEM utilities were introduced in DB2 for z/OS V8 and enriched with DB2 9 for z/OS.

#### ***BACKUP SYSTEM utility***

The BACKUP SYSTEM utility fully replaces such steps as “SET LOG SUSPEND - FlashCopy - SET LOG RESUME”, which you might have used with DB2 V7 with a one statement utility and much less impact on current activity. It requires z/OS V1.5, DFSMSHsm™ with the COPYPOOL construct and storage management subsystem (SMS) managed DASD. It can be used to back up the data copy pool (including the integrated catalog facility (ICF) catalogs that define the DB2 Catalog/Directory and the application data) and the log copy pool (including ICF catalogs for the active logs, bootstrap data sets (BSDSs), and DASD archive logs) for a full backup. It can also back up only the data copy pool, assuming that you will provide all the logs that are required from the point of backup to recover the entire subsystem to a given point-in-time.

The BACKUP SYSTEM utility performs the following main tasks:

- ▶ Allows most updates
- ▶ Disables 32 KB page writes, if the CI size is not 32 KB, to eliminate integrity exposure
- ▶ Disables system checkpoint, data set creation, deletion, rename, and extension operations
- ▶ Acquires new PITR lock in X mode to ensure no restore is taking place on other members
- ▶ Records the recover based log point (RBLP) in the BSDS

This is the starting point for the log apply phase that uses the RESTORE SYSTEM utility.



During the backup, DB2 does the following operations:

- ▶ DB2 invokes the DFSMSHsm ARCHSEND service to take full volume DASD-to-DASD copies of the DB COPYPOOL DFSMSHsm, schedules multiple tasks, and makes copies of the volumes in parallel by using FlashCopy.
- ▶ ARCHSEND returns control to DB2 after the logical copies have completed for all volumes in the DATABASE copy pool (normally several minutes).
- ▶ DB2 updates the BSDS with the system backup information.
- ▶ DB2 invokes DFSMSHsm ARCHSEND to take full volume copies of the LOG copy pool, in the same manner as occurred for the DATABASE copy pool.
- ▶ DB2 then resumes activities.

In DB2 for z/OS V8, the BACKUP SYSTEM utility is designed for local subsystem point-in-time recovery and is global in scope. That is, it pertains to an entire DB2 subsystem or data sharing group and is not to a limited number of objects. It is not directly usable for a disaster recovery solution, because the copy pools are stored on DASD by generation.

### ***RESTORE SYSTEM utility***

You use the RESTORE SYSTEM utility only when you want to recover a subsystem or data sharing group to an arbitrary point-in-time or to the current time. The utility restores only the database copy pool of data. Then it applies logs until it reaches a point in the log that is equal to the log truncation point specified in a point-in-time conditional restart control record (SYSPITR CRCR) that is created with DSNJU003 or to the current conditional restart control record. The RESTORE SYSTEM utility uses the RBLP that is stored in the BSDS by the BACKUP SYSTEM utility as the log scan starting point. The log apply phase uses Fast Log Apply to recover objects in parallel. DB2 handles table space and index space creates, drops, and extends, and marks objects that have had LOG NO events as RECP (table spaces and indices with the COPY YES attribute) or RBDP (indices with COPY NO attribute).

To restore a system to a prior point-in-time with the RESTORE SYSTEM utility:

1. Stop DB2. If data sharing, stop all members.
2. Use DSNJU003 to create a SYSPITR CRCR and specify the point to which you want to recover the system. If data sharing, create a SYSPITR CRCR for each member.
3. If data sharing, delete all coupling facility structures.
4. Start DB2. If data sharing, start all members of the data sharing group.  
DB2 enters into system recover-pending mode, ACCESS(MAINT), and bypasses recovery except for in doubt units of recovery (URs).
5. Execute the RESTORE SYSTEM utility. The utility must be completed on the original submitting member.
  - a. It restores the most recent database copy pool version that was taken prior to the log truncation point.
  - b. It performs a log apply function.
6. If a method other than the BACKUP SYSTEM utility was used to copy the system, restore the data manually and use RESTORE SYSTEM LOGONLY. This option can be used with z/OS V1R3 and later.
  - a. It backs up data with another volume dump solution and uses SET LOG SUSPEND/RESUME.
  - b. It performs a log apply function only.

7. Stop DB2 to reset the system recovery-pending status. If data sharing, stop all members.
8. Display and terminate any active utilities.
9. Display restricted objects, recover objects in RECP status, and rebuild objects in RBDP status.

### ***DB2 9 for z/OS enhancement for the BACKUP and RESTORE SYSTEM utilities***

The BACKUP and RESTORE SYSTEM utilities were added in DB2 for z/OS V8 and use disk volume FlashCopy backups and copy pool z/OS DFSMSHsm V1R5 constructs. In DB2 9 for z/OS, these utilities are enhanced to use new functions that are available with z/OS V1R8 DFSMSHsm:

- ▶ **Object-level recovery**

The backups taken by the BACKUP SYSTEM utility in V8 are used for recovery of the entire DB2 system. With an enhancement in DB2 9 for z/OS, a DB2 object can be recovered from system-level backups. Recovery is done by the RECOVER utility, which is now capable of using system-level backups for the restore phase in addition to image copies.

- ▶ **Dump to tape**

Maintaining multiple copies of all the data on disk can be expensive. With DB2 9 for z/OS, you can implement the backup directly to tape. To use this functionality, you must have z/OS DFSMSHsm V1R8 or later.

- ▶ **Incremental FlashCopy**

The physical copying to disk in the background can be improved by the use of incremental FlashCopy. Even if incremental FlashCopy is used, the dump to tape is always a full dump. Incremental FlashCopy has no benefit when dumping to tape except that the dumping to tape might begin earlier because less data must be copied to disk before writing to tape can be started.

For more details on the disaster recovery related to System z, refer to the following Redbooks publications:

- ▶ *Disaster Recovery with DB2 UDB for z/OS*, SG24-63700
- ▶ *GDPS Family: An Introduction to Concepts and Facilities*, SG24-6374
- ▶ *IBM System Storage DS8000 Series: Architecture and Implementation*, SG24-6786

## **3.2.6 I/O connectivity**

Because the amount of data in a data warehouse environment is increasing everyday, the ability to provide fast data access to the processor unit is required. For example, while building an ad hoc report that can read a large table space, fast data access to the CPU determines how quickly that report can be built. System z offers FICON Express4, which is a new generation of FICON and Fibre Channel Protocol (FCP) features that provide fast data access with 1, 2, and 4 Gbps auto-negotiating links. FICON Express4 helps to support increased CPU performance and to meet the need for increased application performance, while providing a manageable migration to higher speed. FICON Express4 continues the tradition of a robust balanced I/O system design on the System z platform.

FICON Express4 and other System z9® channel enhancements help to improve channel performance and provide support for more devices. They also aid in supporting standards-based FCP enhancements that help improve resource sharing and access control for Linux on System z environments.

FICON distance and bandwidth capabilities also make it an essential and cost effective component of data high availability and disaster recovery solutions when combined with System z Parallel Sysplex and GDPS technology. Parallel Sysplex provides resource sharing, workload balancing, and continuous availability benefits. GDPS provides system-level automation that enables the most advanced, application-transparent, multisite disaster recovery solution with fast recovery time. FICON distance support offers two unrepeatable distance options (4 km and 10 km) when using single mode fiber optic cabling.

All FICON Express4 and FICON Express2 features support the Modified Indirect Data Address Word (MIDAW) facility. MIDAW is new system architecture with software exploitation that helps to improve channel utilization, reduce channel overhead, and potentially reduce I/O response times.

For more details about FICON, refer to the I/O Connectivity Web page at the following address:

<http://www.ibm.com/systems/z/hardware/connectivity/news.html>

### 3.2.7 Parallel access volumes

High I/O delays from the data storage devices can lead to performance problems in data warehouse queries, which sometimes do a high amount of multiple reads from one particular volume. Parallel access volume (PAV) enables a single System z server to simultaneously process multiple I/O operations to the same logical volume, which can significantly reduce device queue delays (IOSQ time). This is achieved by defining multiple addresses per volume. With dynamic PAV, the assignment of addresses to volumes can be automatically managed to help the workload meet its performance objectives and reduce overall queuing.

With PAV, reads are simultaneous. Writes to different domains (a set of tracks on which the disk controller is working) are simultaneous as well. However writes to the same domain are serialized. No double updates are possible to preserve integrity. Large volumes, such as 3390 mod 9, 27, and 54, greatly benefit by using PAV. Multiple paths or channels for a volume have been around for many years. However, multiple unit control blocks (UCBs, which are the same as MVS™ addresses) were only introduced with PAVs.

#### Multiple allegiance

The DS8000 accepts multiple I/O requests from different hosts to the same device address, increasing parallelism and reducing channel overhead. In older storage disk subsystems, a device had an *implicit allegiance*, which is a relationship that is created in the control unit between the device and a channel path group when an I/O operation is accepted by the device. The allegiance causes the control unit to guarantee access (no busy status presented) to the device for the remainder of the channel program over the set of paths associated with the allegiance.

With *multiple allegiance*, the requests are accepted by the DS8000 and all requests are processed in parallel, unless there is a conflict when writing data to a particular extent of the CDK logical volume. Figure 3-4 on page 44 and following characteristics describes the operations of multiple allegiance and PAV:

- ▶ With multiple allegiance, the I/Os come from different system images.
- ▶ With PAVs, the I/Os come from the same system image:
  - For *static PAVs*, aliases are always associated with the same base addresses.
  - For *dynamic PAVs*, aliases are assigned up front, but can be reassigned to any base address as needs dictate by means of the Dynamic Alias Assignment function of the

Workload Manager, a reactive alias assignment. Figure 3-4 illustrates the operation of dynamic PAVs.

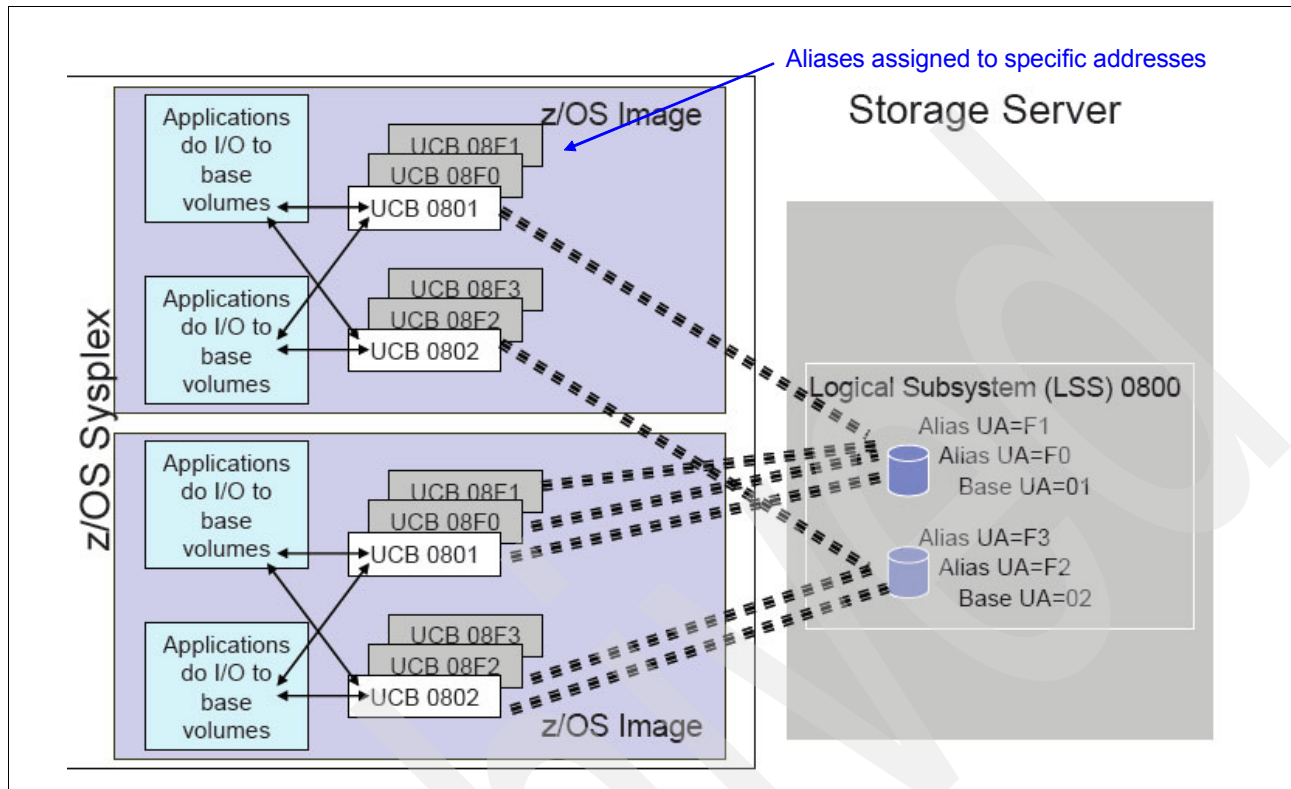


Figure 3-4 Parallel access volumes

## HyperPAV

With HyperPAV, an on demand proactive assignment of aliases is possible. By using HyperPAV, an alias address can be used to access any base on the same control unit image per I/O base. In addition, different HyperPAV hosts can use one alias to access different bases, which reduces the number of alias addresses that are required to support a set of bases in a System z environment with no latency in targeting an alias to a base. This functionality is also designed to enable applications to achieve equal or better performance than possible with the original PAV feature alone while also using the same or fewer operating system resources.

Figure 3-5 illustrates the HyperPAV operations.

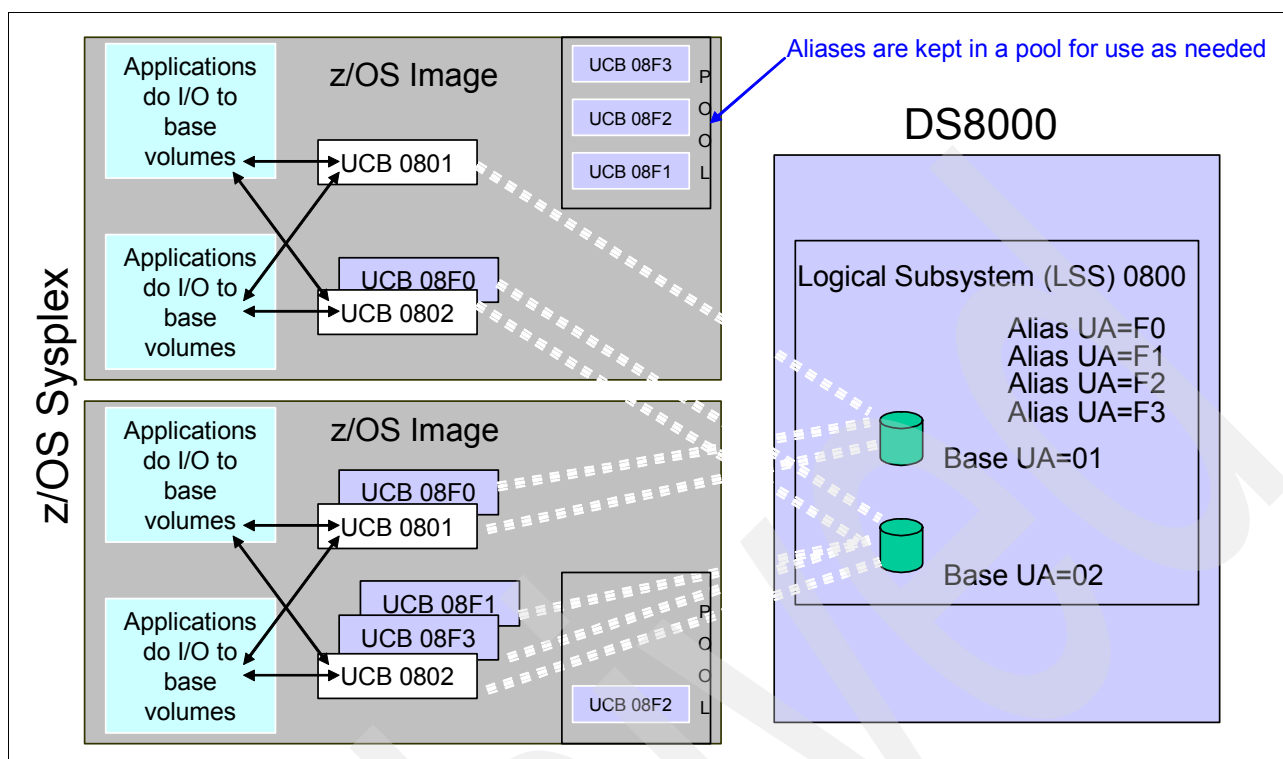


Figure 3-5 HyperPAV

For more information about PAV, HyperPAV, and DS8000 features, refer to *IBM System Storage DS8000 Series: Architecture and Implementation*, SG24-6786.

### 3.2.8 Total cost of ownership

Studies have shown that, when operations and maintenance costs are included in the cost of any system, System z costs become much more favorable and, in some cases, are less than those of other platforms. This is true mostly because many different applications can share a single System z and, in some cases, share the same DB2 subsystem. Therefore, the cost of administering System z and DB2 can be amortized over multiple application workloads, where on other platforms this may not be possible.

IBM continues to introduce innovations that further decrease the total cost of ownership (TCO). The IBM z9 Integrated Information Processor (zIIP), which is designed to maximize resource optimization, is the latest such innovation. It was introduced in 2006 and was immediately exploited by DB2 for z/OS V8. See Figure 3-6 on page 46. The zIIP is priced less than general purpose processors, and the millions of instructions per second (MIPS) it provides do not count toward the software costs of the system. In DB2 V8 and V9 for z/OS, distributed requests over TCP/IP, query requests that use parallelism, and utilities that maintain index structures can exploit zIIPs. BI application costs may directly benefit from DB2 and zIIPs. In DB2 9 for z/OS, remote (Distributed Relational Database Architecture™ (DRDA®)) native SQL procedures also use zIIPs and more are likely to follow.

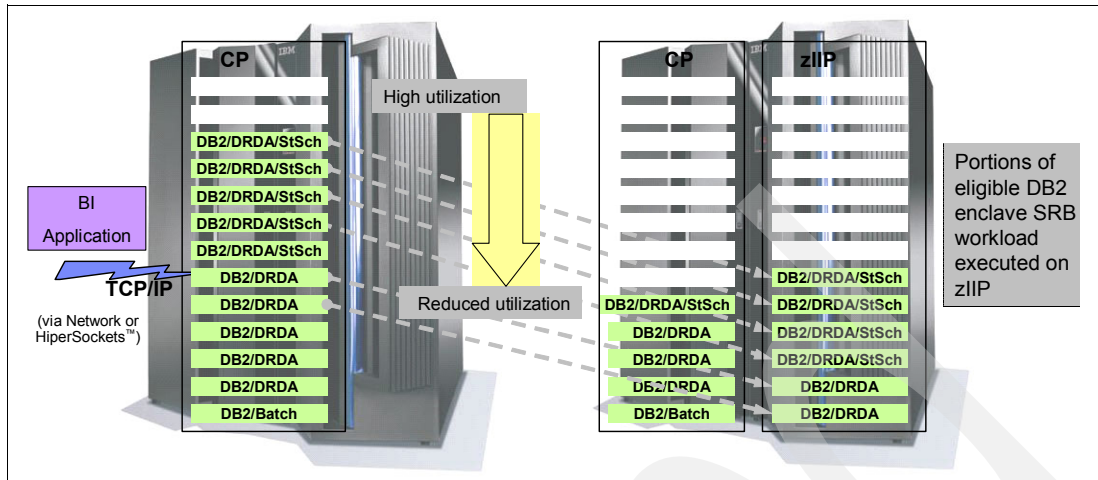


Figure 3-6 The specialty engines

### Specialty engines

Additional TCO improvements can be achieved by making use of the other specialty processors available on the System z platform. The Integrated Facility for Linux (IFL) is used for Linux applications. You can consolidate Linux servers on a single System z9 partition (logical partition (LPAR)), or you can move distributed Linux applications to the System z platform to exploit HiperSockets™ for better performance. The System z Application Assist Processor (zAAP) is used to run Java™ applications.

### System z New Application Charge and DB2 Value Unit Edition

To drive new workload growth on new and existing DB2 for z/OS, IBM is offering customers of DB2 for z/OS one-time-charge (OTC) pricing for qualified *net new workloads* on z/OS. The net new workloads are limited to those that qualify for System z New Application Charge (zNALC) for example, commercial packaged applications, data warehouses, and WebSphere applications running in a partition, machine, or sysplex. The value unit edition (VUE) offering excludes existing qualified workloads.

Refer to the following Web addresses for more information about zNALC and DB2 Value Unit Edition:

- ▶ DB2 for z/OS Value Unit Edition  
<http://www.ibm.com/software/data/db2/zos/edition-vue.html>
- ▶ System z New Application License Charges  
<http://www.ibm.com/servers/eserver/zseries/swprice/zna1c.html>

### 3.2.9 System z10

The System z10 is designed from the ground up to help dramatically increase data center efficiency by significantly improving performance and reducing power, cooling costs, and floor space requirements. It offers unmatched levels of security and automates the management and tracking of IT resources to respond to ever-changing business conditions. The System z10 has the following key features:

- ▶ A single System z10 equal in performance to nearly 1,500 x86 servers
- ▶ Up to 85% less energy costs
- ▶ Up to an 85% smaller footprint
- ▶ Consolidation of x86 software licenses at up to a 30-to-1 (30:1) ratio

- ▶ Quad-Core mainframe
- ▶ The following discipline to data center chaos:
  - Just-in-time capacity to meet ever-changing business conditions
  - Automated management of system performance to favor high-value transactions

Figure 3-7 shows the evolution to System z platform to System z10 compared with the previous generation machines.

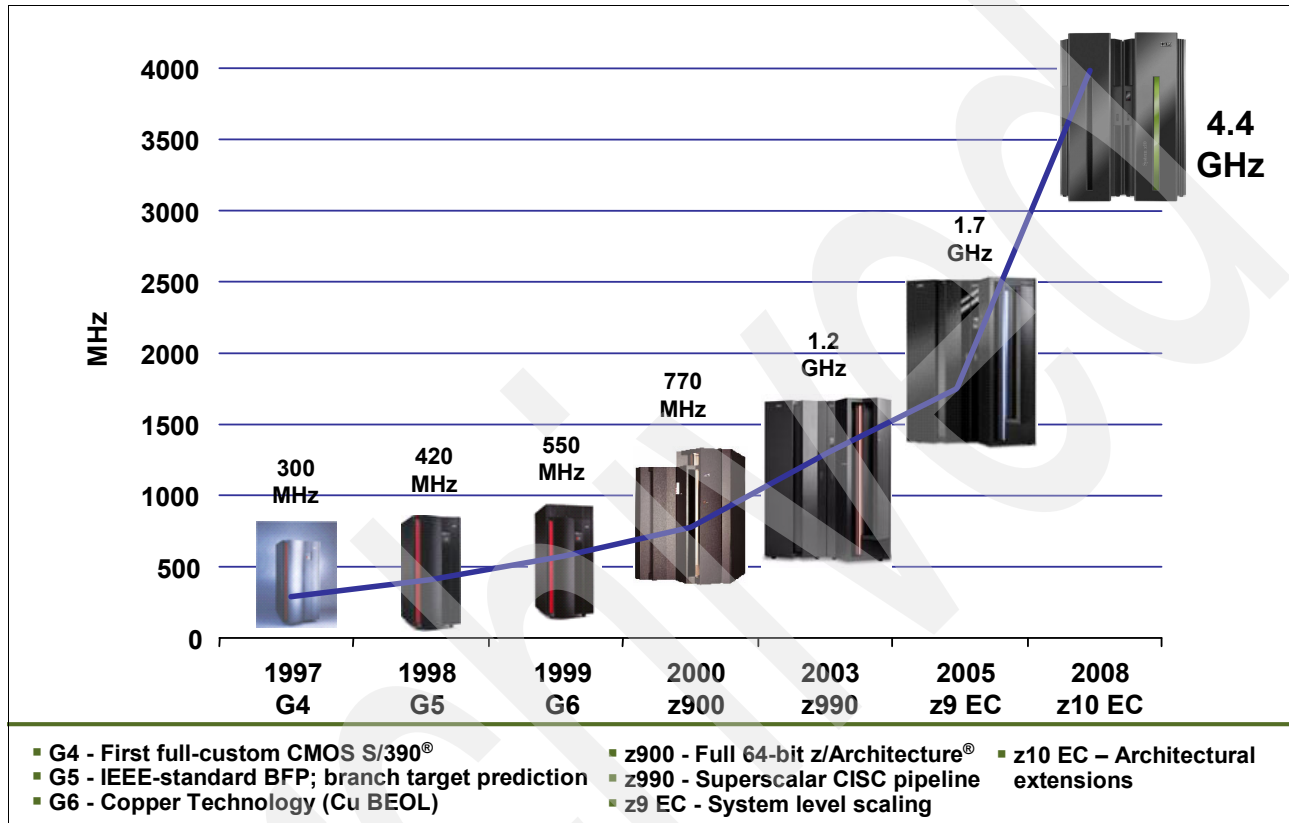


Figure 3-7 Evolution of the System z platform

The System z10 also supports a broad range of workloads. In addition to Linux, XML, Java, WebSphere, and increased workloads from service-oriented architecture (SOA) implementations, IBM is working with Sun™ Microsystems and Sine Nomine Associates to pilot the Open Solaris™ operating system on System z. This collaboration will demonstrate the openness and flexibility of the mainframe.

To learn more about the features of the System z platform (z9BC, z9EC, and z10EC), refer to the following Web addresses:

- ▶ IBM System z9 Business Class  
<http://www.ibm.com/systems/z/hardware/z9bc/features.html>
- ▶ IBM System z9 Enterprise Class  
<http://www.ibm.com/systems/z/hardware/z9ec/features.html>
- ▶ IBM System z10 Enterprise Class  
<http://www.ibm.com/systems/z/hardware/z10ec/features.html>



### 3.2.10 Existing System z customer base

The mainframe has been a solid platform that has delivered value for years. It is a proven performer in both OLTP and data warehouse scenarios. Where a mainframe is installed today with a preponderance of critical IT delivery and support to users, consider the following points about implementing a *data warehouse solution* on the System z platform:

- ▶ Existing infrastructure  
Most of the data feeding the warehouse resides on z/OS and is being continuously updated or added to.
- ▶ Platform preferences and skills in-house  
Regardless of available platforms, this is a System z-centric enterprise that wants to use its existing skills.
- ▶ Budget and cost  
The customer has capacity on the System z platform and does not intend to assume the cost of a new platform, skills, and associated software.
- ▶ Market trends in warehousing solutions and vendors  
Gartner substantiates that the System z is viable as a data warehouse solution.<sup>1</sup> Customer requirements do not dictate an alternative.
- ▶ Complexity  
This entails the sheer volume of data involved, its associated issues if it must be replicated and moved, and other factors.
- ▶ Competitive influences  
The System z solution is adequate to address competitive situations.
- ▶ Mergers and acquisitions or consolidations  
The System z platform in the new joint venture has dominance, or the customer wants to reduce the number of platforms and servers that they currently maintain.
- ▶ Data or application sources  
The majority of the data is on the System z platform and is constantly being added to, updated, and changed. It is too large and *volatile* to re-host.
- ▶ Business processes required and business drivers  
This refers to the customer infrastructure and business process requirements. For example, the intranet or Internet infrastructure is set up for the System z platform with high affinity to warehouse data.

### 3.2.11 DB2 for z/OS with additional data warehousing capabilities

Several DB2 enhancements, such as enhanced index options, new partitioning options, advanced SQL, star join enhancements, and materialized query table (MQT) support, are especially suited for data warehousing.

See Chapter 7, “Functions in DB2 for z/OS for a data warehouse” on page 99, for implementation details.

---

<sup>1</sup> Gartner, Magic Quadrant for data warehouse DBMS Servers, 2006, Publication Date: 25 August 2006, ID Number G00141428 as mentioned in *Exploiting DB2 on System z to meet your data warehousing needs*  
[ftp://ftp.software.ibm.com/software/data/db2bi/data\\_warehousing\\_whitepaper.pdf](ftp://ftp.software.ibm.com/software/data/db2bi/data_warehousing_whitepaper.pdf)



## Table space partitioning

The large volume of data stored in data warehousing environments can introduce challenges to database management and query performance. The table space partitioning feature of DB2 9 for z/OS has the following characteristics:

- ▶ Maximizes the availability or minimizes the run time for specific queries
- ▶ Can have 4096 partitions

A *partition* is a separate physical data set.

- ▶ Allows the loading and refreshing of activities, including the extraction, cleansing, and transformation of data, in a fixed operational window
- ▶ Increases parallelism for queries and utilities
- ▶ Accommodates the growth of data

A universal table space can grow automatically up to 128 TB and adds the functionality of segmented table spaces.

For more details about the partitioning feature of DB2 9 for z/OS, refer to 7.9, “Table space partitioning” on page 148.

## Very large database

DB2 for z/OS can contain an enormous amount of data. DB2 for z/OS can support up to 64,000 databases. With each database containing up to 32,000 objects, it can easily cater to the growing need of a data warehouse environment.

## Star schema enhancements

A common data model that is used in data warehouse environments is the *star schema*, in which a large central fact table is surrounded by numerous dimension tables. Queries generally provide filtering on the independent dimensions, which must be consolidated for efficient access to the fact table.

DB2 for z/OS Version 8 contain the following enhancements, among others, to improve the performance of star schema queries:

- ▶ In-memory work files for efficient access to materialized dimensions or snowflakes
- ▶ Improved cost formula for join sequence determination
- ▶ Predicate localization when OR predicates cross tables

DB2 9 for z/OS further enhances star schema query performance with a new access method. That is Dynamic Index ANDing, for simpler index design, more consistent performance, disaster avoidance, and improved parallelism.

Refer to 7.6, “Star schema processing” on page 124, for implementation details.

## Query parallelism

You can significantly reduce the response time for data or processor-intensive queries by taking advantage of the ability of DB2 to initiate multiple parallel operations when it accesses data in a data warehouse environment.

DB2 for z/OS supports three types of parallelism:

- ▶ I/O parallelism
- ▶ CPU parallelism
- ▶ Sysplex parallelism

## Materialized query tables

MQTs can simplify query processing, greatly improve the performance of dynamic SQL queries, and be particularly effective in data warehousing applications, where you can use them to avoid costly aggregations and joins against large fact tables.

The DB2 optimizer uses partial or entire MQTs to accelerate queries. Its operation and access path are also kept for an easy refresh of the MQT content without specifying the source query again.

Refer to 7.10, “Materialized query tables” on page 152, for more information.

## OLAP functions

New SQL enhancements are made in DB2 9 for z/OS for improving online analytical processing (OLAP) functionalities in a data warehouse. The following OLAP expressions were introduced in DB2 9 for z/OS:

- ▶ RANK and DENSE\_RANK
- ▶ ROW\_NUMBER

Refer to 7.11, “OLAP functions” on page 154.

## Table space and index compression

DB2 for z/OS uses the hardware assisted compression instructions of the System z server for compressing table spaces. DB2 9 for z/OS can also compress index spaces by using software techniques. Table space and index space compression saves a large amount of disk space (in certain cases CPU saving) when implemented in a data warehouse environment, considering the amount of data and the number of indexes that are created for query performance on the large tables. For more details about compression, refer to 7.1, “Index compression” on page 100, and 7.2, “Table space compression” on page 118.

## Index on expression

DB2 9 for z/OS supports the creation of indexes on an expression. The DB2 optimizer can then use such an index to support index matching on an expression. In certain scenarios, it can enhance the query performance. In contrast to simple indexes, where index keys are composed by concatenating one or more table columns specified, the index key values are not exactly the same as the values in the table columns. The values are transformed by the expressions that are specified.

Refer to 7.7, “Index on expressions” on page 139.

## CLONE tables

To overcome the availability problem when running certain utilities, such as LOAD REPLACE in a DB2 for z/OS environment, a cloning feature was introduced in DB2 9 for z/OS. A clone of a table can be created by using the ALTER TABLE SQL statement with the ADD CLONE clause. The clone can then be used by applications, SQL, or utilities, and therefore, provide high availability. Refer to 7.8, “Working with the ADD CLONE SQL command” on page 143, for more details.

## XML support

DB2 9 for z/OS provides pureXML™, which is a native XML storage technology that provides hybrid relational and XML storage capabilities. pureXML provides a huge performance improvement for XML applications while eliminating the need to “shred” XML into traditional relational tables or to store XML usually as character large objects (CLOBs), which are

methods that other vendors use. DB2 9 pureXML exploits z/OS XML System Services for high-performance parsing with improved price performance by using zAAPs and zIIPs.

The pureXML technology includes the following capabilities:

- ▶ XML data type and storage techniques for efficient management of hierarchical structures inherent in XML documents
- ▶ pureXML indexes to speed search subsets of XML documents
- ▶ New query language support (SQL/XML and XPath) based on industry standards and new query optimization techniques
- ▶ Industry-leading support for managing, validating, and evolving XML schemas
- ▶ Comprehensive administrative capabilities, including DB2 utilities and tools
- ▶ Integration with popular APIs and development environments
- ▶ XML shredding, publishing, and relational view facilities for working with existing relational models
- ▶ Proven enterprise-level reliability, availability, scalability, performance, security, and maturity that users expect from DB2

### 3.2.12 Extract, transform, and load on the same platform

Extract, transform, and load (ETL) identifies the processes that extract information from the OLTP system, transform it according to the needs of the data warehouse environment, move it to the platform that houses the data warehouse, and finally loads the data into the data warehouse. The process of extracting information from the OLTP system by necessity runs on the same platform as the OLTP system. The process of transforming it to conform to the needs of the data warehouse is usually performed on the platform where the OLTP system resides. This is because many of the techniques used in the transformation process reduce the amount of data that then must be moved to and loaded into the data warehouse.

Regardless of where the transformation process is performed, it is necessary to move the data to the platform that is housing the data warehouse. Many methods can be employed in this movement of data. Regardless of the method that is employed, if the data warehouse is physically distinct from the platform that is housing the OLTP system, it is necessary to transmit this data over a communications path of some kind. Often this is the single most expensive component of the entire ETL process.

Obviously, if the data warehouse resides on the same platform as the OLTP system, sending data over a communication path is unnecessary. With the Parallel Sysplex capabilities of the System z platform, it is a relatively simple matter to have the data warehouse on the same platform as the OLTP system and, at the same time, on distinct processors or machines from the OLTP system.

The System z platform is capable of handling the different characteristics of OLTP and BI workloads within one logical database system. This can be achieved by DB2 subsystems that are optimized for different workload within a data sharing group. Data sharing is a DB2 feature of exploiting System z Parallel Sysplex technology and, therefore, sharing the workload between multiple DB2 subsystems that access the same data.

Figure 3-8 shows three subsystems at the top that are optimized to handle transactional workload. At the bottom are two subsystems that are optimized for BI. All subsystems are in the same data sharing group.

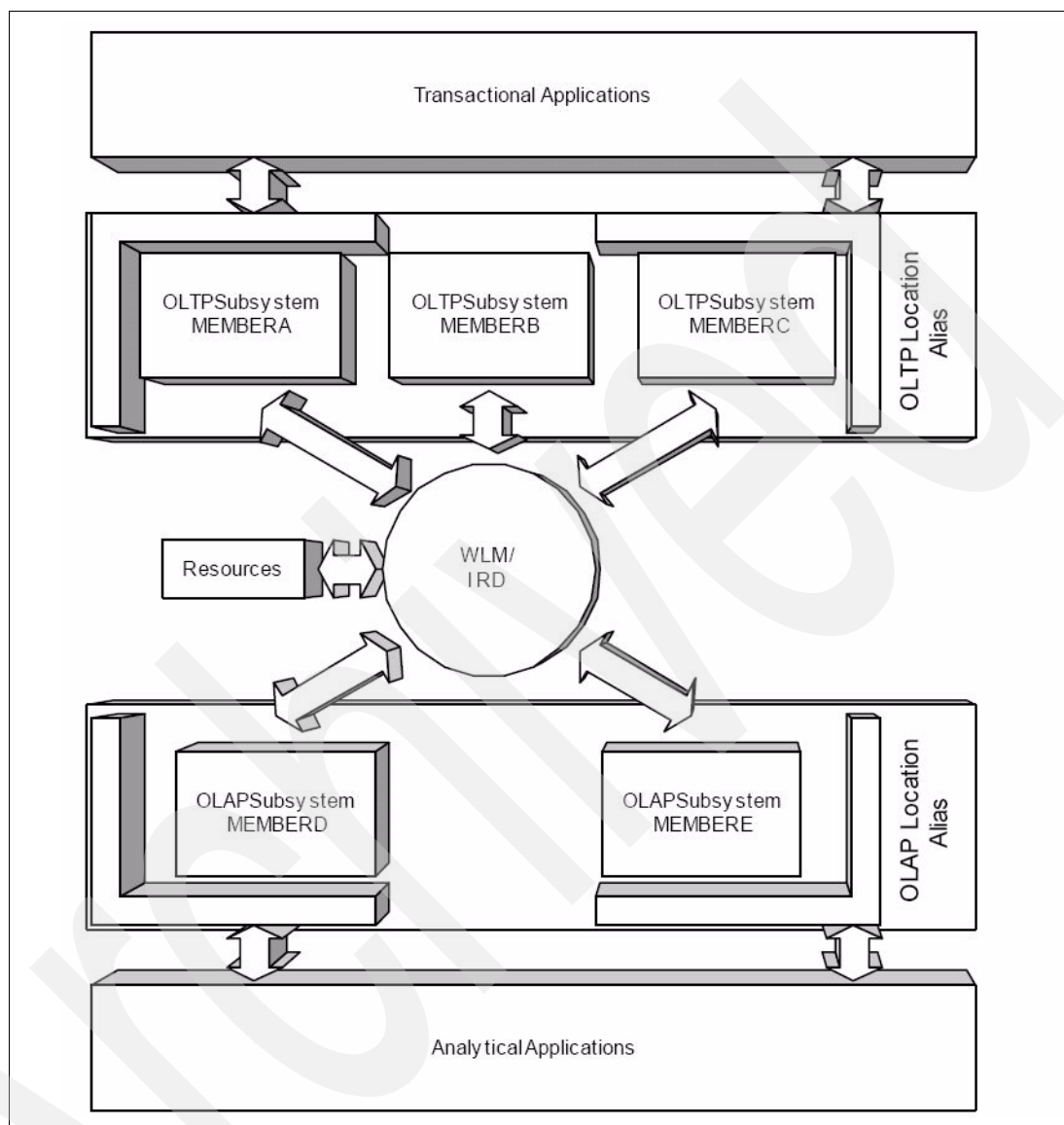


Figure 3-8 Data sharing and mixed workload

If the subsystems are in the same LPAR, the WLM can be configured to assign the correct amount of resources to the subsystems. It can be configured in a way that OLTP subsystems always have a guaranteed amount of resources, no matter how much workload is assigned to the data warehouse members. Further, WLM can be configured to prioritize short-running queries in the BI subsystems.

If the subsystems are not in the same LPAR, the Intelligent Resource Director can manage resources among LPARs. WLM can manage those resources among subsystems within the same LPAR.

The figure also shows that the transactional systems are routed against the data sharing members that are optimized for OLTP workload, and the query and reporting and analytical applications are routed against the data sharing members that are optimized for BI queries.

This is achieved by location alias definitions in DB2. Each application recognizes (“sees”) all data: detailed transactional data and data warehouse aggregates. However, the DB2 subsystems of member D and E are configured with a focus on parallel query execution, while members A, B, and C are used for normal, short-running transactions.

This configuration of System z and DB2 is optimal for the different types of workload that occur if OLTP and BI data is mixed in the same system. It is highly scalable. Detailed data can be accessed in the same way as aggregates, which allows “in database transformations”.

If data movement to another system or platform is desired for some reason, IBM WebSphere II Replication provides the ability to stream high volumes of data with low latency to create near real-time warehouses. Or IBM InfoSphere DataStage can be used to provide complex transformations of the data before loading them into the warehouse.

In addition to DB2 for z/OS as an essential core component hosting data in a data warehouse, other components are required to build a comprehensive solution, mainly to address the following areas:

- ▶ Efficient access to multiple data sources
- ▶ ETL processing
- ▶ Data integration and cleansing
- ▶ Managing incremental updates
- ▶ Analytics and reporting

### **WebSphere Classic Federation**

A data warehouse may collect, consolidate, and aggregate data from various sources and various source types, not all being naturally relational data sources, such as DB2 for z/OS. Some data might reside on System z in the following areas:

- ▶ IMS
- ▶ CICS® controlled VSAM data sets
- ▶ ADABAS
- ▶ Flat files

This data (or at least aggregates of it) must be integrated into the data warehouse so that all information becomes available in a single place.

WebSphere Classic Federation can access this data in a way that makes it unnecessary to set up a specialized extraction process for each of the different data sources, such as hierarchical or relational databases.

The WebSphere Classic Federation Data Server is installed where the data sources reside. A connector is configured and started for each data source that needs to be accessed. These connectors are then used by the data server to access the source data from IMS, VSAM, ADABAS, or other supported sources. The data server maps the various source data structures to a relational structure. Therefore, all data accessed through the WebSphere Classic Federation Data Server looks similar to one relational database, even if the source is structured hierarchically.

Mapping between source and target definitions is done by using the Classic Data Architect, which is an Eclipse-based workstation tool. It allows the importing of IMS DBD or COBOL Copybooks to obtain the structure definitions of the source data. The user can select which information to include in a target table that is simulated as a relational structure by the data server. No further configuration or coding is necessary.

An ETL server may access this data through Java Database Connectivity (JDBC™) or Open Database Connectivity (ODBC) to move it to the data warehouse in the form of staging tables

or aggregates. The ETL server and the WebSphere Classic Federation Data Server are primarily used for either an initial load or a full reload of the data warehouse. Data in legacy data sources can be updated as well, but this is usually not required within a data warehouse environment. Figure 3-9 shows an overview of the components that come with and interact with the WebSphere Classic Federation Server.

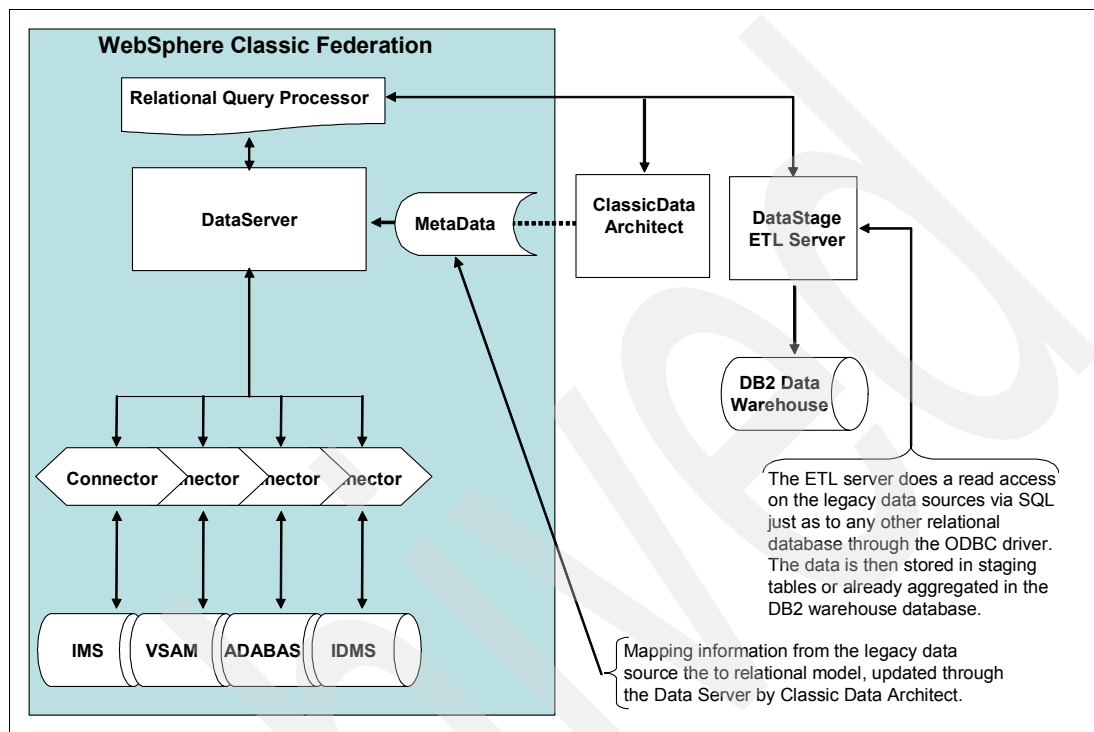


Figure 3-9 WebSphere Classic Federation

## WebSphere Classic Data Event Publisher

After the data is loaded into the data warehouse, it usually must be updated incrementally. It is not necessary to replace the entire content. WebSphere Data Event Publisher and WebSphere Classic Data Event Publisher, respectively, are used to detect the changes in DB2 sources and legacy data sources and provide the information about the changes to the ETL server.

WebSphere Classic Data Event Publisher and Classic Federation have much code in common. If you previously installed and configured WebSphere Classic Federation, you can extend this configuration to allow event publishing. Again, you must map legacy data structures to relational tables by using the Classic Data Architect. You can even configure the server so that this information is used for federation and event publishing at the same time, with just one running data server.

Instead of using a connector to directly access the legacy data sources, a change capture agent is used to detect all manipulations to the data source. In some cases, this is a logger exit (for example, with IMS). In other cases, it is a log reader. The Change Capture Agent sends the change information to the correlation service, which maps the legacy data structure to a relational structure, which is designed with the Classic Data Architect. Finally, Distribution Service and Publisher are used to send the information about the changes through WebSphere MQ to the DataStage data integration server. The DataStage server can read the MQ messages directly from the queue, just as they are transmitted. These events can then be stacked up in staging tables and applied to the data warehouse in a composite update, which is run in a batch window. See Figure 3-10.

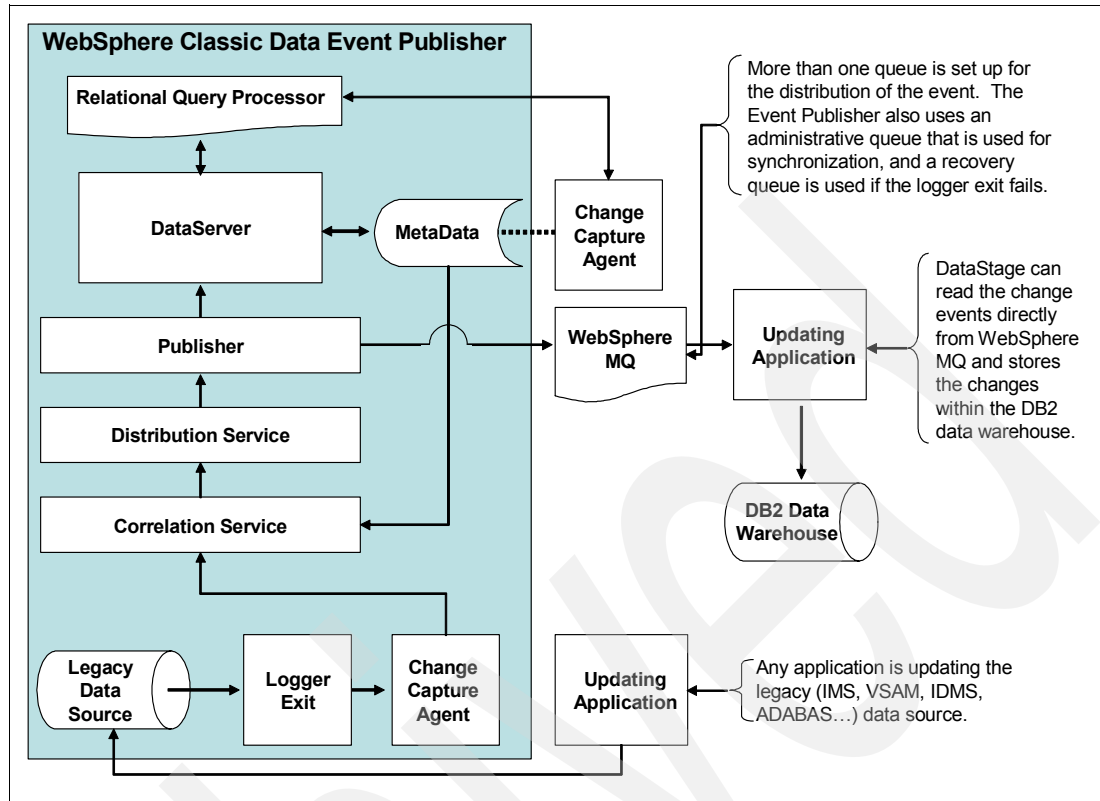


Figure 3-10 WebSphere Classic Data Event Publisher

### WebSphere Data Event Publisher for z/OS and the Q replication server

WebSphere Data Event Publisher for z/OS offers efficient changed-data solutions for DB2 for z/OS databases.

A Q Capture program reads the DB2 log files for transactions that contain changes to the sources in your publications. It places messages that contain those transactions or messages that contain individual changes on queues. You provide the destination for the messages. You can publish XML messages to a Web application, use them to trigger events, and more. You can either start the Q Capture program with the default settings or modify those settings for your environment.

Figure 3-11 on page 56 shows a simple configuration for using Q Capture and event publishing. A queue capture control program reads change information from DB2 for z/OS logs and (based on definition in control tables) makes these changes available in a queue. With WebSphere Data Event Publisher for z/OS, you can propagate the data changes to DataStage and other integration solutions for populating a data warehouse, data mart, or operational data store with the changes.

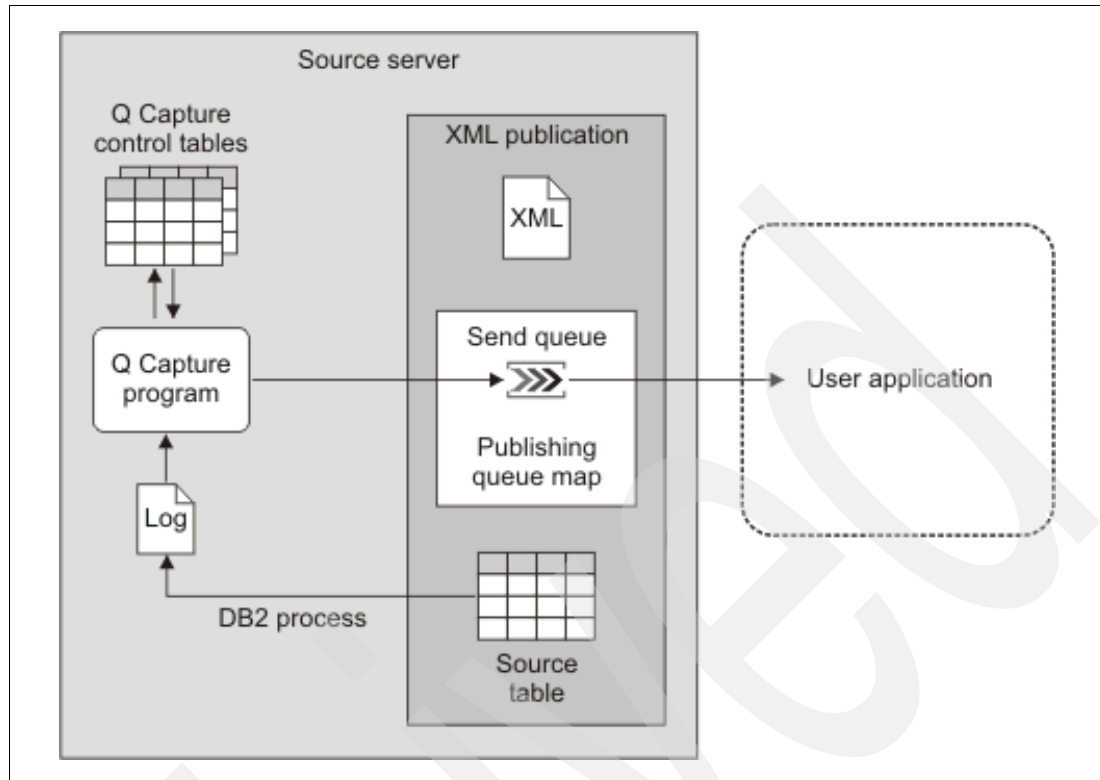


Figure 3-11 Event publishing and Q Capture

## Information Server for System z

The IBM Information Server is a framework for services around data transformation, data quality assurance, metadata management, SOA services, and other operations.

Figure 3-12 on page 57 illustrates the four key integration functions in the Information Server. In the following list, we explain how you can use these components in a data warehouse environment:

- Understand data.

This function is about analyzing data and determining the meaning and relationships of information. Web-based tools can be used to define and annotate fields of business data. Monitoring functions can be used to create reports over time. A metamodel foundation (and the IBM Metadata Server) helps, for example, in managing changes in the transactional (OLTP) data model and their implication in the warehouse model and derived reports.

- Cleanse data.

In a data warehouse environment, multiple data sources maintained by different applications are to be integrated. Consequently, the data values in the transactional system may be stored in different formats and independent systems are likely to duplicate data. The QualityStage component in Information Server supports consolidation, validation, and standardization of data from these multiple data sources, thereby helping to build a consistent, accurate, and comprehensive warehouse model.

- Transform data.

Transforming data in the context of an ETL process is a major requirement for a comprehensive data warehouse solution. The DataStage component in Information Server provides a variety of connectors and transformation functions for this purpose.



High-speed join and sort operations, as well as the parallel engine, help in implementing high-volume and complex data transformation and movement. A comprehensive set of tools are offered to monitor, manage, and schedule ETL jobs.

► Deliver data.

In addition to the components mentioned earlier in this section, Information Server comes with many ready-to-use native connectors to various data sources, located on distributed or mainframe systems.

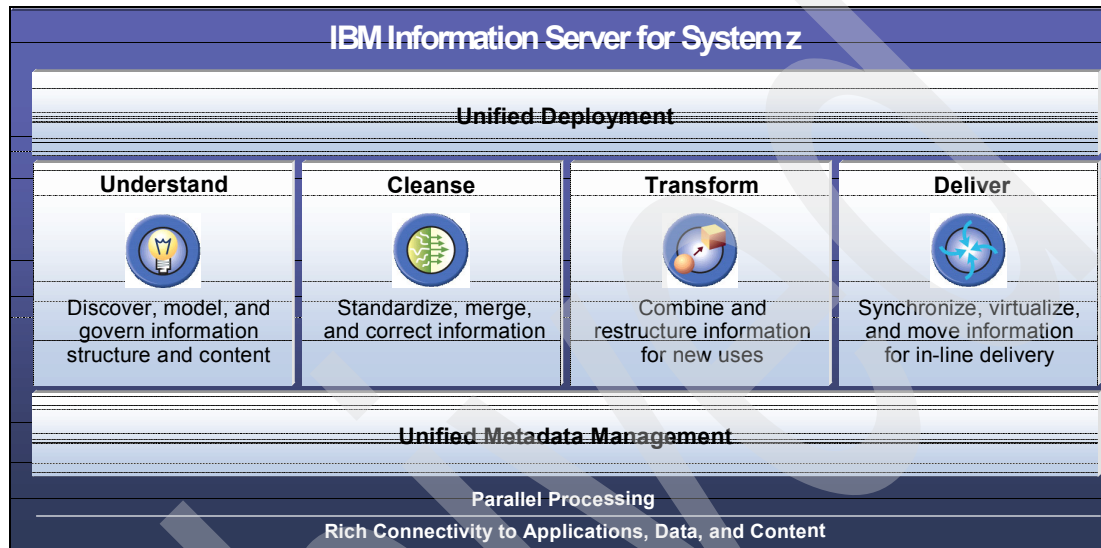


Figure 3-12 Information Server for System z

Information Server 8.0.1 comes with a server and a runtime component. For a System z-based implementation, the server component may be installed on Linux on System z. The client components can be installed on a Microsoft® Windows® system and are used to design and model data and ETL jobs for DataStage and QualityStage.

**Other versions of Information Server:** An Information Server version is available for z/OS (running with UNIX® System Services), and an MVS version of the product is available. Both are currently based on older product levels compared to the one that we used on Linux on System z. Particularly when running multiple jobs concurrently, the available product version on Linux on System z may have advantages.

### Cognos, AlphaBlox, DataQuant, and QMF

Reporting, BI analytics, and queries are an integral part of a data warehouse solution. At the time of writing this book, the following product versions are available on the System z platform:

- Cognos 8 BI Server (now available for Linux on System z)
- AlphaBlox (available for Linux on System z)
- DataQuant (available on z/OS)
- IBM QMF™ (available on z/OS)

For details, see Chapter 14, “Reporting with DataQuant, QMF, and AlphaBlox” on page 397, and Chapter 13, “Reporting and analysis with Cognos 8 BI” on page 305.

Archived

## The architecture for the BI solution on System z

In the previous chapters, we provide a short overview of the data warehouse characteristics and explain the System z functions that can help in implementing a business intelligence (BI) solution. In this chapter, we describe the business requirements that must be satisfied by the BI solution, the architecture implemented for the solution during this project, and other possible configuration scenarios.

This chapter contains the following sections:

- ▶ 4.1, “Business requirements for a data warehouse environment” on page 60
- ▶ 4.2, “The business intelligence architecture with System z” on page 61
- ▶ 4.4, “Business requirements revisited” on page 68

## 4.1 Business requirements for a data warehouse environment

Organizations that have an existing data warehouse environment or are building a new data warehouse environment must consider the following characteristics in their setup:

- Flexibility

With rapid changing business scenario's, a BI solutions must be flexible to manage demands for new data sources and new application requirements. For example, assume that a large organization acquires a small organization. The BI solution of the large organization should be flexible to incorporate the data of the acquired organization.

- High data volumes

Such regulations as the Sarbanes-Oxley Act (SOX), Basel II, Data Protection Act (United Kingdom), and the U.S. Patriot Act were created to protect investors' interests, avoid fraud, and improve financial reporting. Regulations also emphasize the growing need to reproduce versions of data, applications, and even entire business states. This regulation require companies to keep a long record of activities. The implementation of these regulations leads to large data storage requirements. The increasing number of users and services adds additional storage requirements.

- Near real-time data

With the emergence of dynamic data warehousing, queries in a BI environment need access to near real-time data. The data changes from online transaction processing (OLTP) should be replicated to the data warehouse database with no or small latency. The data transfer from OLTP to the data warehouse database preferably should not be done over a communication line because it will increase latency.

- High availability and recovery

With the emergence of dynamic data warehousing, a data warehouse environment should be available 24x7. Downtime due to database changes, operating system, or software upgrades is no longer acceptable.

- Query and reporting

To implement a good BI solution, a reporting tool is required to generate various forms of reports, charts, graphs, cube views, and so on to make quick business decisions.

- Management

Enterprise likely need a BI environment that is easily manageable and does not require a large number of people to manage. Tools that support the administration of a BI environment are of a great advantage.

- OLTP performance

The data warehouse data is initially extracted from the OLTP environment, and ongoing updates are required. The performance impact on OLTP must be minimized.

- Different data sources

Easy integration of various data sources to a data warehouse database is required. In today's environment, the data is derived from several sources. In addition to DB2 databases on System z and on distributed systems, non-DB2 databases, legacy sources on System z, such as IMS and VSAM, and multiple formats from distributed systems must be accessed.

- Security

Data warehouses hold data that contains confidential information such as customer details, business processes, and business strengths. This data must be kept secured to avoid any intrusion.

## 4.2 The business intelligence architecture with System z

Figure 4-1 shows a possible architecture for a data warehouse on the System z platform. The components are described in more detail later in this section. The components can be arranged in various configurations, using the different configuration possibilities that are provided by System z, for example monoplex, sysplex, number of logical partitions (LPARs) in the sysplex, and DB2 data sharing.

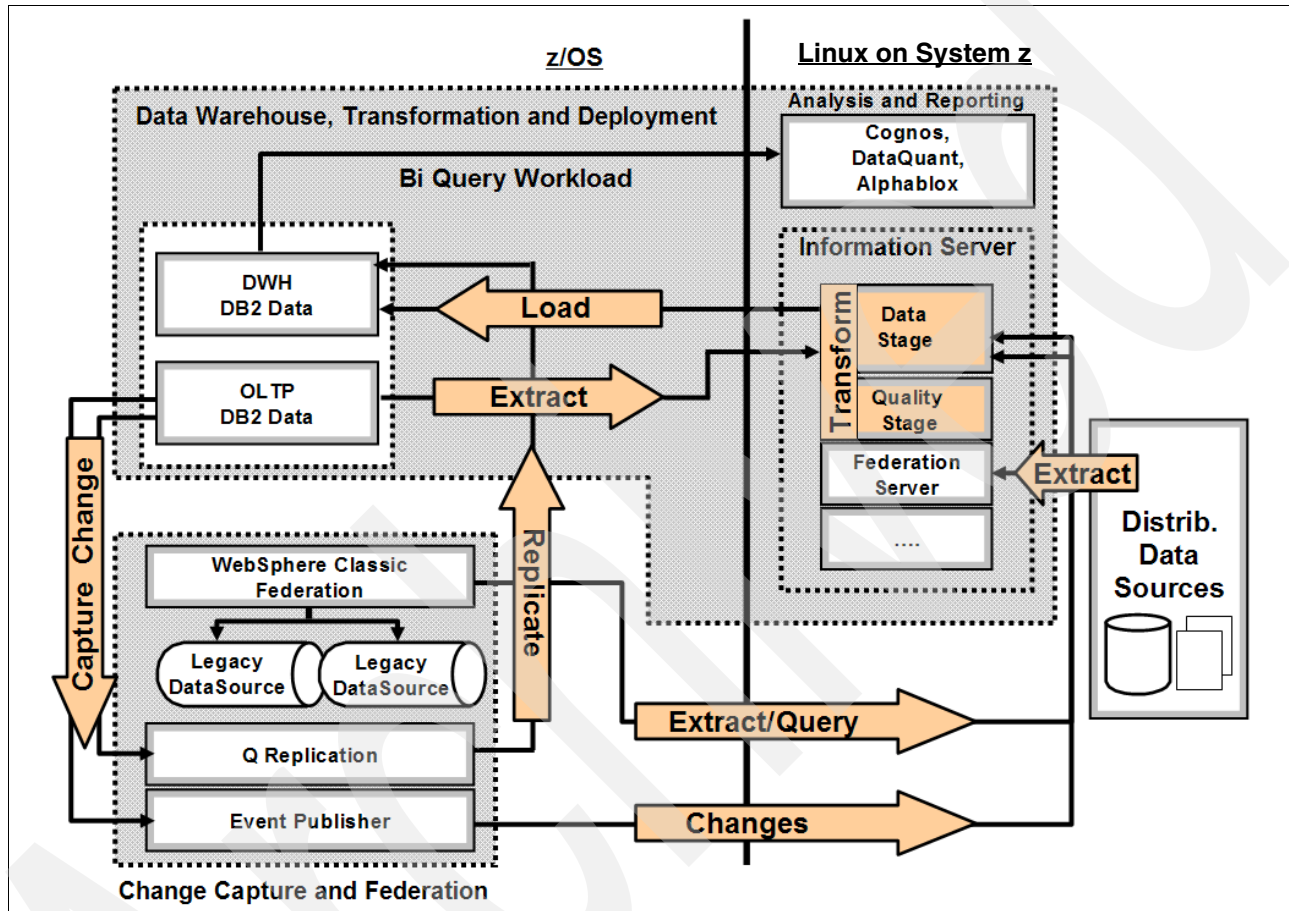


Figure 4-1 Data flow across components

Figure 4-1 shows several paths to update the data warehouse:

- ▶ OLTP database → Change Capture → Q Replication → Data warehouse database  
This path is for tables that must be replicated with minor transformations and low latency.
- ▶ OLTP database → Change Capture → Event Publisher → DataStage → Data warehouse database  
This path is used to propagate selected changes in predefined intervals to DataStage. DataStage processes the captured data (for example, aggregation and join with other sources) and loads it into the data warehouse database. If data cleansing is required, the data is fed into QualityStage before going to DataStage.
- ▶ OLTP database → DataStage → Data warehouse database  
This path is entirely controlled by DataStage. DataStage extracts data from the OLTP database, does any processing that is required, and loads data into the data warehouse database. This path can be used during the initial load of the data warehouse database.

- Legacy data → WebSphere Classic Federation → DataStage → Data warehouse database

WebSphere Classic Federation running on z/OS simulates the legacy data (for example, IMS, VSAM, and Sequential file) as a relational database management system (RDBMS). It is then read by DataStage, processed as required, and loaded into the data warehouse database.

- Distributed data sources → Federation Server → DataStage → Data warehouse database  
This path is used to use distributed data sources, such as various databases (IBM and third-party products), and other sources, such as flat files or spreadsheets, as input to the data warehouse database.
- Analysis and reporting are performed with such products as Cognos 8 BI, DataQuant, and AlphaBlox. These products access the data warehouse database and generate the required reports, spreadsheets, dashboards, and so on.

## 4.2.1 The components involved

The components can be ordered into the following groups:

- Base infrastructure

- Parallel Sysplex

The Parallel Sysplex is already described in 3.2.1, “Availability and scalability” on page 32. The Parallel Sysplex is used in the configuration options that are described later in this chapter.

- Linux on System z

Linux on System z runs on Integrated Facility for Linux (IFL) processors or the general purpose processors. It hosts products for the extract, transform, and load (ETL) process, such as Information Server, and the Analysis and Reporting components.

- Database management system for the OLTP and data warehouse

- DB2 9 for z/OS

The architecture on the System z platform can be implemented with one DB2 9 data sharing group for both OLTP and the data warehouse or with separate DB2 9 data sharing groups for OLTP transactions and one data sharing group for the data warehouse database. For a description of data sharing, see 3.2.1, “Availability and scalability” on page 32.

Implementation with a single data sharing group for both databases gives performance benefits if in-database transformations are used. The trade-off is the connection of shared objects, such as the db2 catalog, directory tables, and EDM pools, between OLTP and the data warehouse database.

- Products to implement the ETL process

ETL is the process is to extract data from OLTP, transform it according to BI solution needs, and load it into data warehouse database. On System z, the ETL process is done by using the following products:

- WebSphere Replication Server - Q Replication

Q Replication executes on z/OS and replicates changes to another data source with small or no latency. In a dynamic data warehouse environment, requirements indicate that changes in OLTP data should be propagated to a data warehouse database with almost no latency for BI queries to have access to near- real-time data. See 8.2.2,

“Configuring Q replication subscriptions” on page 171, for details about implementation.

- WebSphere MQ for z/OS

Depending on the type of replication, WebSphere MQ interface is required by WebSphere Replication Server - Q Replication to safely transfer captured changes to the apply program running on the remote system. See “Creating the WebSphere MQ objects” on page 165 for details about use.

- WebSphere Classic Event Publisher for z/OS

WebSphere Classic Event Publisher executing on z/OS is used to capture changes made in the OLTP database and provides these changes to DataStage via a message queue. DataStage does the required processing and loads the derived information into the data warehouse database. See 8.2.3, “Configuring event publishing using the DB2 Replication Center” on page 183, for details.

- Information Server

Information Server executing on Linux on System z is used to profile, cleanse, and transform information from mainframe and distributed data sources. Refer to 9.1, “Overview of components for ETL on Linux on System z and z/OS” on page 202, for details about the ETL components of Information Server. The following components of Information Server are used in data warehouse environment:

- InfoSphere DataStage

Extracts and transforms data from multiple sources and loads it onto the data warehouse database.

- InfoSphere QualityStage

Cleans and standardizes information from heterogeneous sources.

- Federation Server

Integrates diverse and distributed information sources.

- WebSphere Classic Federation

This product executes on z/OS and is used to access data from legacy data sources. It is used to integrate the data from sources, such as VSAM, IMS, and Sequential file, to a data warehouse database, which then becomes a central repository.

- Products for business intelligence analysis

A couple of choices are available to provide analytics to the user. In our scenario, we use Cognos 8 BI, as described in Chapter 13, “Reporting and analysis with Cognos 8 BI” on page 305. Other choices are DataQuant and AlphaBlox as described in Chapter 14, “Reporting with DataQuant, QMF, and AlphaBlox” on page 397.

## 4.2.2 Configuration alternatives

You can use the products that we previously describe to build various different configurations. The following configurations are the most common:

- Configuration with one DB2 data sharing group

The components that are involved in this configuration are two z/OS LPAR's in Parallel SYSPLEX, one data sharing group, ETL tools, one WebSphere MQ queue sharing group, and one Linux for System z LPAR.

Figure 4-2 on page 64 shows a configuration that is made up of a Parallel SYSPLEX with two LPARs and two coupling facilities (one for the backup). DB2 9 for z/OS is configured in one data sharing group with four DB2 members: two for handling OLTP transactions and

two for data warehouse queries. This can be implemented by configuring the DB2 connection to a particular DB2 member in a data sharing group. In this configuration, DB2 is configured for high availability and scalability. It also gives better performance when we have “in-database transformations”.

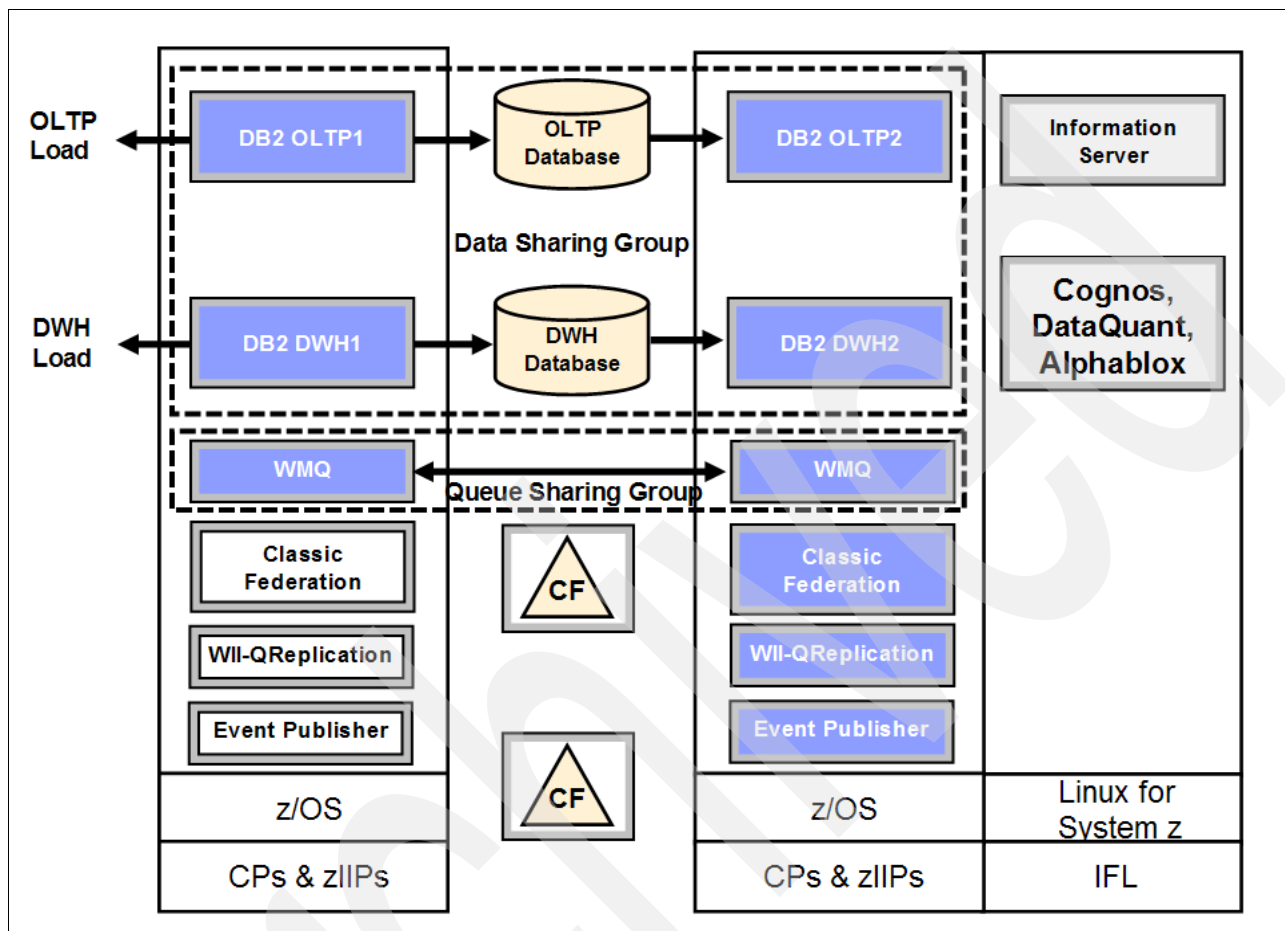


Figure 4-2 Data warehouse architecture on System z with one data sharing groups

With this DB2 configuration, consider the following points:

- Shared resources  
Both OLTP and the data warehouse subsystem must share some DB2 resources, such as Catalog, Directory tables, and EDM pool. The sharing of resources in certain cases can impact performance of OLTP transactions, which is not acceptable in a production environment.
- Tuning and monitoring overhead  
To avoid contention for DB2 resources between OLTP transactions and data warehouse queries, more effort is required to tune the environment in doing such tasks as Workload Monitor (WLM) tuning, buffer pool separations, group buffer pool (GBP) management, monitoring the lock contention between OLTP transactions, and data warehouse queries.
- Inability to use the sysplex load balancing feature to distribute workload over multiple members  
The workload is directed to the particular DB2 member (OLTP or data warehouse member) instead of the complete data sharing group.



Appropriate WLM policies must be defined to assign the right priorities for the members.

The ETL tools for z/OS are identically configured in both LPARs. The ETL components of one LPAR are started (drawn with color), and the components in the other LPAR are started by Automatic Restart Manager (ARM) in case of an LPAR failure. In case of a component failure, ARM starts the component in the same LPAR.

WebSphere MQ for z/OS is configured in a queue sharing group to provide high availability of the queues used by WebSphere Replication Server - Q Replication.

The Information Server is used for transformation and filtering and reporting tools, such as Cognos software, Alpha Box, and DataQuant, running on Linux for System z, which runs on specialist processor IFL. Information Server uses WebSphere Application Server on Linux on System z, therefore, to provide high availability of WebSphere Application Server similar to the components that we discussed earlier in this section. WebSphere Application Server can be configured in a cluster mode with multiple WebSphere Application Server profiles on two or more Linux on System z LPARs and proper high availability concepts for metadata hosted on Linux on System z.

For high availability of Cognos 8 BI, Alpha Box, and DataQuant, they can be configured under standby mode on the other Linux for System z LPARs. The components that are used by ETL and the reporting tools on Linux on System z can be configured in separate Linux images to gain better isolation and workload management.

For more details about high availability architectures of Linux on System z, refer to “High Availability Architectures For Linux on IBM System z” white paper at the following address:

[http://www.ibm.com/servers/eserver/zseries/library/whitepapers/pdf/HA\\_Architectures\\_for\\_Linux\\_on\\_System\\_z.pdf](http://www.ibm.com/servers/eserver/zseries/library/whitepapers/pdf/HA_Architectures_for_Linux_on_System_z.pdf)

► Configuration with two DB2 data sharing group

The components involved in this configuration are two z/OS LPARs in Parallel Sysplex, one data sharing group, ETL tools, one WebSphere MQ queue sharing group, and one Linux for System z LPAR.

Figure 4-3 on page 66 shows a configuration that is made up of a Parallel Sysplex with two LPARs and two coupling facilities (one for the backup). DB2 9 for z/OS is configured in two data sharing groups: one for OLTP and one for data warehouse workload respectively. Each data sharing group contains two DB2 members, one member on each LPAR. This configuration is built for the high availability, scalability, and isolation between the two types workloads. The sysplex distributor can be used with WLM to distribute the workload among the members depending upon their current status and the workload running on them.

Appropriate WLM policies must be defined to assign the right priorities for the members of different data sharing groups.

The ETL tools for z/OS are identically configured in both the LPARs. The ETL components of one LPAR are started (drawn with color), and the components in the other LPAR are started by ARM in case of an LPAR failure. In case of a component failure, ARM starts the component in the same LPAR.

WebSphere MQ for z/OS configured in a queue sharing group to provide high availability of the queues that are used by WebSphere Replication Server - Q Replication.

The Information Server is used for transformation and filtering and reporting tools, such as Cognos 8 BI, AlphaBlox, and DataQuant running on Linux for System z, which runs on the IFL specialist processor.

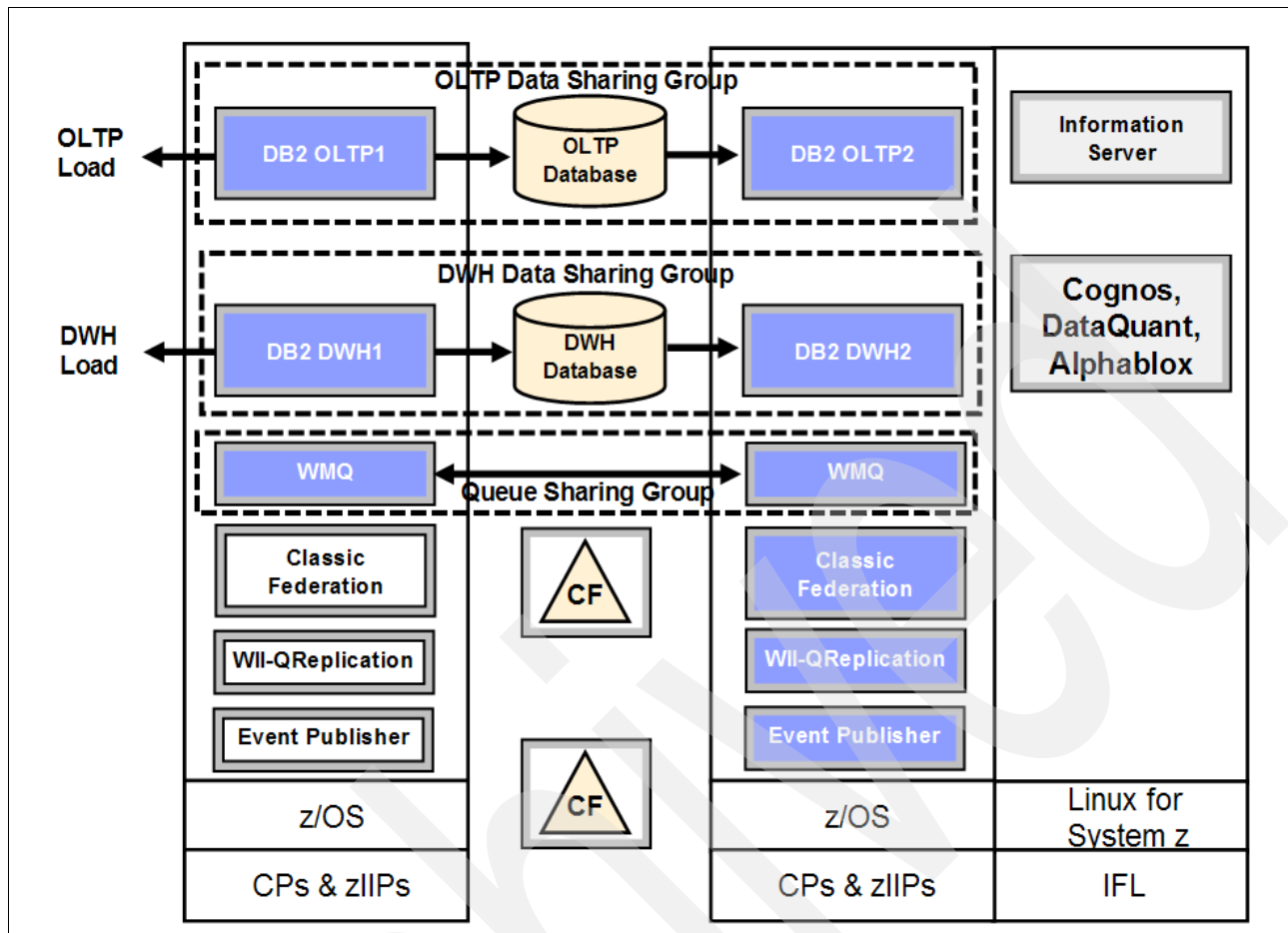


Figure 4-3 SYSPLEX with two LPARs and two data sharing groups

The Information Server uses WebSphere Application Server on Linux on System z to provide high availability of the WebSphere Application Server. Similar to the components we discussed earlier in this section, WebSphere Application Server can be configured in a cluster mode with multiple WebSphere Application Server profiles on two or more Linux on System z LPARs and proper high availability concepts for metadata hosted on Linux on System z.

For high availability of Cognos 8 BI, AlphaBlox, and DataQuant, you can configure them in standby mode on the other Linux for System z LPARs. The components used by ETL and reporting tools on Linux on System z can be configured in separate Linux images for a better isolation and workload management.

For details about high availability architectures of Linux on System z, refer to the “High Availability Architectures For Linux on IBM System z” white paper at the following address:

[http://www-03.ibm.com/servers/eserver/zseries/library/whitepapers/pdf/HA\\_Architectures\\_for\\_Linux\\_on\\_System\\_z.pdf](http://www-03.ibm.com/servers/eserver/zseries/library/whitepapers/pdf/HA_Architectures_for_Linux_on_System_z.pdf)

- Our test configuration: MONOPLEX, one LPAR, two stand-alone DB2 systems, ETL tools, one WebSphere MQ queue manager, and one Linux for System z LPAR

This configuration can be used to build a test environment where the benefits of sysplex and data sharing, such as scaling, rolling updates, and failover, are not required. This scenario is described in 6.2, “System configuration” on page 93.

- ▶ Additional configurations

Further configurations, for example, allow for a more strict separation between OLTP and data warehouse database resources, such as the use of separate LPARs for OLTP and data warehouse databases. The detailed evaluation of those different scenarios is not within the scope of this book.

Various combinations of different configurations are possible that are related to ETL and reporting tools on Linux on System z, for example, separate Linux LPAR for each component of ETL tools using z/VM® for running Linux for System z LPARs.

## 4.3 When DB2 for z/OS is a good fit

Having described the BI architecture on System z and some of the z/OS unique features to support this architecture, this question arises: When is DB2 for z/OS a good fit to implement the data warehouse?

When considering DB2 for z/OS to implement the warehouse, keep in mind the following criteria:

- ▶ The majority of source systems are on z/OS within IMS, VSAM, DB2, or sequential files. Or there is a requirement for tight integration with existing resources and systems on the System z platform.
- ▶ The data warehouse, data marts, or operational data store already exist on System z.
- ▶ Existing skills and investments are on the System z platform.
- ▶ The customer already maintains a System z-centric IT solution due to their favorable cost of ownership and comfort.
- ▶ The customer is implementing an operational BI application with embedded analytics within applications. These types of applications can leverage the System z transaction scalability capabilities.
- ▶ There is a requirement for a true real-time operational data store:
  - Operational data is already on the System z platform.
  - Data must be virtually in-sync with the operational data.
  - Availability, security, and resiliency requirements are high.
  - Auditable data warehoused requirements exist.
- ▶ Independent software vendor (ISV) packages, such as SAP® and PeopleSoft® on System z, offer both transactional (OLTP) and informational (warehouse and BI) systems. System z supports multiple images of SAP and other solutions, which simplifies support for these complex applications.
  - These packaged applications, which have tightly integrated components, have always made it desirable for the operational data and the warehouse to be housed in the same environment. Collocation reduces operational complexity, allowing for the reuse of skills and infrastructure in the form of processes, tools, and procedures.
  - There is a desire to consolidate distributed marts or data warehouses to an existing System z data serving platform. The customer may have spare System z capacity.
- ▶ The absolute best in reliability, availability, serviceability, security, and workload management is needed.

## 4.4 Business requirements revisited

We now revisit the business requirements listed in 4.1, “Business requirements for a data warehouse environment” on page 60, with the functions that are provided by the components described in 4.2, “The business intelligence architecture with System z” on page 61:

- Flexibility

Additional data sources can be incorporated with the help of the WebSphere Classic Federation, Federation Server, or data replication. To handle the additional load capacity, System z can be scaled up by adding more system resources or by configuring Parallel Sysplex.

- High data volumes

DB2 for z/OS with DS8000 disk subsystem is well suited to process databases in the range of terabytes.

- Near real-time data

Q replication provides replication of tables from the OLTP database to the data warehouse database with low latency.

- High availability/recovery

DB2 for z/OS, which is configured in data sharing mode, provides for system failover and continuous processing during system upgrades (rolling updates). When used in a Parallel Sysplex scenario, the ARM feature starts the failed components immediately on the same LPAR or the other LPAR (in case of an image failure).

- Query and reporting

Cognos 8 BI, AlphaBlox, and DataQuant/QMF offer a broad set of functions for analytics and reporting.

- Management

The System z platform and DB2 provide an excellent infrastructure for this requirement. System z hardware, z/OS operating system, and DB2 for z/OS are designed with reliability characteristics such as self monitoring, redundancy, self healing, and dynamic configuration and management.

- OLTP performance

The impact of data warehouse processing on OLTP performance can be controlled granularly with WLM on z/OS. One efficient way is to decrease the priority of queries depending on the elapsed execution time.

- Different data sources

With WebSphere Classic Federation and with the Federation Server, two products are available to access non-DB2 resources on z/OS and on various distributed systems.

- Security

The System z platform has proven to be an unmatched architecture for data security by fulfilling the highest industry security certifications. Features to support data encryption are built-in on a hardware level. The hardware supported encryption is used to ensure that backups are not readable if they fall into the wrong hands. With DB2 9 for z/OS, the network communication between the database and client can be Secure Sockets Layer (SSL)-encrypted based on a network trusted context.



## Part 2

# Design and implementation of our warehouse scenario

In this part, we describe in detail the implementation of the enterprise warehouse based on our fictitious application.

This part includes the following chapters:

- ▶ Chapter 5, “The business scenario and data models” on page 71
- ▶ Chapter 6, “The system environment” on page 91
- ▶ Chapter 7, “Functions in DB2 for z/OS for a data warehouse” on page 99
- ▶ Chapter 8, “Q replication and event publishing” on page 159
- ▶ Chapter 9, “Setting up ETL components for a data warehouse” on page 201
- ▶ Chapter 10, “Full load using DataStage” on page 257
- ▶ Chapter 11, “Incremental update with DataStage” on page 281
- ▶ Chapter 12, “An operational business intelligence implementation” on page 295
- ▶ Chapter 13, “Reporting and analysis with Cognos 8 BI” on page 305
- ▶ Chapter 14, “Reporting with DataQuant, QMF, and AlphaBlox” on page 397

Archived

## The business scenario and data models

In this chapter, we introduce the fictitious organization and business requirements that we use to demonstrate the usefulness of an enterprise business intelligence (BI) solution and data warehouse on DB2 for z/OS. The scenario, some components of the chosen data model, and business-oriented queries used throughout our implementation are based on the TPC Benchmark H Standard Specification Revision 2.5.0 documentation printed 10 July 2006.

We use the components of the TPC-H benchmark because they provide a realistic data model and a set of business-oriented ad hoc queries that demonstrate possible querying and analysis tasks in important areas that an organization may require. The generation of new source information into the transaction systems has been implemented by using a custom designed Web page to simulate operational transactions.

The scenario provides background information about why we implemented particular data warehouse functionality within DB2 for z/OS and other components of the end-to-end solution such as Information Server and Cognos 8 BI. We also provide a summary of our conceptual implementation, the products installed, and the DB2 capabilities that we demonstrate in our scenario.

This chapter contains the following sections:

- ▶ 5.1, “Background information” on page 72
- ▶ 5.2, “Business requirements” on page 72
- ▶ 5.3, “Solution overview” on page 75
- ▶ 5.4, “The transactional data model” on page 78
- ▶ 5.5, “The operational and dimensional data model” on page 85
- ▶ 5.6, “Referential integrity for a data warehouse” on page 87
- ▶ 5.7, “Data modeling options: IBM Industry Models” on page 88

## 5.1 Background information

Our fictitious organization, called *RedParts Distribution*, is a company that works within the metal product wholesale and distribution industry. RedParts receives metal parts from suppliers and distributes these parts to their customers worldwide in the hope of making a profit. Customers are usually retail business who then sell the parts mostly through retail stores. At various times, RedParts runs promotions for their customers depending upon the home country and economic region to which they belong. It is also common to offer discounts between 1% and 9% to customers depending upon the volumes of parts ordered in the past.

Orders are placed by customers over the Internet or are initiated at one of their many physical branches. Branches are spread throughout the world and can be classified as small, medium, or large depending upon the number of orders that they take during the previous quarter. After an order is received and verified, it is packed by a clerk and eventually shipped to the customer. The order fulfillment is complete when the customer receives the requested parts.

On occasion, parts are returned by the customer to RedParts. At the time of the return, the reason for the return is recorded as damaged parts, unwanted parts, or no reason provided. RedParts measures the return rate for each part because it provides a good basis to determine whether there are potential problems with particular suppliers, the parts themselves do not perform, or the parts are not reliable and should not be distributed. On a monthly basis, these return rates are made available to customers on the Internet if they require them. The return rates are shown by part, brand, and manufacturer.

RedParts manages the distribution and tracking of all orders from their warehouse in Germany. Analysis and reporting of order information is extremely difficult because this information is not located in one single place, and information from the branch ordering system is not regularly available at the time reporting needs to occur.

Table 5-1 lists the sources of information that RedParts currently uses to fulfill their operational and strategic reporting requirements.

Table 5-1 Scenario data sources

Application	Database	Description
OLTP_B	Branch Ordering System Relational database	This database contains order information for orders initiated at the physical branches.
OLTP_W	Web Ordering System Relational database	This database contains order information for orders initiated over the Internet.
OLTP	RedParts Reference System Relational database	This database contains details about customers, suppliers, and parts for all orders initiated at branches or over the Internet.
Additional product and supplier external data	Flat File format	Product and supplier extract files are used to demonstrate the value of WebSphere Classic Federation Server. In our scenario, this external data contains updated part and price information and must be merged with the existing part information held within the reference system (online transaction process (OLTP))

## 5.2 Business requirements

RedParts Distribution wants to improve their operational reporting capabilities and strategic reporting and analysis capabilities. They realize that, to do this, their corporate information must be consolidated into a single environment, such as a data warehouse. In addition they



have had problems in the past due to the time it takes to receive updates on order status and details from their current segregated reporting implementation.

With the help of some external consultants, they put together the following requirements:

- ▶ Single integrated repository of order information for both Internet and branch orders is required.
- ▶ Historical order information is to be kept permanently at this stage for ad hoc querying and reporting. It is expected that the central fact data table within the reporting environment, which will be used to hold detailed order information, will grow significantly due to the number of parts being ordered worldwide every day.
- ▶ Operational and recent order information must be kept for a rolling 12-month period.
- ▶ Operational data must be updated in near real time to ensure accurate reporting of costs and revenue.
- ▶ Reporting and querying must be continuously available 24 hours a day, 7 days a week, for multiple users.
- ▶ Strategic analysis of selected data must be available from a business intelligence portal, with some drill-up-and-down and slice-and-dice functionality.
- ▶ Near real-time information is required in some cases for order and line item information. A wholesaler cannot wait for a nightly batch extract, transform, and load (ETL) refresh from OLTP systems.
- ▶ A change capture of history is required for customer information. For example, if a customer has a change of address, then historical reporting should show the address used at the time of an order and not necessarily the customer's current address.
- ▶ An application with embedded information is required for processing and taking orders from customers. This application must contain data that is derived from both the operational and dimensional data store of our data warehouse. Relevant information that can be of use to the operator at the time must be displayed, which may impact decisions such as the level of service or amount of discount that are offered.

Reports and queries are required within the following subject areas:

- ▶ Pricing and promotions
- ▶ Supply and demand
- ▶ Profit and revenue
- ▶ Customer satisfaction
- ▶ Shipping management

Table 5-2 shows specific deliverables that are identified by the wholesaler as examples to implement in our scenario. Along with the deliverable, we give an indication where, within the end-to-end solution, this has been implemented.

Table 5-2 Sample scenario deliverables

#	Deliverable	Description	Deliverable details
1	Top volume customer query	The Top volume customer query retrieves a list of customers based on them having placed a large quantity order. <i>Large quantity orders</i> are defined as those orders whose total quantity is above a certain level. The query lists the customer name, nation, order key, date, total price, and quantity for the order. Total price is sorted in descending order.	Query  Used to demonstrate ad hoc querying capability using Cognos 8 BI.  The implementation is in 13.5, "Ad hoc queries with Query Studio" on page 374.

#	Deliverable	Description	Deliverable details
2	Top four customers per quarter	This query lists the top four customers for each financial quarter in a given financial year.	<p>Query, report, and Web application</p> <p>Used to demonstrate DB2 functionality as listed in Table 5-3 on page 77, a DataQuant report, and an AlphaBlox Web application.</p> <p>Implementation is in the following sections:</p> <ul style="list-style-type: none"> <li>▶ 7.11, “OLAP functions” on page 154</li> <li>▶ 14.1.3, “A small DataQuant report for RedParts Distribution” on page 400</li> <li>▶ 14.3.3, “A small AlphaBlox Web application for RedParts Distribution” on page 412</li> </ul>
3	Part return rate analysis	This query performs analysis and explores customer “part return rates” and determines the highest return rates by manufacturer and customer nation. Return rates are calculated by using the number of parts ordered and the number of parts returned. We also want to analyze return rates by customer promotion campaigns, part packaging, and container types	<p>Online analytical processing (OLAP) analysis</p> <p>Used to demonstrate OLAP multidimensional analysis capability within Cognos 8 BI.</p> <p>Implementation is in 13.6, “Multidimensional analysis with Analysis Studio” on page 385.</p>
4	Order priority checking query	This query determines how well the order priority system is working and gives an assessment of customer satisfaction. It provides a count of the number of orders that are received in a given quarter of a given year in which at least one line item was received by the customer later than its committed date.	<p>Query</p> <p>Used to demonstrate DB2 functionality as listed in Table 5-3 on page 77.</p> <p>Implementation is in Example 7-18 on page 113.</p>
5	Line-item pricing summary  Customer account balance	<p>The line-item pricing summary query provides a summary pricing list for all line items shipped as of a given time period. The query lists the line item and the charge amount using the following calculation: L_EXTENDEDPRICE* (1-L_DISCOUNT)*(1+L_TAX)</p> <p>The customer account balance is a simple query that is used to show the current account balance for a customer. It is planned that this query can be embedded in an order processing application. The query is required to match on a customer name and the account balance multiplied by 1,000.</p>	<p>Query</p> <p>Used to demonstrate DB2 functionality as listed in Table 5-3 on page 77.</p> <p>Implementation is in 7.7, “Index on expressions” on page 139.</p>
6	Total product ordered for a branch and time period	This query identifies the total quantity product that is ordered through a selected branch for all customers in a given region and time period.	<p>Query</p> <p>Used to demonstrate DB2 functionality as listed in Table 5-3 on page 77.</p> <p>Implementation is found in Example 7-36 on page 126.</p>

#	Deliverable	Description	Deliverable details
7	Order count query	This query counts the orders by the customer key and the order date.	<p>Query</p> <p>Used to demonstrate DB2 functionality as listed in Table 5-3 on page 77.</p> <p>Implementation is in 7.10.2, "MQTs used in our scenario" on page 153.</p>
8	Returned items report	<p>This reports lists the top 20 customers, in terms of their effect on lost revenue for a given quarter, who have returned parts. The report lists the customer's name, address, nation, phone number, account balance, comment information, and revenue lost. The customers are listed in descending order of lost revenue. Revenue lost is defined as the following calculation for qualifying line items:</p> $\text{SUM}(\text{L\_EXTENDEDPRICE} * (1 - \text{L\_DISCOUNT}))$	<p>Report</p> <p>Used to demonstrate reporting capability using Cognos 8 BI.</p> <p>Implementation is in 13.4, "Reports with Report Studio" on page 367.</p>
9	Order processing Web application	<p>This implementation is an example of an application with embedded analytical information, derived from our operational and dimensional data store of our data warehouse.</p> <p>As part of the implementation, we use a materialized query table (MQT) to improve query response times.</p>	<p>Used to demonstrate operational BI and DB2 functionality as listed in Table 5-3 on page 77.</p> <p>Implementation is in 12.1, "OLTP application with embedded analytics" on page 296.</p>

For the DB2 functionality that we selected to demonstrate in our scenario, refer to Table 5-3 on page 77.

## 5.3 Solution overview

The solution that has been proposed to RedParts Distribution is based on demonstrating the following criteria:

- ▶ Operational reporting from a traditional operational data store (ODS)
- ▶ Near real-time reporting within a data warehouse environment
- ▶ Benefits of using DB2 for z/OS for a data warehouse environment
- ▶ Best-of-breed reporting solutions available such as Cognos 8 BI and AlphaBlox
- ▶ Both traditional ETL and near real-time data extraction options

The solution includes the following technology options:

- ▶ DB2 for z/OS as data warehouse environment
- ▶ InfoSphere DataStage enterprise edition for ETL parallel jobs
- ▶ Event Publisher and DataStage to propagate data changes
- ▶ WebSphere Information Server – Q Replication to demonstrate low latency replication
- ▶ Cognos 8.3 BI for querying and reporting

**Note:** IBM has announced that Cognos 8 BI for Linux on System z will be available in the second half of 2008. For more information, refer to the following Web address:

<http://www.ibm.com/software/data/info/new-systemz-software/>

- ▶ AlphaBlox
- ▶ WebSphere Classic Federation Server to read the monthly part and supplier sequential files

This simple example has been chosen to demonstrate the value of Federation Server. However, in reality, the data exists in a different data source format other than sequential file.

Figure 5-1 shows a conceptual overview of our data warehouse environment.

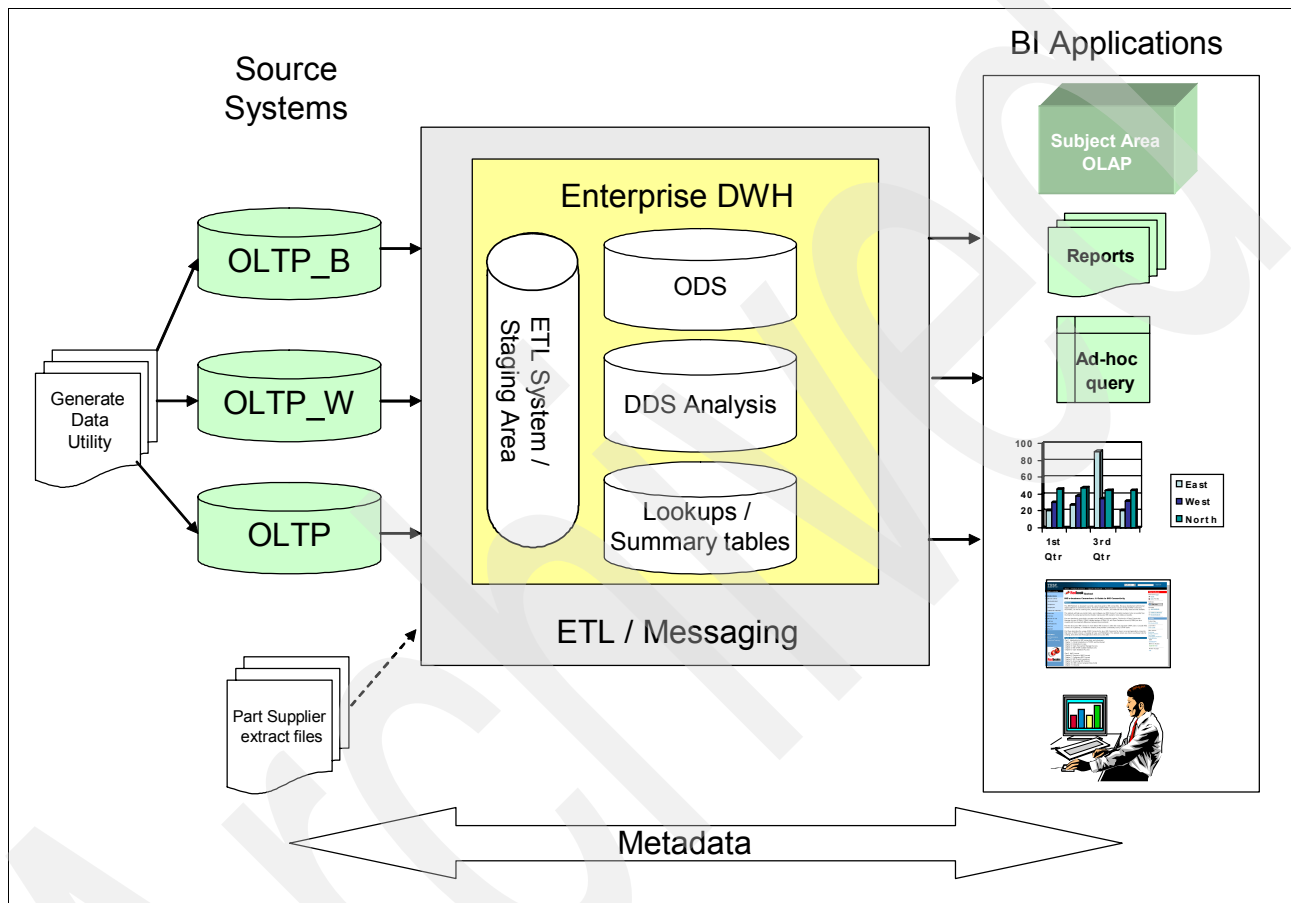


Figure 5-1 Conceptual overview for scenario

This is not a complete conceptual overview of what a data warehouse environment should include. It has been simplified for the purposes of our scenario and the book to demonstrate some of the concepts.

We have shown in the diagram that metadata management should occur throughout a data warehouse implementation, because we believe this to be an important component within any solution. For our scenario, any data staging or ETL system functionality, such as data cleansing, and conforming has been implemented within DataStage.

The data model definitions for our environment are described in 5.4.1, "The OLTP database model" on page 78, and 5.5, "The operational and dimensional data model" on page 85.

Our simulated enterprise data warehouse environment contains a number of components. For definitions and discussion about these and other components, see Chapter 1, "Definitions" on page 3.

The overall implementation is based on the scenario that source data from OLTP systems is being loaded into an ODS for integrated operational reporting. We also demonstrate that optionally some part and supplier information can be loaded from extract files by using WebSphere Classic Federation Server. Our ODS is based on the TPC-H data model. The ODS allows RedParts to perform their operational reporting needs, such as ensuring that business intelligence reporting covers both orders initiated at branches and those orders made over the Internet. Selected information within the ODS is updated in near real time from the respective OLTP systems, providing more value to their BI applications.

The data warehouse environment also includes a dimensional star schema model to demonstrate an area within the solution that is optimized for reporting and analysis. In a real scenario, this dimensional model may be a data mart that contains a subset of information from the data warehouse based on a subject area. For simplicity and to provide an example, we include a single “order transaction” subject area and refer to this as the *dimensional data store* (DDS).

Lookup tables are common within a data warehouse environment. We simulate this in our scenario by using some of the existing TPC-H tables, such as Nation and Region, and refer to them as *lookups*. We have not specifically implemented any summary tables, but note that a true data warehouse environment may include these. As part of our example deliverables, we implement summary MQTs to satisfy further reporting requirements.

Table 5-3 shows the DB2 for z/OS functionality that we implement in our scenario. If the functionality is demonstrated in one of the deliverables that we are producing, the numbers in the second column refer to the deliverable number in Table 5-2 on page 73.

Table 5-3 Demonstrated DB2 functionality

DB2 functionality	Demonstrated where
Universal table space - Partitioned by growth	The example dimensional data model includes partition by growth on the transaction fact table to allow DB2 to handle space management due to the high volume of orders received.
Universal table space - Partitioned by range	The ODS data model had “partitioned by range” applied to the Part table based on particular parts coming from particular suppliers. This is for query performance when queried on a part range or particular part value.
Materialized query tables	Deliverable 7, Order count query Deliverable 9, Order processing Web application
Index compression	Deliverable 4, Order priority checking query
Star join	Deliverable 6, Total product ordered for a branch
Index on expression	Deliverable 5, Line-item pricing summary and customer account balance  An index has been successfully placed on the following expressions: (L_EXTENDEDPRI*(1-L_DISCOUNT)*(1+L_TAX))  (ACCTBAL*1000)
DB2 OLAP functions	Deliverable 2, Top four customers per quarter The RANK and ROW_NUMBER functions are used to identify the top four customers per quarter.

## 5.4 The transactional data model

For the implementation in this book, we assume an existing transactional system with data stored in a dedicated DB2 for z/OS subsystem and (optionally) in a legacy non-relational data source. The data model describes an order processing schema, but we do not make assumptions that are specific to a certain industry.

Furthermore, to demonstrate connections to legacy data sources, we assume that some data resides in flat files on z/OS and is maintained by a mainframe application. This data is considered to be part of the transactional environment as well.

In real life, the transactional system continuously creates and updates data in its database schema. In our example, orders are created, and customer data is updated. A J2EE™ application is used to populate this data structures and simulate this kind of data update.

### 5.4.1 The OLTP database model

The transactional (OLTP) data model for our fictitious company includes three parts that are implemented by tables in three different database schemas: OLTP\_B, OLTP\_W, and OLTP.

The company is assumed to accept orders through two channels:

- ▶ One channel handles branch offices where orders are typed manually into a system
- ▶ One channel handles Internet (Web) application orders in which customers submit their order directly to the system

The orders for both channels are maintained by different applications and stored in different database schemas: OLTP\_B for orders from branch offices and OLTP\_W for orders from the Web. These two sets of database tables can conceptually also be the result of a company merger or an acquisition where two existing order management systems are maintained with a different set of orders.

For simplicity, in our scenario, both the tables in schema OLTP\_B and OLTP\_W are identical and modeled as outlined in Figure 5-2. They include tables for order, shipment, and line-item data. The structure of the table is derived from the tables that are used for the TPC-H benchmark, but have slight modifications that require additional transformation logic before they can be used in the context of the data warehouse.

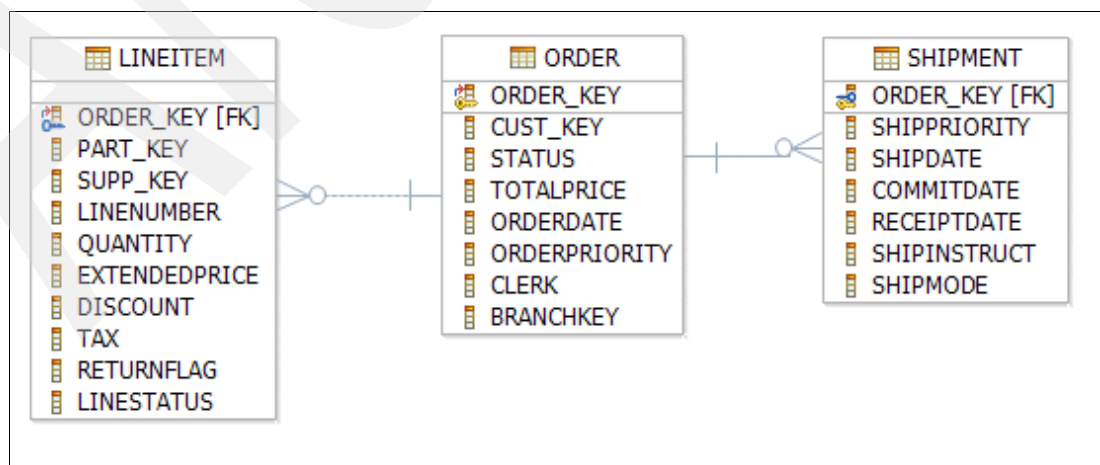


Figure 5-2 Transactional schema (OLTP\_B and OLTP\_W)

An *order record* always refers to one shipment record and has one or more line-item records. The *line items* refer to parts and include quantity and pricing information along with discount and tax. The data in these tables and the relationship between them is consistent, although some fields contain random values.

The tables in schema OLTP in Figure 5-3 are used to maintain data that is common for both operational systems, such as customers, parts, suppliers, and data in related tables. The assumption here is that the company runs an additional application to maintain parts, suppliers, and customers (maybe through a master data management system). The supplier's nation relates to the nation and region tables through the nationkey. However, the customer table has only the country in plain text. Parts and suppliers have an *n:m* relationship. Therefore, an additional *partsupp* table is used to model this relationship.

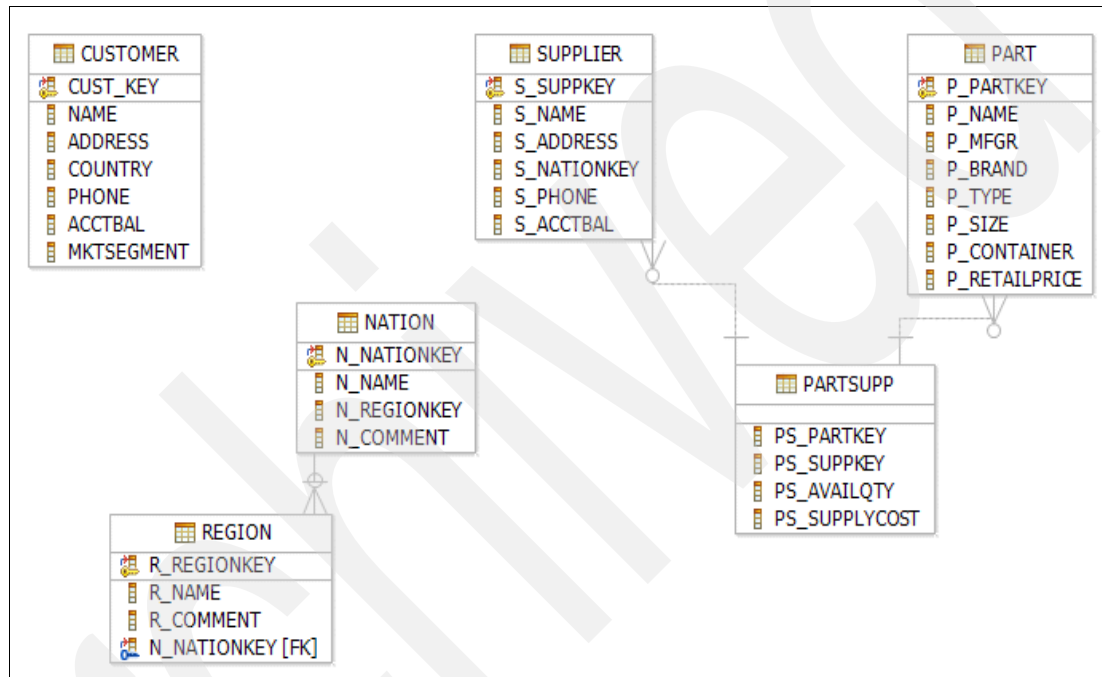


Figure 5-3 Transactional schema (OLTP) based on the TPC-H data model

Figure 5-4 shows the relationship between the three mentioned schemas. The order processing systems use a set of lineitem, order, and shipment tables (at the left and right in the figure). Both share part, partsupp, and supplier (in the middle) as well as customer, nation, and region information.

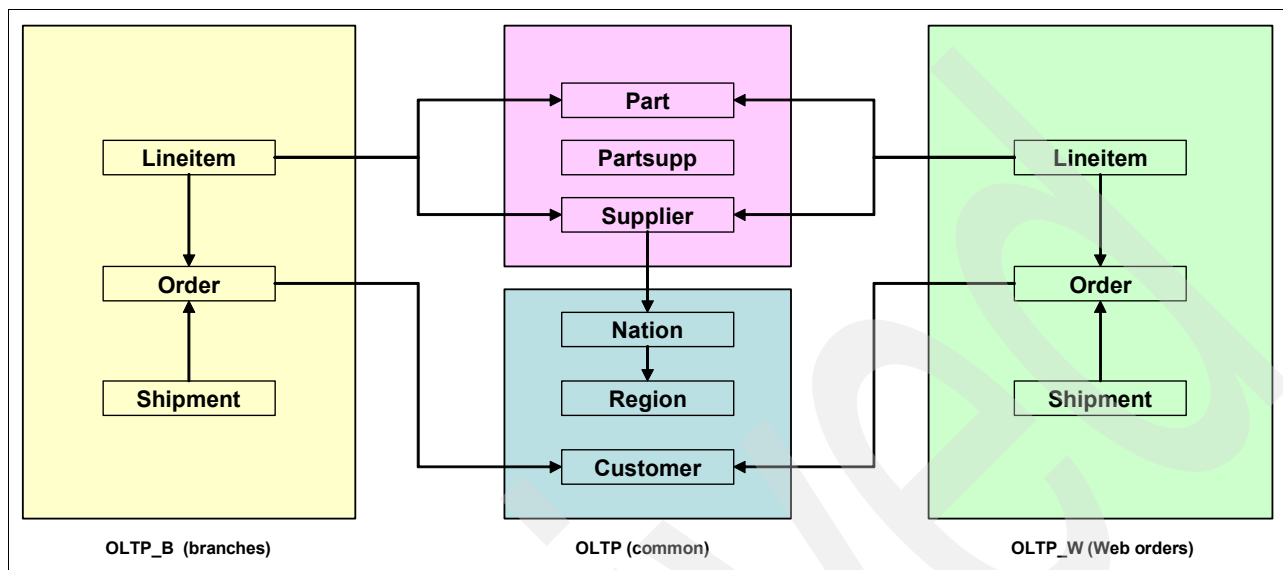


Figure 5-4 The transactional data model

## 5.4.2 Creating the schema for the OLTP model

To create the table spaces for our transactional environment, we use Data Definition Language (DDL) statements such as in Example 5-1. For concurrent updates to rows on a page, LOCKSIZE ROW is specified. To accommodate different languages, we create the table space with Unicode CCSID.

### Example 5-1 Table space DDL

```
CREATE TABLESPACE TSCUSTOM IN DBOLTP
  USING STOGROUP SGOLTP
  COMPRESS YES
  BUFFERPOOL BP2
  LOCKSIZE ROW
  CCSID UNICODE;
```

The DDL in Example 5-2 shows how we create the tables for our transactional environment. We use generated identity columns to create unique keys for the tables. Another option is to leverage the sequences concept in DB2 z/OS.

### Example 5-2 Table DDL

```
CREATE TABLE OLTP.CUSTOMER
(
  CUST_KEY INTEGER NOT NULL GENERATED ALWAYS
    AS IDENTITY(START WITH 1001, CACHE 100),
  NAME VARCHAR(25),
  ADDRESS VARCHAR(40),
  COUNTRY CHAR(25),
  PHONE CHAR(15),
  ACCTBAL DECIMAL(12,2),
```



```

MKTSEGMENT CHAR(10),
PRIMARY KEY (CUST_KEY)
)
IN DBOLTP.TSCUSTOM;

```

---

You can find the complete set of DDL statements for creation of the transactional schema in Appendix B, “Schema definitions” on page 431.

### 5.4.3 Data from legacy data sources

For the scenario in the book, we assume that the parts, suppliers, and part suppliers tables are managed by a separate application, using a non-relational legacy data source, such as IMS or VSAM files. For demonstration purposes, and due to resource constraints, we use a flat sequential file for parts, suppliers, and part suppliers data.

The data in this legacy data source, which is in the form of sequential files, is made available through WebSphere Classic Federation simulated as a relational database management system (RDBMS). To understand the implementation of WebSphere Classic Federation and mapping sequential file tables using WebSphere Classic Federation and WebSphere Classic Data Architect, refer to 9.4, “Setting up WebSphere Classic Federation” on page 210.

Refer to “PART data” on page 81, “SUPPLIER data” on page 82, and “PART SUPPLIER data” on page 83 for details about the data format and sample data in the sequential files.

#### PART data

In our scenario PART data is in a sequential file named GAURAN.WCF.PART. Example 5-3 shows the sample data in the sequential file.

*Example 5-3 PART data*

---

1	goldenrod lace spring peru powder	Ma
2	blush rosy metallic lemon navajo	Ma
3	dark green antique puff wheat	Ma
4	chocolate metallic smoke ghost drab	Ma
5	forest blush chiffon thistle chocolate	Ma
6	white ivory azure firebrick black	Ma
7	blue blanched tan indian olive	Ma
8	ivory khaki cream midnight rosy	Ma
9	thistle rose moccasin light floral	Ma
10	floral moccasin royal powder burnished	Ma
11	chocolate turquoise sandy snow misty	Ma
12	peru ivory olive powder frosted	Ma
13	ghost blue olive sky gainsboro	Ma
14	linen seashell burnished blue gainsboro	Ma

---

**Note:** The data in Example 5-3 shows only three columns of the data. The complete data record is of 176 characters.

Table 5-4 describes the format of the PART table record.

Table 5-4 PART table format

Column name	Offset	Length
PARTKEY	2	9
NAME	13	
MFGR	70	
BRAND	97	
TYPE	109	
SIZE	138	
CONTAINER	149	
KEY_SRC	164	

## SUPPLIER data

In our scenario, SUPPLIER data is in sequential file named GAURAN.WCF.SUPPLIER. Example 5-4 shows the sample data in the sequential file.

Example 5-4 Supplier data

2	Supplier#000000002	89eJ5ksX3ImxJQBvx0bC
3	Supplier#000000003	q1,G3Pj60jIuUYfUoH18BFTKP5aU9bEV
4	Supplier#000000004	Bk7ah4CK8SYQTepEmvMkkgMw
5	Supplier#000000005	Gcdm2rJRz15q1TVz
6	Supplier#000000006	tQxuVm7s7Cn
7	Supplier#000000007	s,4TicNGB4u06PaSqNBUq
8	Supplier#000000008	9Sq4bBH2FQEmaF0ocY45sRTxo6yuoG
9	Supplier#000000009	1KhUgZegwM3ua7dsYmekYBs
10	Supplier#000000010	Saygah3gYWMp72i PY
11	Supplier#000000011	JfwTs,LZrV, M,9
12	Supplier#000000012	aLIW q0HY
13	Supplier#000000013	HK71HQyWoqRW0X8GI FpgAifw,2PoH
14	Supplier#000000014	EXsn05pTNj4iZR
15	Supplier#000000015	o1XVbNBfVzRqgokr1T,I

**Note:** The data in Example 5-4 shows only three columns of the data. The complete data record is 126 characters.

Table 5-5 shows the format of the SUPPLIER table record.

Table 5-5 Supplier table format

Column name	Offset	Length
SUPPKEY	2	9
NAME	13	25
ADDRESS_TEXT	40	40
NATIONKEY	84	9
PHONE	95	15
ACCTBAL	114	12

### PART SUPPLIER data

In our scenario, PART SUPPLIER data is in sequential file named GAURAN.WCF.PARTSUPP. Example 5-5 shows the sample data in the sequential file.

Example 5-5 PART SUPPLIER data

1	2	3325	771.64
1	2500002	8076	993.49
1	5000002	3956	337.09
1	7500002	4069	357.84
2	3	8895	378.49
2	2500003	4969	915.27
2	5000003	8539	438.37
2	7500003	3025	306.39
3	4	4651	920.92
3	2500004	4093	498.13
3	5000004	3917	645.40
3	7500004	9942	191.92
4	5	1339	113.97
4	2500005	6377	591.18

Table 5-6 shows the format of the PART SUPPLIER table record.

Table 5-6 PART SUPPLIER table format

Column name	Offset	Length
PARTKEY	2	9
SUPPKEY	15	9
AVAILQTY	28	9
SUPPLYCOST	41	12

## 5.4.4 Simulating the transactional environment

To populate the tables of the transactional data model, we use a J2EE application running on WebSphere Application Server. This application (Figure 5-5) accesses DB2 for z/OS by using the Universal JDBC driver (type 4).

**Data Warehouse on System z -- Redbook OLTP schema generation tool**

Database content

Refresh

Transactional Content	OLTP_B	OLTP_W
Order	100	100
Shipment	100	100
Lineitem	200	200

Common Content	OLTP
Customer	201
Parts	250
Suppliers	1000
Parts-Suppliers	1000

Start generation for customer or order records

Target for Orders

☒ OLTP\_B (branches)  
☐ OLTP\_W (web)

Specified if the new rows should be created in schema OLTP\_B or OLTP\_W

New orders

0

Number of new orders to create.  
An order also has a shipment associated with it.

Year and month for new orders

20084

Date for the orders that are created  
Day will be created randomly and ship dates derived from the generated order date.

Items per order

0

Each order can have multiple items - this number defines how many items should be created per order.

New customers

0

Number of new customer records to create.

Updated customer

0

Number of customer records to update.  
A customer will be picked randomly and its country setting will be changed.

Statements per transaction

1

User transactions are used to group multiple statements into transactions. With fewer statements per transactions, processing takes longer. With more statements per transactions, timeouts or lock issues may occur.

Start Generator

Figure 5-5 Web application simulating OLTP workload for the transactional environment

The application simulates a transactional system and offers the following options:

- Overview of records in the OLTP database tables
- Creation of new orders

This option creates new rows in the order and shipment tables. Depending the number of items to create per order, additional rows per order are created in the lineitem table as well. Before creating entries in the tables, the set of available part and supplier keys are retrieved from the partsupplier table. To demonstrate reporting based on order date, you can use the application to specify a year and month for which to create the order. Additional dates, such as shipdate, are derived from the order date by adding a random number of days.

For generation, the application requests the desired target schema (OLTP\_B or OLTP\_W) for which the new records should be created.

- Creation of new customer records

All the columns for the OLTP.CUSTOMER table are generated with random values. Before creating entries in the customer table, the set of available nations is read from the name column of the nation table. One of these nation names are randomly selected and stored in the country column of the customer table.

- Update of customer records

To simulate updates to existing records in the OLTP system, a number of customer records are selected based on the cust\_key column. Then the application randomly assigns a different country name to the customer and performs an update statement on the customer table.

The J2EE applications takes advantage of user transactions, managed in WebSphere Application Server, to group SQL statements, queries, and record creation in transactions. This reduces the number of locks that are held during processing and allows for concurrent processing on the operational data tables.

## 5.5 The operational and dimensional data model

The purpose of the operational data model is to provide a consolidated view for the various data sources described in 5.4, “The transactional data model” on page 78. The model in Figure 5-6 on page 85 is defined in a single database schema, DWODS, and is derived from the TPC-H data model.

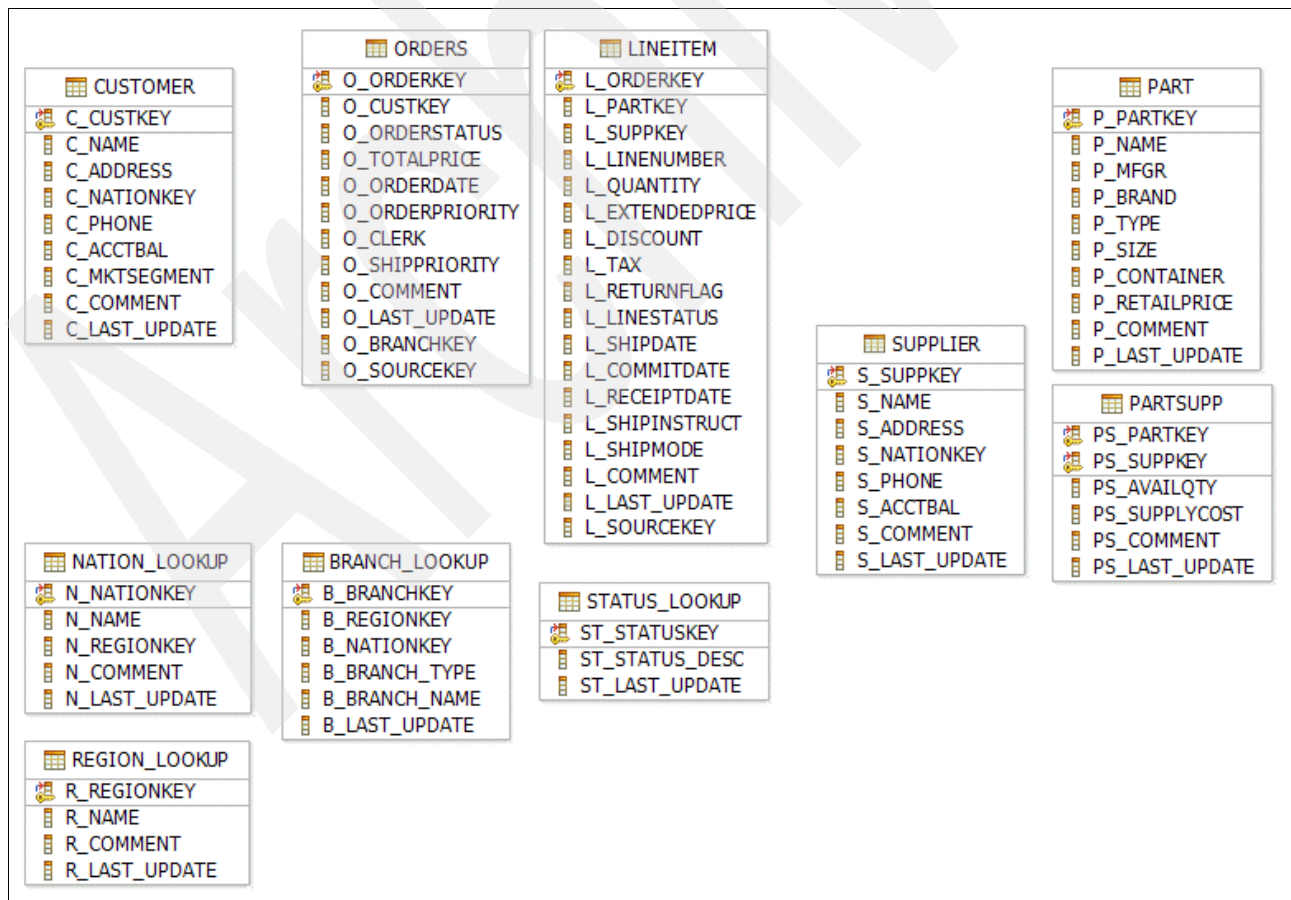


Figure 5-6 Model for the operational data store, based on TPC-H

This model differs from our OLTP model in that it has no dedicated shipment table, but rather this information is included in ORDERS and LINEITEM. Some fields, such as CUSTOMER.C\_NATIONKEY, also reference a lookup table, nation\_lookup, which is not the case in the OLTP model.

We do not define referential integrity constraints on the operational data model to ease loading data from the OLTP system. We assume that consistency of the data is ensured by the corresponding ETL process. For a description of referential integrity, see 5.6, “Referential integrity for a data warehouse” on page 87.

The operational data model also uses the same (primary) keys as the OLTP model. In a real scenario, this approach can cause conflicts if multiple source systems with overlapping keys are consolidated. In our scenario, the keys in the different OLTP sources are already guaranteed to be unique. Therefore no new keys have to be introduced.

We also create a dimensional data model (see Figure 5-7) in schema DWHDDS, which is a star schema with an order transaction fact table (ORDER\_TRANSACTION\_FACT) and dimensions for order status, customer, branch, supplier, and part. It also includes a date dimension.

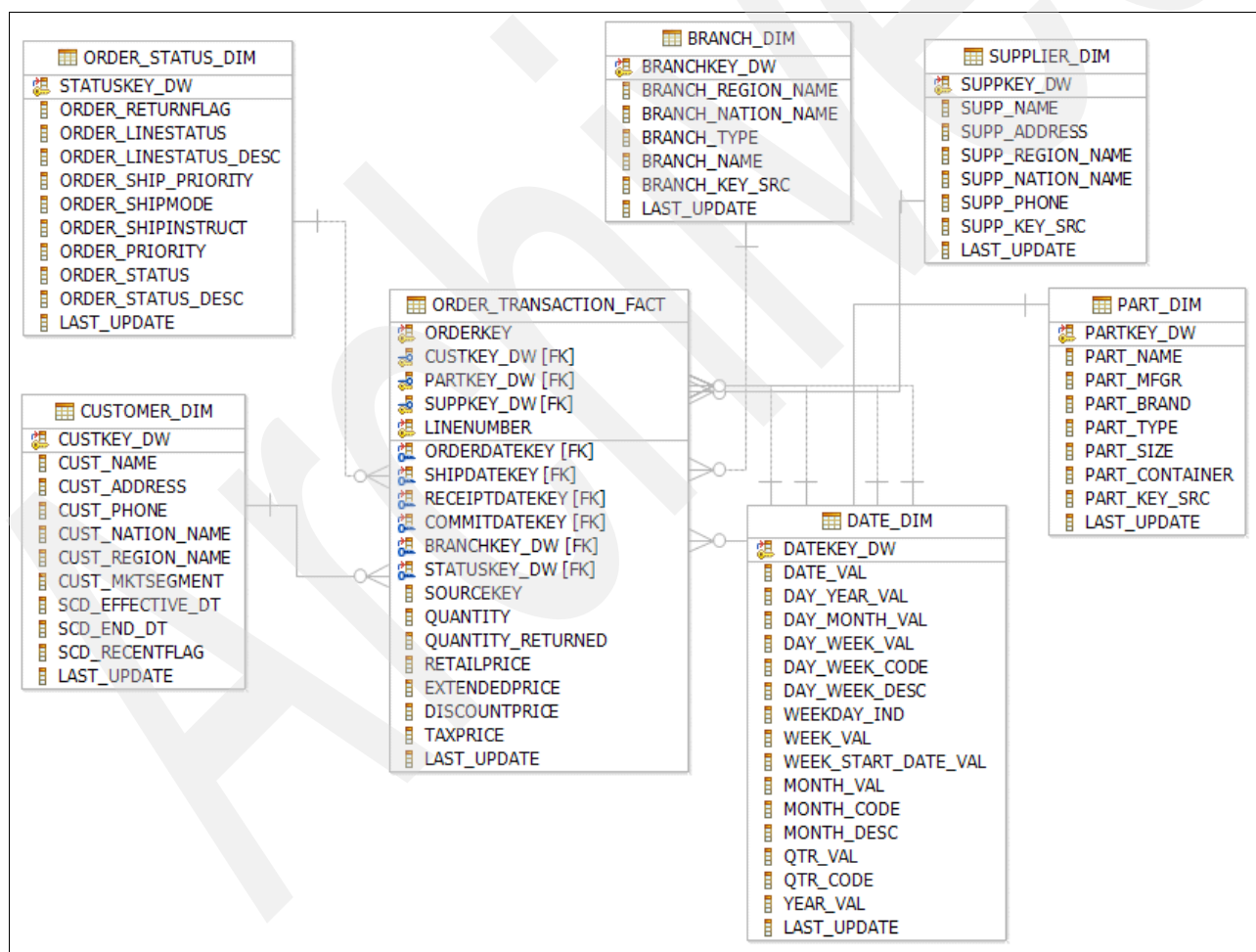


Figure 5-7 Dimensional data model

This dimensional data model is also used later to define cubes for analysis and reporting.

## 5.6 Referential integrity for a data warehouse

Referential integrity is used to ensure the legitimate relationships between rows of one table with the rows of another related table. In a data warehouse environment, usually the following relationships must be maintained:

- ▶ Every foreign key in the fact table must have the associated primary key in a dimension table.
- ▶ Every primary key in a dimension table does not need to be associated with any foreign key of fact table.

For example, in a data warehouse database, a fact table row that has customer ID value should have the same corresponding customer ID value in one of the rows of the customer dimension table.

A fact record that violates the referential integrity can be dangerous because it appears to be stored incorrectly. The relationship can be easily destroyed if we load a fact table with a record that does not have the corresponding key value in the dimension table, or we delete a row from a dimension table that still has a reference in the fact table. This can lead to unpredictable results for a BI query. Therefore, it is always advisable to maintain a legitimate relationship between the fact table, dimension table, or other related tables by using an automated mechanism in a data warehouse environment. The relationships can be maintained using the following methods:

- ▶ Referential integrity in the database

The primary key and foreign key concept of RDBMS can be used to implement this kind of relationship in the database where a foreign key column in a fact table refers to one of the primary keys in the dimension table. Enforcing referential integrity in a database always guarantees clean data, but it also has disadvantages with it as in the following examples:

- Bulk load

In a data warehouse environment, millions of rows of data of a size equivalent to several GBs are inserted in one load. The load has a performance impact when done in a referential integrity enforced database. Every row that is inserted is tested for referential integrity to check for the foreign key in the parent table before it is inserted.

- Administration

Enforcing referential integrity in a database also leads to manageability problems. For example, if one row in a dimension table is deleted, it either puts in check pending state the fact table, which may be several millions of rows in size, or it deletes thousands of rows from the fact table that refer to the corresponding foreign key.

- Authenticity of source data

Most of the data in data warehouse is loaded from OLTP database where in check pending state is always enforced. Therefore, in certain cases, if a data warehouse database is loaded in a right sequence from OLTP, the data in data warehouse database always has the relationship maintained. Enforcing referential integrity on a database that is always loaded with clean data is an overhead.

- Using the ETL tool

Another way to maintain data integrity is to check or prepare data just before loading the data and not enforcing the referential integrity in the database. This process can be implemented by doing the data check in the ETL tool before loading the data in the database. The ETL tool can use surrogate keys or any other method to check for the referential integrity before loading the data. Refer to 10.4.2, “Surrogate key stage utilization” on page 267, for details about using the ETL tool in our scenario for implementing referential integrity.

- Sanity checks

Referential integrity can be checked in a database after the database change or load to find and resolve the violations. Finding and resolving the violations on a very large fact table can be slow and requires manual intervention after every load. A referential integrity check can be limited to the recently loaded data considering that the previous data has all the relationships in place.

**Note:** The decision about which methods to use to implement referential integrity depends on the scenario and the environment. In our scenario, we used DataStage to check the referential integrity relationships.

## 5.7 Data modeling options: IBM Industry Models

For the scenario in this book, we use a rather simplistic model that is easy to explain and allows us to demonstrate and implement the concepts for data warehouse on the System z platform. In a real scenario, the data models for an operational and dimensional data store and the entire setup of a data warehouse will most likely become more complex.

Business users, business analysts, and data modelers typically work closely together in a company to create the desired models that best reflect the existing entities and that support the required reporting and analysis. While all companies are different, the task of defining the right models can be considerably shortened by using pre-built, industry-specific models. These models can then be further customized to accommodate individual requirements.

In addition to data models for reporting and analysis, the IBM Industry Models contain key performance indicators, definitions for data marts, and processes. In times where compliance requirements and governance play an increasing role, the experience and effort that went into defining the industry models can considerably reduce costs and project risk when implementing a data warehouse solution.

The following IBM Industry Models are available today:

- IBM Financial Markets Industry Models
  - Brokerage
  - Treasury services
- IBM Banking Industry Models
  - Retail
  - Commercial
  - Card Services
  - Financing



- ▶ IBM Insurance Industry Models
  - Property and Casualty
  - Life and Pensions
  - Personal or Commercial
  - Group or Individual
  - Health
- ▶ IBM Retail Data Warehouse
  - Retailers (direct, indirect, drug stores)
  - Manufacturers
  - Distribution
- ▶ IBM Telecommunications Data Warehouse
  - Telephone companies (local exchange carriers, cell phone)
  - Long distance carriers
  - Cable or satellite operators
  - Internet service providers

The underlying SQL definitions can easily be applied and customized for an existing DB2 for z/OS environment. The IBM industry models provide DDL scripts that can be enriched with individual table space and table definition attributes as described in this book. Finally, the models can also integrate with Cognos software for reporting and analysis.

For more information about the IBM industry models, refer to the IBM Industry Models page at the following address:

<http://www.ibm.com/software/data/ips/products/industrymodels/>

Archived

## The system environment

In this chapter, we describe the system environment that we used and implemented for this project. The implementation is only a subset of the architecture that we propose in Chapter 4, “The architecture for the BI solution on System z” on page 59. Due to constraints in time and system resources, we chose a simplified but comprehensive implementation of the architecture.

This chapter contains the following sections:

- ▶ 6.1, “Implemented architecture” on page 92
- ▶ 6.2, “System configuration” on page 93
- ▶ 6.3, “System parameters of DB2 subsystems” on page 94
- ▶ 6.4, “Workload Manager configuration” on page 95
- ▶ 6.5, “zIIP utilization” on page 96

## 6.1 Implemented architecture

The general architecture of our scenario is described in 4.2, “The business intelligence architecture with System z” on page 61. With this architecture, we can the following paths to execute the extract, transform, and load (ETL) steps and query the data warehouse database:

- ▶ Replicate tables with Q replication
- ▶ Manipulate selected changed data via Event Publisher and DataStage, and load this data into the data warehouse database
- ▶ Extract and load data via DataStage
- ▶ Access legacy data on z/OS via Classic Federation
- ▶ Perform analysis and reporting by using Cognos software

Figure 6-1 shows the implementation that we have chosen. This is a small configuration that nevertheless implements all functions that are described by the architecture.

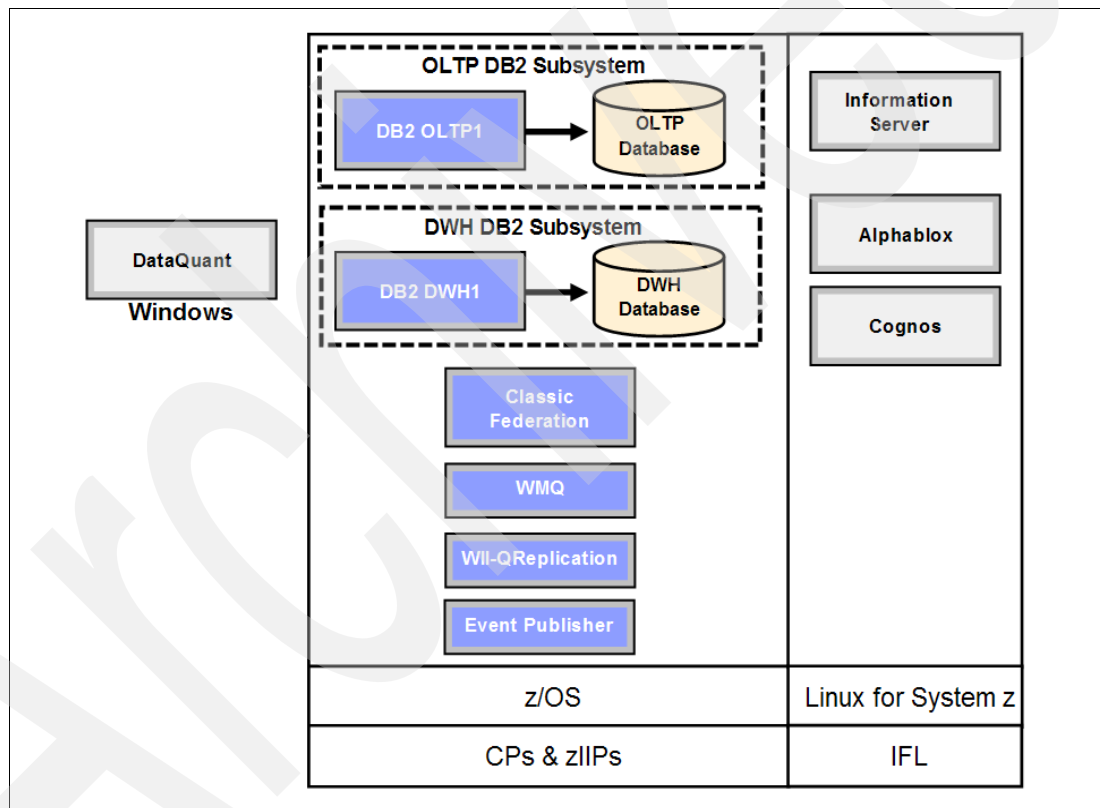


Figure 6-1 Implemented architecture

We ran all components in a single z/OS logical partition (LPAR) and used two DB2 subsystems, with no data sharing group.

In addition to the base components as described in 6.2, “System configuration” on page 93, we installed the following products:

- ▶ WebSphere Replication Server and Event Publisher  
For details, see Chapter 8, “Q replication and event publishing” on page 159.
- ▶ WebSphere Classic Federation  
For details, see 9.4, “Setting up WebSphere Classic Federation” on page 210.
- ▶ BatchPipes® to connect the products on Linux on System z with the products on z/OS  
For details, see 9.3, “Setting up BatchPipes” on page 206.
- ▶ Information Server  
For details, see 9.5, “Installing IBM Information Server” on page 228.
- ▶ Cognos 8 BI for analytics  
For details, see Chapter 13, “Reporting and analysis with Cognos 8 BI” on page 305.
- ▶ DataQuant 9.1  
For details, see 14.1, “DataQuant” on page 398.
- ▶ AlphaBlox  
For details, see 14.3, “AlphaBlox” on page 411.

## 6.2 System configuration

We used the following hardware (Figure 6-2 on page 94) during this project:

- ▶ IBM System z9 BC

We used two LPARs within the System z: one for z/OS and one for Linux on System z. The z/OS LPAR had four dedicated general purpose processors and two z9 Integrated Information Processors (zIIPs).

- The z/OS LPAR has the following base software:

- DB2 9.1 PUT0801
- VTAM® 1.8
- RACF® 1.8
- DFSMS/MVS™ 1.8
- DFSORT™ 1.8
- JES2 1.8
- z/OS 1.8

- The Linux on System z LPAR uses two shared processors. This LPAR has the following base software:

- Linux on System z (SUSE® Linux Enterprise Server (SLES) 10 SP1)
- WebSphere Information Server 8.0.1
- WebSphere Application Server 6.1

- ▶ IBM eServer pSeries® p550 2-way  
Software in the p550 is AIX® (Version 5300-07).
- ▶ Disk space on DS8300

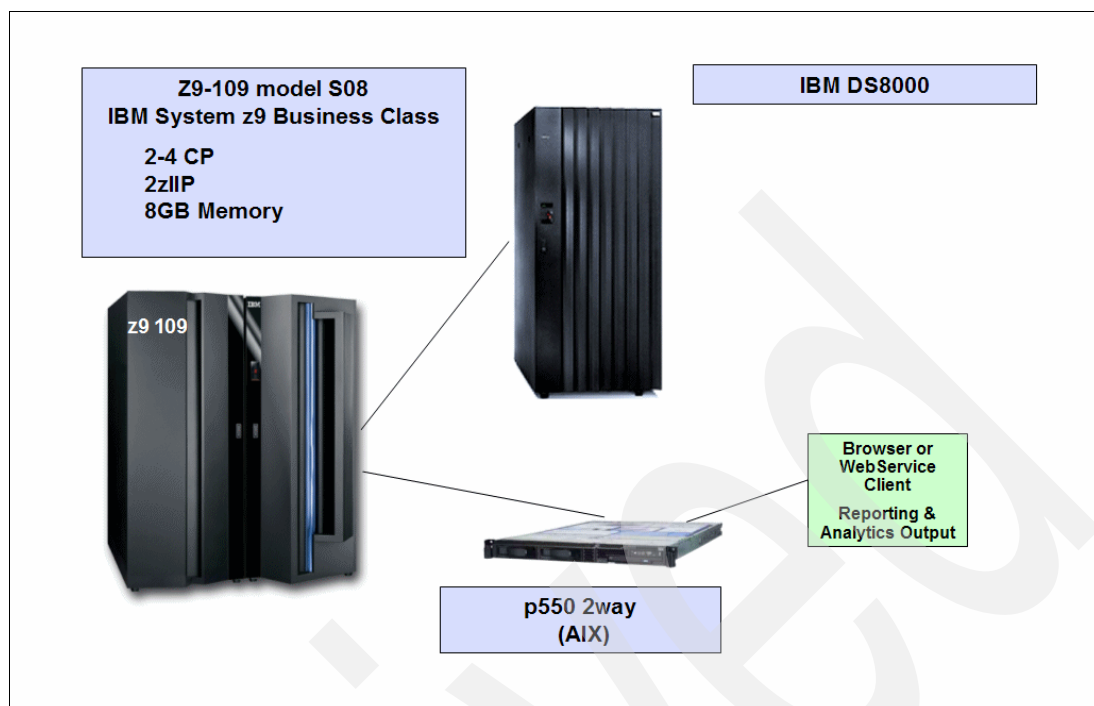


Figure 6-2 Data warehouse hardware

## 6.3 System parameters of DB2 subsystems

Some DSNZPARM settings need attention in a data warehouse environment with large databases and complex queries. Table 6-1 lists and briefly describes those settings.

Table 6-1 DSNZPARM definitions for a data warehouse

Recommendation for data warehouse	Comments
CDSSRDEF=ANY	ANY Allow parallelism for DWH. any: parallelism, 1: no parallelism
CONTSTOR=NO	NO For best performance, specify NO for this parameter. To resolve storage constraints in DBM1 address space, specify YES. See also MINSTOR.
DSVCI=YES	YES The DB2-managed data set has a VSAM control interval that corresponds to the buffer pool that is used for the table space.
MGEXTSZ=YES	YES Secondary extent allocations for DB2-managed data sets are to be sized according to a sliding scale.
MINSTOR=NO	NO Recommendation: For best performance, specify NO for this parameter. To resolve storage constraints in DBM1 address space, specify YES. See also CONTSTOR.
OPTIXIO=ON	OPTIXIO=ON: Provides stable I/O costing with significantly less sensitivity to buffer pool sizes. This is the new default and recommended setting.

Recommendation for data warehouse	Comments
PARAMDEG=X	#Processors <= X <= 2*#Processors If the concurrency level is low, the ratio can be higher.
SRTPOOL=8000	8000 The sort pool is defined as 8 MB.
STARJOIN=	The STARJOIN and SJTABLES parameter define the join processing of a query as described in 7.6, “Star schema processing” on page 124. If you use this parameter, check the performance improvements carefully.
MXQBCE	This is the maximum number of combinations considered by the optimizer at preparation time when joining N tables. If needed, set to this value to $(2^{**}N - 1)$ , where N is the TABLES_JOINED_THRESHOLD. The MXQBCE default is 1023 in DB2 9, with N=10.

## 6.4 Workload Manager configuration

Because the requirements of online transaction processing (OLTP) and data warehouse subsystems are different, the priorities for their workload are also different. For example, OLTP transactions are short and require a quick response time. However, traditional data warehouse transactions are long running and do not have any requirements about response time.

With the evolution of dynamic data warehousing and near real-time data in the data warehouse database, there are requirements for a quick response time for small queries running on data warehouse database. For example, to finalize the discount percentage to the customer, the customer's last 10 transactions details are queried from the data warehouse database.

As mentioned in 6.2, “System configuration” on page 93, our OLTP and data warehouse DB2 subsystems are on the same LPAR. Therefore, we configured two service classes (SERD911 for OLTP) and (SERD912 for DWH). Considering the requirements of OLTP, SERD911 was defined as a multiperiod service class with Average Percentile Response Time Goals. Example 6-1 shows our OLTP DB2 service class definitions.

*Example 6-1 Service class definitions for the OLTP DB2 subsystem*

#	Duration	Imp	Goal description
1	10000	1	90% complete within 00:00:00.500
2	25000	1	70% complete within 00:00:02.000
3			Discretionary

Considering the data warehouse database requirements, SERD912 was built with multiperiod definitions with the first period having Average Percentile Response Time for queries that require a quick response time. The second period is with execution velocity of 80% for ad hoc

*Example 6-2 Service class definitions for the data warehouse DB2 subsystem*

In these examples for service class definitions, we do not make any recommendations or suggest a best practice. The definition of service class must be decided depending upon the applications that are running, other workloads, and system configuration.

## 6.5 zIIP utilization

To demonstrate the exploitation of zIIP in a data warehouse environment running on System z, we configured two zIIP processors in our LPAR. We ran an RMF™ Workload Activity Report for the time interval in which we ran BI queries on a DB2 9 for z/OS subsystem coming from AIX and Linux on System z via DRDA. Figure 6-3 shows the RMF Workload Activity Report for the DDF workload for the data warehouse DB2 subsystem for the time interval in which ran our BI queries.

*Figure 6-3 RMF Workload Activity Report*



**Note:** The IIPCP value under APPL% signifies the percentage of CPU time used by zIIP eligible transactions running on standard central processor (CPs) due to zIIP either not being available or being busy. This is a subset of APPL% CP.

DB2 does not force a zIIP eligible transaction to only run on a zIIP. If the zIIP is busy and DB2 is not making its goal, it flows to a standard CP as seen in Figure 6-3 on page 96. The value of IIPCP is 4.56%. This is because, as the zIIP processors were busy (an average of 176% utilization for two processors for the time interval), some zIIP eligible transactions ran on standard CPs to meet the WLM service goal.

If the zIIP HONORPRIORITY is set to NO, we can force all zIIP eligible work to run on the zIIP.

Reporting class REPD912 is for the data warehouse DB2 subsystem DDF workload. The value IIP under SERVICE TIMES represents the zIIP service time in seconds. CPU represents the service time consumed on standard CPs and special purpose processors. The redirect percentage for zIIP, which in this case signifies the percentage of total DDF workload run on zIIP, is calculated as follows:

$$\text{Redirect \%} = \text{Service Time IIP} / \text{Service Time CPU} * 100$$
$$\text{Redirect \%} = 848.738 / 1583.286 * 100 = 53.6 \%$$

The redirect percentage for the interval is 53%. Therefore, the zIIP use in this case helps to save 53% of DB2 9 for z/OS software license cost.

**Redirect percentage:** The redirect percentage can vary depending upon the type of workload. This calculation was used to demonstrate the use of zIIP and its value in reducing the TCO in a data warehouse environment. A higher redirect percentage can be obtained for parallel SQL execution.

Archived

## Functions in DB2 for z/OS for a data warehouse

Several functions in DB2 for z/OS can be exploited while implementing a data warehouse solution. In this chapter, we describe some of the DB2 9 functions that we considered of special interest in our data warehouse scenario.

This chapter contains the following sections in which we use the data model and consider the business requirements on which this book is based:

- ▶ 7.1, “Index compression” on page 100
- ▶ 7.2, “Table space compression” on page 118
- ▶ 7.3, “Index-use tracking by using real-time statistics” on page 121
- ▶ 7.4, “Not logged table spaces” on page 121
- ▶ 7.5, “Exploiting the DB2 STATEMENT CACHE” on page 123
- ▶ 7.6, “Star schema processing” on page 124
- ▶ 7.7, “Index on expressions” on page 139
- ▶ 7.8, “Working with the ADD CLONE SQL command” on page 143
- ▶ 7.9, “Table space partitioning” on page 148
- ▶ 7.10, “Materialized query tables” on page 152
- ▶ 7.11, “OLAP functions” on page 154

## 7.1 Index compression

In data warehouse applications, it is common to apply extensive data compression to table spaces, which reduces disk space requirements. To improve performance, it is also common to have several indexes defined on the table with the purpose of solving queries with index-only access. Star schema implementations require specific indexes to support star schema processing, such as star join and pair-wise access methods. It is also common to end up with more disk space requirements for indexes than for table spaces.

DB2 9 for z/OS introduces the ability to create indexes of page sizes larger than 4K, because the index compression function is based on larger page sizes. Index compression provides a solution to these requirements, but its implementation requires a good understanding of its principles. Since data compression has been around DB2 since Version 3, it is easy to get confused when applying data compression analogies to index compression.

In general, a data warehouse environment is a good candidate for index compression. As a guideline, you may consider index compression where a reduction in index storage consumption is more important than a possible increase in CPU consumption.

In this section, we explain how index compression works and show its differences from data compression. We also show some data warehousing implementation examples.

For more information about index compression, refer to *Index Compression with DB2 9 for z/OS*, REDP-4345.

### 7.1.1 How index compression works

DB2 9 for z/OS uses a hybrid compression algorithm that uses *prefix compression*, as well as other methods, to compress index pages. Prefix compression is made possible by the fact that the keys on each index page are stored in sorted order from the left byte to right, either in ascending or descending order. The leftmost bytes that are common from one key to the next constitute the prefix. The prefix length varies throughout the index, but generally the longer the prefix is, the better the compression will be. Also, some index page elements, such as the key map are no longer written to disk because this information can be reconstructed when the compressed page is read from disk again.

We see in the following pages that the structure of the data, that is the presence of patterns, the order of the columns in the index, and the uniqueness of the keys, influences the degree of compression that can be achieved. We explain how the index compression ratio is not always the same as the space savings on disk.

DB2 only compresses *leaf pages*, which represent the vast majority of the index space. Figure 7-1 on page 101 shows a schematic representation of the b-tree index structure. Only the leaf pages (shaded) are compressed. Leaf pages are normally the large majority of the index pages. In some small indexes, the influence of non-compressed non-leaf pages may make the benefits of index compression negligible.

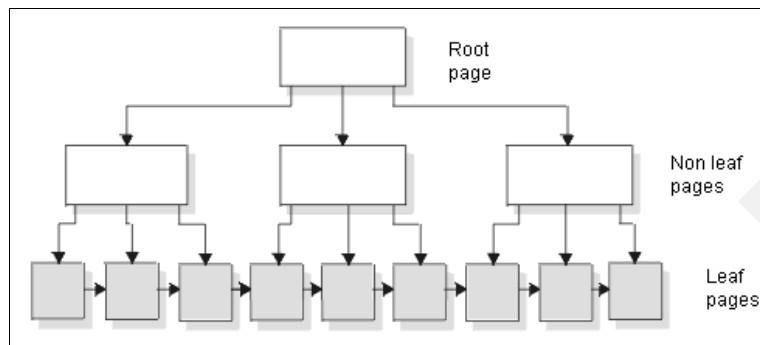


Figure 7-1 B-tree schematic representation

Index compression is made possible by the ability of creating indexes with page sizes bigger than 4K. The index compression implementation is a hybrid approach:

- On disk, pages are always 4K.
- You cannot define a compressed index in a 4K buffer pool. Pages are stored in page sizes of 8K, 16K, or 32K. In the following pages, we explain the impact of buffer pool selection, because the maximum disk saving is influenced by this choice.

Unlike data compression, log records for index keys and image copies for indexes are not compressed. Index compression does not use the Ziv-Lempel algorithm, does not use a compression dictionary, and is not assisted by hardware.

Table 7-1 shows the major implementation differences between indexes and data compression.

Table 7-1 Comparing index and data compression

	Index compression	Data compression
Technique	Prefix compression	Ziv-Lempel hardware assisted
Dictionary	Not needed, compression starts immediately after definition of the index. Altering an index to COMPRESS YES requires rebuilding the index	Must be built before getting data compressed
Data on disk	Compressed	Compressed
Data on DB2 log	Not compressed	Compressed
Data in buffer pool	Not compressed	Compressed
Image copies	Not compressed	Compressed

Figure 7-2 on page 102 shows a schematic comparison between data and index compression at run time:

- Data compression keeps data compressed into disk, and in the buffer pools, the data is uncompressed when passed to the application. Storage savings are perceptible at the disk and buffer pool levels. The CPU cost associated with data compression is higher for high hit ratio operations because each hit means a decompressed row passed to the application.
- Index compression keeps index data on 4K pages on disk and must expand the data into a larger buffer pool once read. Storage savings are for disk only. Because the buffer pool contains data (the index) already, no additional CPU impact is on operations with a high hit

ratio. If I/O is needed to read an index, the extra CPU cost for a index scan is probably relatively small, but the CPU cost for random access is likely to be significant. CPU cost for deletes and updates can be significant if I/O is necessary.

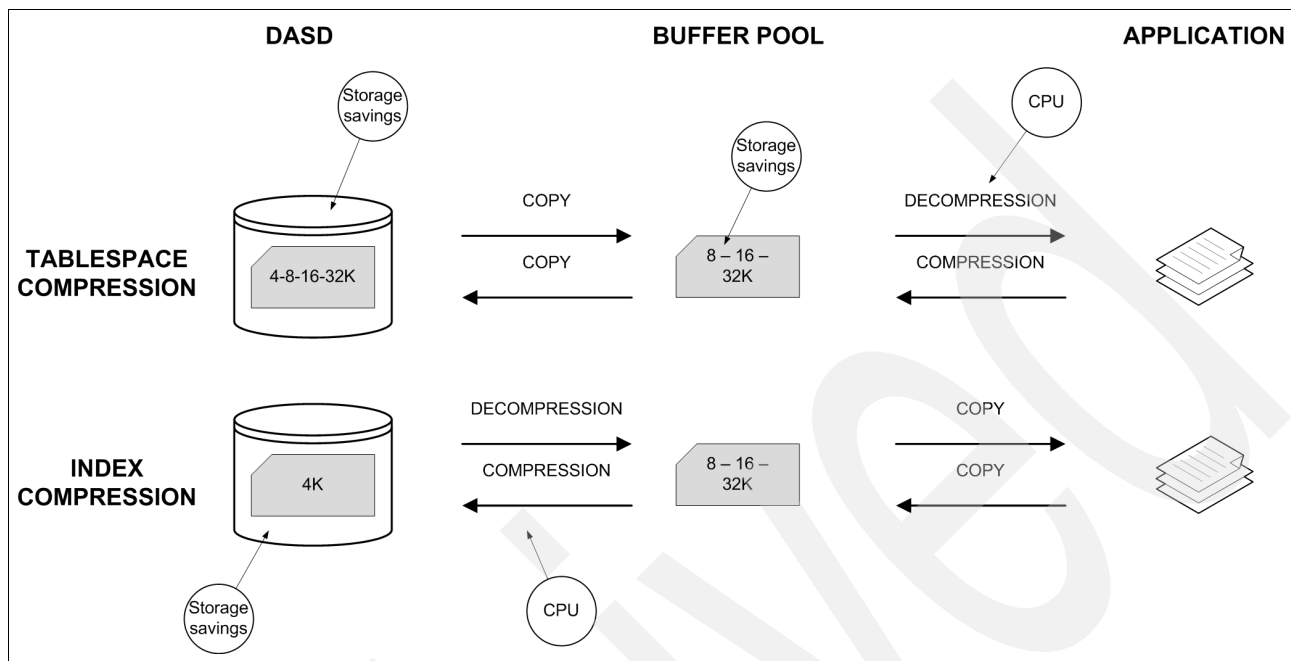


Figure 7-2 Data and index compression compared at run time

## 7.1.2 Implementation guidelines

The following considerations apply when implementing an index compression strategy:

### ► CPU

There is an impact on CPU depending on the type of workload:

- Pages are decompressed on read and compressed on write.
- A high buffer pool hit ratio does not impact CPU because the data is kept uncompressed on the buffer pool.
- Random access may be impacted by CPU penalties.
- Index scanning operations might use prefetch I/O that can decompress asynchronously. While still using extra CPU, this scenario may not be affected by elapsed time elongation. Some I/O bound queries may be improved by index compression.

- The compression ratio to be achieved depend on the data and how the columns are distributed in the index.
- The choice of the buffer pool determines the maximum disk space savings and the efficiency of memory utilization.
- Fixing a buffer pool in real storage by using the PGFIX(YES) option can improve performance for indexes with a high I/O rate, but it is not useful for the buffer pools of compressed indexes. DB2 does not perform I/O from the buffer pool of a compressed index, but from a 4 KB page I/O work area that is permanently page fixed. The pages in the work area are used for reading or writing to disk and do not act as a cache for the application. You may consider isolating compressed indexes to their own buffer pool if you need to page fix buffer pools for non-compressed indexes.

## Understanding the impact of buffer pool choice

The implementation of index compression stores data on disk on 4 KB pages. You have to decide whether to use a buffer pool of 8 KB, 16 KB, or 32 KB for the expansion of these pages.

DB2 must make sure that, when uncompressed, the data in each larger index page in the buffer pool page, compresses and fits in a 4 KB page. The maximum amount of disk storage saving that you can achieve when using index compression is limited by the buffer pool that you use for the index:

- ▶ For 8 KB, up to 51% compression
- ▶ For 16 KB, up to 76% compression
- ▶ For 32 KB, up to 88% compression

Figure 7-3 shows the maximum disk space compression that may be achieved as a function of the buffer pool choice. A single 8, 16, or 32 KB page in the buffer pool compresses onto a single 4 KB page on disk.

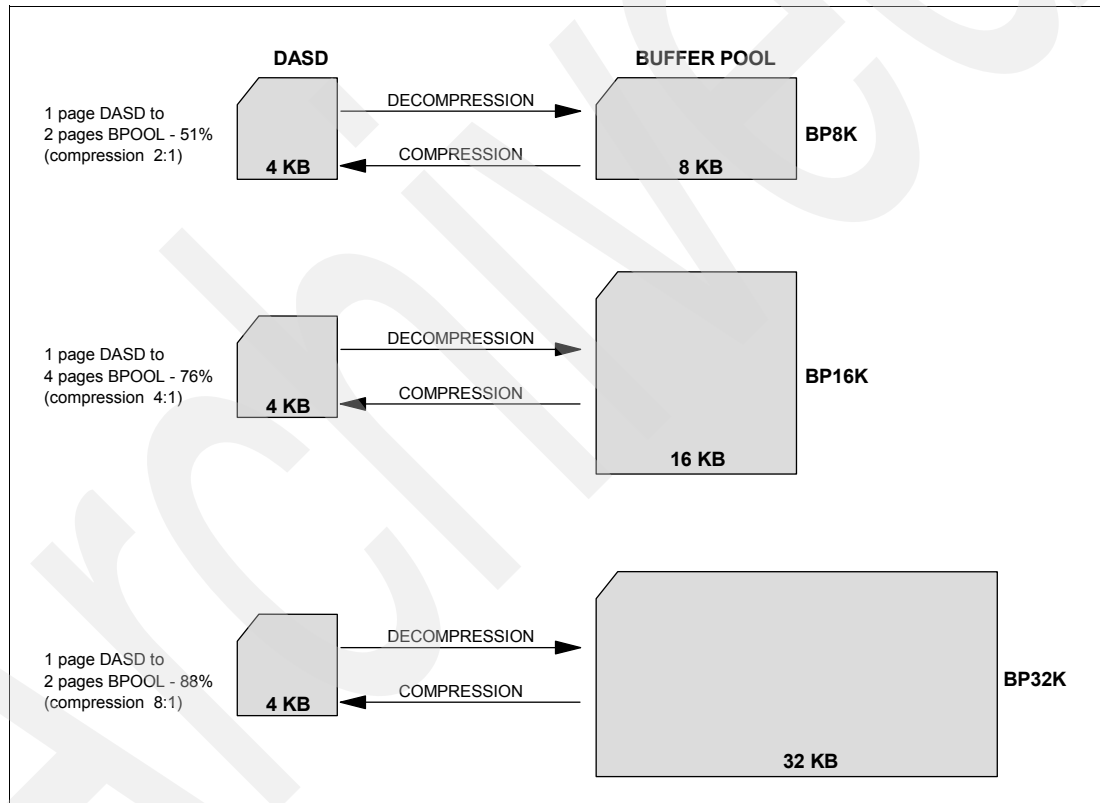


Figure 7-3 Maximum savings on disk in function of the buffer pool, assuming ideal index compression

Because an ideal or a high degree of index data compression is not always possible, using an inappropriate buffer pool page size can waste buffer pool memory without an increase in disk storage savings.

Figure 7-4 shows the impact on disk savings and buffer pool memory utilization as a function of the buffer pool choice for an index that achieves a compression ratio of four to one (4:1).

We can make the following observations:

- ▶ If the choice is BP8K:
  - A single page of the buffer pool contains 8K of uncompressed data. When compressed (because of the compression ratio 4:1), the 8K of uncompressed data squeezes into 2K on the disk page.
  - Because a compressed page must be contained in a single page when expanded in the buffer pool, a single disk page cannot contain more than 2K.
  - About the half of each leaf page on disk is empty, which increases the number of disk pages needed for containing the data. For example, 1000K of uncompressed leaf page data requires 500K of disk ( $1000\text{K uncompressed} = 250\text{K compressed} \times 50\% \text{ waste per page}$ ).
  - Selecting BP8K produces a 50% savings on disk, even if the data compression ratio is 4:1.
  - The limiting factor is the page size of the selected buffer pool, and a better disk savings is achieved by moving to a bigger buffer pool. If you have an 8K page in the buffer pool, and you always compress it to a 4K page, the best effective compression ratio that you can achieve is two to one (2:1).
- ▶ If the choice is BP16K:
  - Each disk page is full with 4K of index compressed data. This produces the maximum savings from a disk point of view. This perspective is that 1000K of uncompressed data is contained in 250K of disk space, which is a 4:1 disk savings.
  - Each buffer pool page contains  $4\text{K} \times (4:1) = 16\text{K}$  of expanded index data. This is a waste of a  $16\text{K} / 4\text{K} = 25\%$  of the buffer pool memory.
  - The limiting factor is the compression ratio of the index data, which produces a waste of buffer pool memory. A better resource utilization (buffer pool memory) is achieved by selecting a smaller buffer pool page size.
  - In this case, both disk and buffer pool pages are optimally used because they are full of data.
  - This produces a 75% savings on disk with full buffer pool page utilization.
- ▶ If the choice is BP32K:
  - Each disk page is full with 4K of index compressed data. This produces a maximum savings from a disk point of view. This perspective is that 1000K of uncompressed data is contained in 125K of disk space, which is an eight to one (8:1) disk savings.
  - Each buffer pool page contains  $4\text{K} \times (8:1) = 32\text{K}$  of expanded index data. This is a waste of a  $32\text{K} / 4\text{K} = 12.5\%$  of the buffer pool memory.
  - The limiting factor is the compression ratio of the index data, which produces a waste of buffer pool memory. A better resource utilization (buffer pool memory) is achieved by selecting a smaller buffer pool page size.
  - In this case, both disk and buffer pool pages are optimally used as they are full of data.
  - This produces an 88% savings on disk with full buffer pool page utilization.



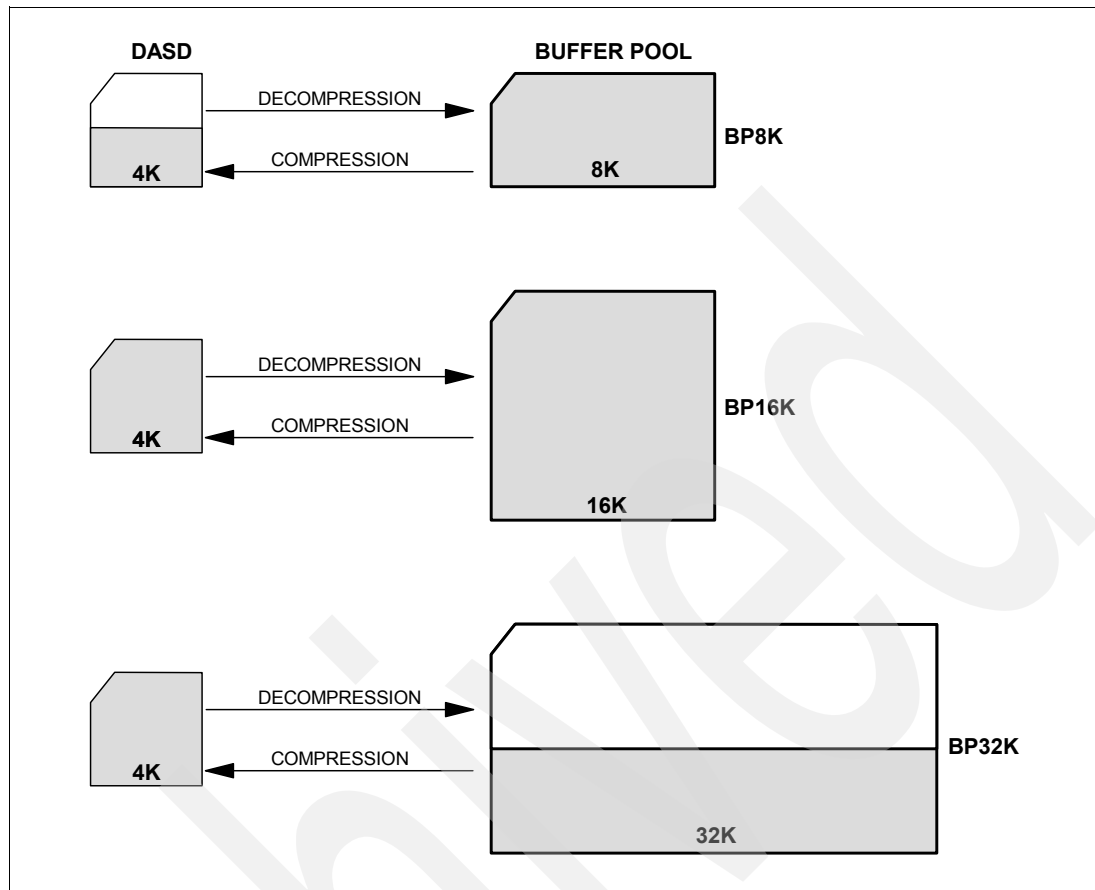


Figure 7-4 Impact of buffer pool selection for a compression ratio of 4:1

### Estimating space savings and selecting the best buffer pool

DSN1COMP has been enhanced in DB2 9 for z/OS to support index compression. This stand-alone utility is used to determine the following information:

- ▶ The space savings on disk due to index compression
- ▶ The buffer pool size that fits best the index

You can run this utility on the following types of data sets that contain uncompressed data or indexes:

- ▶ DB2 full image copy data sets
- ▶ VSAM data sets that contain DB2 table or index spaces
- ▶ Sequential data sets that contain DB2 table or index spaces

Consider the following points while using this tool:

- ▶ The user under which this job is executed must have read access on the VSAM data set that is the target of the utility.
- ▶ The LEAFLIM option of DSN1COMP limits how many leaf pages are examined. If not specified, all leaf pages in the data set are used.
- ▶ The estimation does not consider uncompressed non-leaf pages, which can have an important influence for small indexes.
- ▶ You can run DSN1COMP even when the DB2 subsystem is not operational. Before you use DSN1COMP, when the DB2 subsystem is running, issue the DB2 STOP DATABASE command to ensure that the target data sets are not allocated by DB2.

Example 7-1 shows how DSN1COMP can be used for analyzing a non-compressed index VSAM data set.

*Example 7-1 DSN1COMP JCL sample*

---

```
//DSN1COMP EXEC PGM=DSN1COMP ,
//STEPLIB DD DISP=SHR,DSN=SYS1.DSN.V910.SDSNLOAD
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DISP=OLD,DSN=DSN912.DSNDBC.COMPRESS.CUSTOMER.I0001.A001
```

---

Example 7-2 shows the output of the DSN1COMP execution and provides the following information:

- ▶ During this example, the parameter LEAFLIM was not used, and all the index pages were analyzed.
- ▶ A total of 10,000 K RIDs were processed, representing 136.179 K of uncompressed data.
- ▶ The tool estimated that this data can be compressed to 76.781 K. This means that ~ 56% (76.781 K / 136.179 K) of the original leaf page space is required based on how well the data compresses. However the actual savings on disk storage is determined by the adopted buffer pool.
- ▶ For this index, the limiting factor to the compression efficiency is how well the data compresses. The adoption of a buffer pool larger than 8K does not improve disk savings. For this example, an 8K buffer pool size yields the best result, because it provides the full compression benefit, but only results in a 12% increase in buffer pool space.

*Example 7-2 DSN1COMP output sample*

---

```
1DSN1999I START OF DSN1COMP FOR JOB CRISDXXX DSN1COMP
DSN1998I INPUT DSNAME = DSN912.DSNDBC.COMPRESS.CUSTOMER.I0001.A001 , VSAM
```

```
DSN1944I DSN1COMP INPUT PARAMETERS
PROCESSING PARMS FOR INDEX DATASET:
NO LEAFLIM WAS REQUESTED
```

```
DSN1940I DSN1COMP COMPRESSION REPORT
```

```
      38,611 Index Leaf Pages Processed
    10,000,000 Keys Processed
    10,000,000 Rids Processed
    136,719 KB of Key Data Processed
    76,781 KB of Compressed Keys Produced
```

**EVALUATION OF COMPRESSION WITH DIFFERENT INDEX PAGE SIZES:**

**8 K Page Buffer Size yields a**

```
43 % Reduction in Index Leaf Page Space
The Resulting Index would have approximately
57 % of the original index's Leaf Page Space
12 % of Bufferpool Space would be unused to
ensure keys fit into compressed buffers
```

**16 K Page Buffer Size yields a**

```
44 % Reduction in Index Leaf Page Space
The Resulting Index would have approximately
56 % of the original index's Leaf Page Space
55 % of Bufferpool Space would be unused to
ensure keys fit into compressed buffers
```

**32 K Page Buffer Size yields a**

**Note:** The primary motivation with which you look at index compression is for disk space savings. However, performance can be improved for I/O bound index scan processing where you may observe elapsed time reduction.

*Example 7-5 Altering an index defined on a 4K buffer pool to COMPRESS YES*

```
ALTER INDEX WORK.SJOIN_ALL_KEY COMPRESS YES;
-----+-----+-----+-----+-----+-----+-----+-----+
DSNT408I  SQLCODE = -676, ERROR:  THE PHYSICAL CHARACTERISTICS OF THE INDEX ARE
        INCOMPATIBLE WITH RESPECT TO THE SPECIFIED STATEMENT. THE STATEMENT
        HAS FAILED. REASON 2
-----+-----+-----+-----+-----+-----+-----+-----+
ALTER INDEX WORK.SJOIN_ALL_KEY BUFFERPOOL BP8K0 COMPRESS YES;
-----+-----+-----+-----+-----+-----+-----+-----+
DSNT404I  SQLCODE = 610, WARNING:  A CREATE/ALTER ON OBJECT WORK.SJOIN_ALL_KEY
        HAS PLACED OBJECT IN REBUILD PENDING
```

**Attention:** The alter of an index to COMPRESS YES or COMPRESS NO places the object in REBUILD PENDING status and, therefore, is unavailable for applications, as shown in Example 7-6.

*Example 7-6 Index in REBUILD PENDING after alter to COMPRESS YES or COMPRESS NO*

```
DSNT360I  -D912 *****
DSNT361I  -D912 *   DISPLAY DATABASE SUMMARY
        *   GLOBAL
DSNT360I  -D912 *****
DSNT362I  -D912      DATABASE = STARJOIN  STATUS = RW
        DBD LENGTH = 36332
DSNT397I  -D912
NAME      TYPE PART  STATUS              PHYERRLO PHYERRHI CATALOG  PIECE
-----
SJOINRAL  IX      L*   RW,PSRBD
```

## Analyzing compression efficiency by using the DB2 catalog

You can use the query shown in Example 7-7 to analyze how compression worked for a particular index.

*Example 7-7 Sample catalog query for index compression analysis*

```
SELECT
  SUBSTR(NAME,1,23) AS IDX
,COMPRESS AS COMP
,UNIQUERULE AS UNIQ
,COLCOUNT AS COLS
,AVGKEYLEN AS AVGKEYL
,PGSIZE
,SPACE
,NLEAF
,NLEVELS
FROM SYSIBM.SYSINDEXES
  WHERE DBNAME = 'COMPRESS'
ORDER BY IDX,COMP ,PGSIZE,UNIQ;
```

To obtain the space allocated information, you must execute the STOSPACE online utility before executing this query. This utility updates the DB2 catalog columns that indicate how much space is allocated for storage groups and related table spaces and indexes.

The amount of space allocated by each object is gathered from the appropriate VSAM catalog. Because the utility must have access to it, the DBD for the objects involved, the target database, table spaces, and index spaces must not be STOPPED.

The output from STOSPACE consists of updated values in the DB2 catalog. The following list describes the ones of interest for indexes. In each case, an amount of space is given in KB. If the value is too large to fit in the SPACE column, the SPACEF column is updated:

- ▶ SPACE in SYSIBM.SYSINDEXES shows the amount of space that is allocated to indexes.
- ▶ SPACE in SYSIBM.SYSINDEXPART shows the amount of space that is allocated to index partitions.

Example 7-8 shows a job control language (JCL) that can be used for executing STOSPACE.

*Example 7-8 STOSPACE JCL sample*

---

```
//STOSPACE EXEC DSNUPROC,SYSTEM=D912,
//          LIB='SYS1.DSN.V910.SDSNLOAD',
//          UID='STOSPACE'
//SYSIN DD *
STOSPACE STOGROUP SYSDEFLT
/*
```

---

To test the effect of different compressed index configuration, 12 indexes were created on the CUSTOMER table of the data model used for this book.

Example 7-9 shows a simplified Data Definition Language (DDL) of the non-compressed version of the indexes. For each one of these, three more were created on 8K, 16K, and 32K buffer pools.

*Example 7-9 Sample index for compression impact analysis*

---

```
CREATE INDEX ORIG.CUSTOMER
ON ORIG.CUSTOMER
(C_CUSTKEY ASC);

CREATE UNIQUE INDEX ORIG.CUSTOMER_UQ
ON ORIG.CUSTOMER
(C_CUSTKEY ASC);

CREATE INDEX ORIG.CUSTOMER_BIGX
ON ORIG.CUSTOMER
(C_CUSTKEY ASC,
 C_NAME ASC,
 C_ADDRESS ASC,
 C_NATIONKEY ASC,
 C_PHONE ASC,
 C_ACCTBAL ASC,
 C_MKTSEGMENT ASC,
 C_COMMENT ASC);
```

---

Example 7-10 shows the output of this query after execution of STOSPACE. It can be used for analyzing space saving efficiency. A sample of the data of the three first columns of the table is shown in Example 7-11. The data is as defined in the TPC-H standard.

*Example 7-10 Index compression query output sample*

---

IDX	COMP	UNIQ	COLS	AVGKEYL	PGSIZE	SPACE	NLEAF	NLEVELS
CUSTOMER	N	D	1	4	4	166320	38611	3
CUSTOMER_CPRS_8K	Y	D	1	4	8	86400	19813	3
CUSTOMER_CPRS_16K	Y	D	1	4	16	86400	19567	3
CUSTOMER_CPRS_32K	Y	D	1	4	32	86400	19567	4

---

CUSTOMER_UQ	N	U	1	4	4	136800	33113	3
CUSTOMER_CPRS_UQ_8K	Y	U	1	4	8	86400	19921	3
CUSTOMER_CPRS_UQ_16K	Y	U	1	4	16	86400	19921	3
CUSTOMER_CPRS_UQ_32K	Y	U	1	4	32	86400	19921	4
CUSTOMER_BIGX	N	D	8	167	4	2052000	502109	4
CUSTOMER_BIGX_CPRS_8K	Y	D	8	167	8	1738800	428885	4
CUSTOMER_BIGX_CPRS_16K	Y	D	8	167	16	1738800	428885	4
CUSTOMER_BIGX_CPRS_32K	Y	D	8	167	32	1789200	428885	5

**Important:** Consider that this method provides information about disk allocation savings, but cannot be used for analyzing buffer pool memory usage.

The following considerations can be made after the analysis of this example:

- ▶ The limiting factor to index compression for this example is how well the data can compress. Increasing the buffer pool page size does not increase disk savings but increases the buffer pool memory waste. A buffer pool with a page size of 8K is the best indication for any of the indexes in this example.
- ▶ Disk space requirements for the small indexes is ~ 63% (86.400K / 136.800K) of the original. For the big indexes, it is ~ 84% (1.738.800 / 2.052.000). This is related to the random nature of the data on the TCP-H table and is not a characteristic of index compression.

**Important:** The NLEAF column in SYSINDEXES can be used for estimating performance of index processing. It is the number of active leaf pages for a non-compressed index, but for compressed indexes, it is the number of pages on disk. In reality, the needed pages in the buffer pool are going to increase by a factor that depends on the compression ratio and the page size of the buffer pool.

The degree of compression and, in consequence, the disk savings may be impacted by the distribution of the columns inside the index, because index compression works better when the data presents long prefixes.

Considering the sample data of the TPC-H table CUSTOMER shown in Example 7-11, we note the following points:

- ▶ An index on ADDRESS does not compress well because there is no pattern or prefix in the sample data.
- ▶ The column order KEY, NAME presents a shorter prefix than the sequence NAME, KEY, and this second column distribution compresses better.

*Example 7-11 Customer table, TPC-H model data sample*

KEY	NAME	ADDRESS
1	Customer#000000001	IVhzIApeRb ot,c,
2	Customer#000000002	XSTf4,NCwDVaWNe6tEgvwfmRchLXak
3	Customer#000000003	MG9kdTD2WBH
4	Customer#000000004	XxVSJsLAGt
5	Customer#000000005	KvpyuHCp1rB84WgAiGV6sYpZq7Tj
...		

To validate these observations, four indexes were created as shown in Example 7-12. The only difference among them is the order of the involved columns and the fact that two of them are COMPRESS YES.

*Example 7-12 Sample index creation for analysis of column position influence*

---

```
CREATE INDEX IDX_KEY_NAME ON CUSTOMER (KEY,NAME);
CREATE INDEX IDX_KEY_NAME_COMP ON CUSTOMER (KEY,NAME) COMPRESS YES;
CREATE INDEX IDX_NAME_KEY ON CUSTOMER (NAME,KEY);
CREATE INDEX IDX_NAME_KEY_COMP ON CUSTOMER (NAME,KEY) COMPRESS YES;
```

---

After executing the STOSPACE utility, the catalog query shown in Example 7-7 on page 108 produces the results in Example 7-13.

*Example 7-13 Index comparison - Influence on compression ratio of column order*

---

-----	-----	-----	-----	-----	-----	-----	-----	-----
IDX	COMP	UNIQ	COLS	AVGKEYL	PGSIZE	SPACE	NLEAF	NLEVELS
-----	-----	-----	-----	-----	-----	-----	-----	-----
IDX_KEY_NAME	N	D	2	25	4	403920	97088	4
IDX_KEY_NAME_COMP	Y	D	2	25	8	292320	69533	4
IDX_NAME_KEY	N	D	2	25	4	403920	97088	4
IDX_NAME_KEY_COMP	Y	D	2	25	8	198720	48077	4

---

Note the following observations:

- ▶ An uncompressed index uses the same space and number of leaf pages for both column combinations.
- ▶ The index IDX\_KEY\_NAME\_COMP compresses ~ 72% (292.320/403.920), with a savings of ~ 28% in disk requirements.
- ▶ The index IDX\_NAME\_KEY\_COMP compresses better at ~49% (198.720/403.920), with a savings of ~ 51% in disk requirements.
- ▶ This example confirms that, because of the way index compression works, the column distribution within an index affects the degree of compression that can be achieved.

Using the JCL in Example 7-14, we printed the first leaf page, which contains the same information, of both indexes.

*Example 7-14 DSN1PRNT JCL sample*

---

```
//PRINT2 EXEC PGM=DSN1PRNT,
//          PARM=(PRINT(002F0,002F5),FORMAT)
//STEPLIB DD DISP=SHR,DSN=SYS1.DSN.V910.SDSNLOAD
//SYSUDUMP DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DISP=OLD,DSN=DSN912.DSNDBC.COMPRESS.xxxxxxx.I0001.A001
```

---

Example 7-15 on page 112 shows an extract of the job output for the combination NAME, KEY. Example 7-16 on page 112 shows an extract of the index on KEY, NAME. The character representation of the index page contents, to the right of the examples, helps you to understand how the column disposition within an index can impact the degree of compression:

- ▶ The (NAME, KEY) order presents a longer prefix, and the data portion and less data needs to be written for each key
- ▶ The (KEY, NAME) has a shorter prefix, which requires more data to be written when compressing.

- ▶ IPKEYLN is the maximum length of index keys, which is the same for both indexes, '0020'X.
- ▶ IPNKEYS is the number of key values that are contained in the page. Even if the key length is the same for both indexes, the index on (NAME,KEY) contains 208 keys, while the index on (KEY,NAME) holds only 154.

**Example 7-15 Print of first leaf page for index NAME - KEY**

```

PAGE: # 00000003 -----
TYPE 2 INDEX LEAF PAGE:  PGCOMB='10'X  PGLOGRBA='00030138F569'X  PGNUM='00000003'X  PGFLAGS='7C'X

INDEX PAGE HEADER:  IPIXID='003D'X  IPFLAGS='4C68'X  IPNKEYS=208  IPFREESP='034C'X  IPFREEP='1B12'X
                   IPFCHSIZ='0000'X  IPFCHNXT='0000'X  IPKEYLN='0020'X  IPPRNTPG='00000002'X

COMPRESSED INDEX LEAF PAGE FOLLOWS (UNFORMATTED):
0000 10000301 38F56900 0000037C 00000000 00000000 003D4C68 00D0034C 1B120000 *.....5.....@.....<.....*
0020 00000020 00000002 010A0003 00D00000 00000005 00000000 00000000 80000001 *.....*
0040 06A30101 190000C3 A4A2A396 9485997B F0F0F0F0 F0F0F0F0 F1800000 01020200 *.....C.....#000000001.....*
0060 011912F2 80000002 02020101 1912F380 00000302 02020119 12F48000 00040202 *...2.....3.....4.....*
0080 03011912 F5800000 05020204 011912F6 80000006 02020501 1912F780 00000702 *...5.....6.....7.....*
00A0 02060119 12F88000 00080202 07011912 F9800000 09020208 011911F1 F0800000 *...8.....9.....10.....*
00C0 0A020209 011912F1 8000000B 02020A01 1912F280 00000C02 020B0119 12F38000 *.....1.....2.....3.....*
00E0 00D02020 0C011912 F4800000 0E02020D 011912F5 8000000F 02020E01 1912F680 *.....4.....5.....6.....*
0100 00001002 020F0119 12F78000 00110202 10011912 F8800000 12020211 011912F9 *.....7.....8.....9.....*
0120 80000013 02021201 1911F2F0 80000014 02021301 1912F180 00001502 02140119 *.....20.....1.....*
.....

```

**Example 7-16 Print of first leaf page for index KEY - NAME**

```

PAGE: # 00000003 -----
TYPE 2 INDEX LEAF PAGE:  PGCOMB='10'X  PGLOGRBA='000301380E98'X  PGNUM='00000003'X  PGFLAGS='7C'X

INDEX PAGE HEADER:  IPIXID='003B'X  IPFLAGS='4C68'X  IPNKEYS=154  IPFREESP='0AAE'X  IPFREEP='141C'X
                   IPFCHSIZ='0000'X  IPFCHNXT='0000'X  IPKEYLN='0020'X  IPPRNTPG='00000002'X

COMPRESSED INDEX LEAF PAGE FOLLOWS (UNFORMATTED):
0000 10000301 380E9800 0000037C 00000000 00000000 003B4C68 009A0AAE 141C0000 *.....@.....<.....*
0020 00000020 00000002 010A0003 009A0000 00000005 00000000 00000000 80000001 *.....*
0040 00140101 19008000 000100C3 A4A2A396 9485997B F0F0F0F0 F0F0F0F0 F1020200 *.....C.....#000000001.....*
0060 01190302 00C3A4A2 A3969485 997BF0F0 F0F0F0F0 F0F0F020 02010119 030300C3 *.....C.....#000000002.....C*
0080 A4A2A396 9485997B F0F0F0F0 F0F0F0F0 F3020202 01190304 00C3A4A2 A3969485 *.....#000000003.....C.....*
00A0 997BF0F0 F0F0F0F0 F0F0F402 02030119 030500C3 A4A2A396 9485997B F0F0F0F0 *...#000000004.....C.....#0000*
00C0 F0F0F0F0 F5020204 01190306 00C3A4A2 A3969485 997BF0F0 F0F0F0F0 F0F0F602 *00005.....C.....#000000006.*
00E0 02050119 030700C3 A4A2A396 9485997B F0F0F0F0 F0F0F0F0 F7020206 01190308 *.....C.....#000000007.....*
0100 00C3A4A2 A3969485 997BF0F0 F0F0F0F0 F0F0F802 02070119 030900C3 A4A2A396 *...C.....#000000008.....C....*
0120 9485997B F0F0F0F0 F0F0F0F0 F9020208 0119030A 00C3A4A2 A3969485 997BF0F0 *...#000000009.....C.....#00*
.....

```

A portion of the character representation of the leaf page (to the right of the report) is reproduced for clarity in Example 7-17.

**Example 7-17 How data is stored with index compression**

NAME - KEY	KEY - NAME
*.....C.....#000000001.....*	*.....C.....#000000001...*
*...2.....3.....4.....*	*.....C.....#000000002.....C*
*...5.....6.....7.....*	*.....#000000003.....C.....*
*...8.....9.....10.....*	*...#000000004.....C.....#0000*
*.....1.....2.....3.....*	*00005.....C.....#000000006.*
*.....4.....5.....6.....*	*.....C.....#000000007.....*
*.....7.....8.....9.....*	*...C.....#000000008.....C....*



**Important:** If possible, consider the attributes of the data and create indexes with the columns combination that provides the longer prefix for better compression results. When the column distribution within an index has been fixed, you cannot control or influence the degree of compression that can be achieved on the data on which the index is based. However you can control the buffer pool placement, which may be a compromise between disk saving and memory utilization.

## Performance evaluation examples

To validate the index compression impact on performance, we used the TPC-H order priority checking query. This query counts the number of orders placed in a given quarter of a given year in which at least one line item was received by the customer later than its committed date. The query lists the count of such orders for each order priority sorted in ascending priority order. This is considered a good example of a data warehouse query. This and other selected queries were included in assembler programs for performance evaluation.

**Note:** The purpose of these example is not to provide figures on expected CPU impact, but rather to show a method of analysis. For CPU impact on index compression, refer to *Index Compression with DB2 9 for z/OS*, REDP-4345.

### TPC-H order priority checking query and index compression

Example 7-18 shows the query and the indexes that were created for the testing. Refer to Example A-1 on page 425, which shows one of the programs that was used for these tests, and Example A-2 on page 428, which shows the JCL for assembly, bind, and execution.

*Example 7-18 TPC-H order priority checking query*

---

```

SELECT
    O_ORDERPRIORITY,
    COUNT(*) AS ORDER_COUNT
FROM ORIG.ORDERS
WHERE
    O_ORDERDATE >= '1992-01-01'
    AND O_ORDERDATE < '1993-01-04'
    AND EXISTS (
        SELECT 1
        FROM ORIG.LINEITEM
        WHERE
            L_ORDERKEY = O_ORDERKEY
            AND L_COMMITDATE < L_RECEIPTDATE
    )
GROUP BY O_ORDERPRIORITY
ORDER BY O_ORDERPRIORITY;

CREATE INDEX ORDERS2 ON ORDERS (O_ORDERKEY);
CREATE INDEX ORDERS2_COMPR ON ORDERS (O_ORDERKEY) COMPRESS YES;

```

---

Example 7-19 shows the physical properties of these indexes.

*Example 7-19 Index for compression test - Physical properties*

---

IDX	COMP	UNIQ	COLS	AVGKEYL	PGSIZE	SPACE	NLEAF	NLEVELS
ORDERS2	N	D	1	7	4	198720	46949	3
ORDERS2_COMPR	Y	D	1	7	8	97920	23256	3

---

Figure 7-5 shows the Optimization Service Center access path graphical representation for the query by using the compressed version of the index. Its shows that all the leaf pages of the index (23.256) will be scanned.

**Note:** During our tests, the optimizer always selected the non-compressed version of the index, even if the number of leaf pages was notably less for the compressed version.

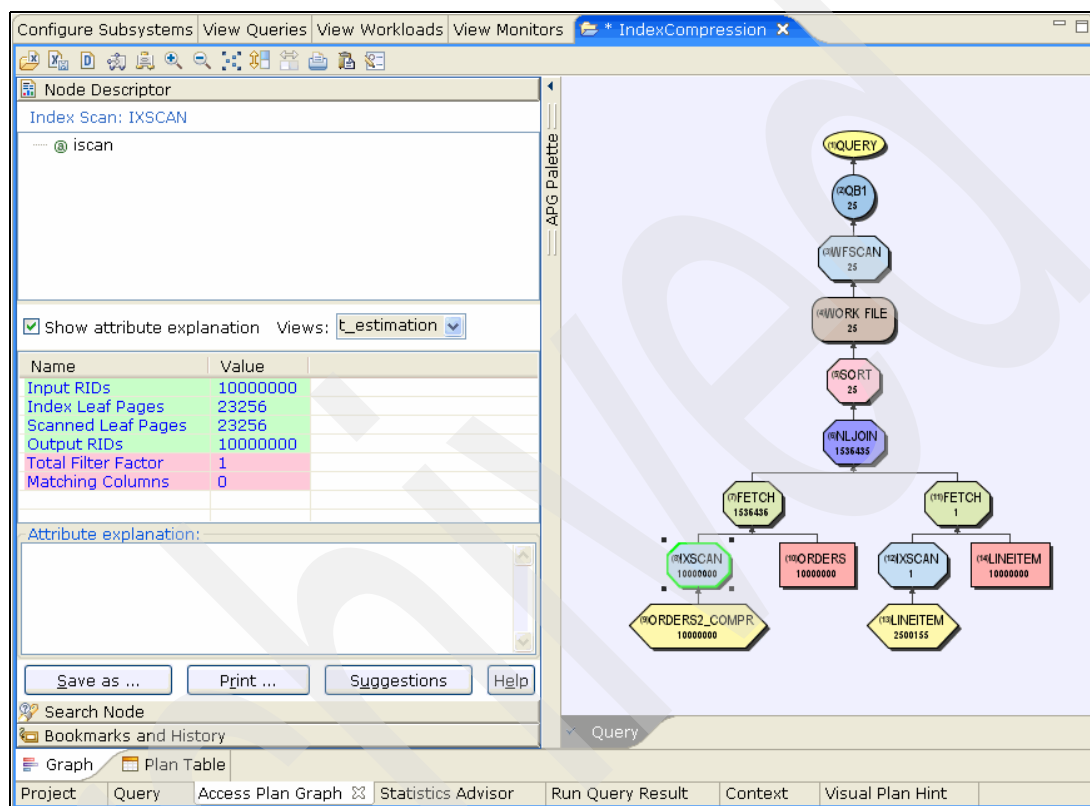


Figure 7-5 Access path involving a compressed index

**Important:** You must consider the CPU impact on DBM1 during the analysis of index compression performance:

- ▶ All synchronous CPU time is counted as class 2 CPU time.
- ▶ All asynchronous CPU time is counted as DBM1 SRB time.

To evaluate the DBM1 CPU impact, you may proceed as follows:

1. Issue a MODIFY TRACE command to produce a new statistics record before starting your testing.
2. As the single user of the DB2 subsystem, perform your tests several times to produce a valid average.
3. Issue again a MODIFY TRACE command to cut another STATISTIC record.
4. (Optional) Issue an SMF switch command to have SMF data immediately available for analysis (/I SMF).

The JCL in Example 7-20 can be used to issue the MODIFY command.

*Example 7-20 JCL sample for issuing a MODIFY TRACE command*

---

```
//DB2COMM EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB DD DISP=SHR,DSN=SYS1.DSN.V910.SDSNLOAD
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD *
    DSN SYSTEM(D912)
    -DIS TRACE(*)
    -MOD TRA(S) CLASS(1,3,4,5,6) TN0(01)
/*
```

---

The IBM Tivoli® OMEGAMON XE for DB2 Performance Expert on z/OS command sample in Example 7-21 can be used to generate an accounting and statistics report for CPU impact analysis.

*Example 7-21 OMEGAMON XE for DB2 Performance Expert on z/OS command report sample*

---

```
GLOBAL
    SSID(D912)
    LEVEL(DETAIL)
    INDEX(YES)
    OWNER(CRIS)

ACCOUNTING
    REDUCE
        FROM(04/21/08,05:00:00.00)
        TO(04/21/08,23:00:00.00)
    REPORT
        DDNAME(ACCREPO)
    TRACE
        DDNAME(ACCTRA)

STATISTICS
    REDUCE
        FROM(04/21/08,05:00:00.00)
        TO(04/21/08,23:00:00.00)
    REPORT
        /*STATS REPORT*/
        DDNAME(STATS)
    TRACE
        DDNAME(STATSTRA)

EXEC
```

---

Example 7-22 and Example 7-23 show combined extracts of OMEGAMON XE for DB2 Performance Expert on z/OS reports. These reports show accounting and statistics information for the execution of two sample programs against compressed and non-compressed index versions. The difference in the number of leaf pages for the indexes is 46949 (uncompressed) – 23256 (compressed) = 23693. The OMEGAMON XE for DB2 Performance Expert on z/OS reports indicate that approximately 23K fewer getpages were executed during the execution of the program by using the compressed index.

**Example 7-22 Performance Expert report sample using a non-compressed index**

#OCCURS	#COMMIT	INSERTS	OPENS	PREPARE	CLASS2	EL.TIME	BUF.UPDT	LOCK	SUS			
PRIMAUTH			#DISTR	SELECTS	UPDATES	CLOSES	CLASS1	EL.TIME	CLASS2	CPUTIME	SYN.READ	#LOCKOUT
PLANNAME			#ROLLBK	FETCHES	MERGES	DELETES	CLASS1	CPUTIME		GETPAGES	TOT.PREF	
CRIS			5	5	0.00	1.00		0.00		30.660371	0.00	0.00
RED01			0	0.00	0.00	1.00		31.760353		20.949162	0.00	0
			0	6.00	0.00	0.00		20.953251		673.7K	24463.00	

PROGRAM NAME	TYPE	#OCCURS	SQLSTMT	CL7	ELAP.TIME	CL7	CPU TIME	CL8	SUSP.TIME	CL8	SUSP
RED01	PACKAGE	5	8.00		30.660365		20.949157		7.353675		12.1K

CPU TIMES	TCB TIME	PREEMPT	SRB	NONPREEMPT	SRB	TOTAL TIME	PREEMPT	IIP	SRB
SYSTEM SERVICES ADDRESS SPACE	0.012398	0.000000		0.009259		0.021657			N/A
DATABASE SERVICES ADDRESS SPACE	0.000754	0.000000		19.736916		19.737670			0.000000
IRLM	0.000003	0.000000		0.019269		0.019272			N/A
DDF ADDRESS SPACE	0.000121	0.000000		0.000108		0.000229			0.000000
TOTAL	0.013276	0.000000		19.765553		19.778829			0.000000

**Example 7-23 Performance Expert report sample using a compressed index**

#OCCURS	#COMMIT	INSERTS	OPENS	PREPARE	CLASS2	EL.TIME	BUF.UPDT	LOCK	SUS			
PRIMAUTH			#DISTR	SELECTS	UPDATES	CLOSES	CLASS1	EL.TIME	CLASS2	CPUTIME	SYN.READ	#LOCKOUT
PLANNAME			#ROLLBK	FETCHES	MERGES	DELETES	CLASS1	CPUTIME		GETPAGES	TOT.PREF	
CRIS			5	5	0.00	1.00		0.00		30.884626	0.00	0.00
REDC1			0	0.00	0.00	1.00		31.327963		20.978062	0.00	0
			0	6.00	0.00	0.00		20.981970		650.0K	24450.00	
PROGRAM NAME	TYPE	#OCCURS	SQLSTMT	CL7	ELAP.TIME	CL7	CPU TIME	CL8	SUSP.TIME	CL8	SUSP	
REDC1	PACKAGE	5	8.00		30.884621		20.978057		6.858745		11.9K	
CPU TIMES			TCB TIME		PREEMPT SRB		NONPREEMPT SRB		TOTAL TIME		PREEMPT IIP SRB	
SYSTEM SERVICES ADDRESS SPACE			0.010801		0.000000		0.009304		0.020104		N/A	
DATABASE SERVICES ADDRESS SPACE			0.000843		0.000000		24.674301		24.675144		0.000000	
IRLM			0.000005		0.000000		0.019982		0.019987		N/A	
DDF ADDRESS SPACE			0.000153		0.000000		0.000117		0.000270		0.000000	
TOTAL			0.011801		0.000000		24.703704		24.715505		0.000000	

The reports also show a CPU regression of about 25% on the DATABASE SERVICES ADDRESS SPACE CPU.

### **Index compression and index scan**

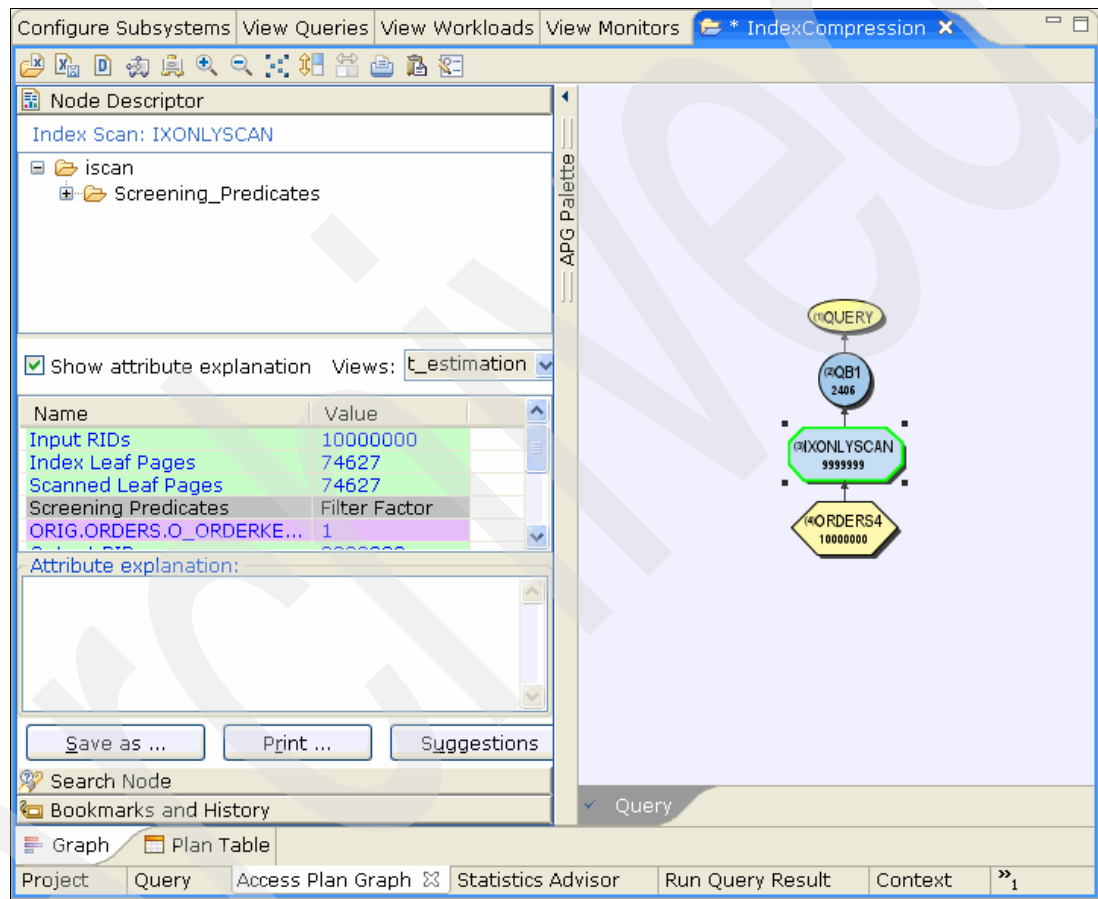
The previous example concentrates on a data warehouse query in which the activity against a compressed index was part of a bigger process. To validate the impact of index compression

on a query where the getpages are done mostly on the indexes, we used the query shown in Example 7-24.

*Example 7-24 Sample query index only scan*

```
SELECT
O_ORDERDATE ,
COUNT ( * ) AS ORDER_COUNT
FROM ORIG . ORDERS
where O_ORDERKEY > 0
GROUP BY O_ORDERDATE;
```

Figure 7-6 shows the access path of the test programs.



*Figure 7-6 Index only scan on an uncompressed index*

The index in Example 7-25 was created on four versions: uncompressed and compressed on 8K, 16K, and 32K buffer pools.

*Example 7-25 Index creation for index only scan of compressed index test*

```
CREATE INDEX ORIG.ORDERS4
ON ORIG.ORDERS
(O_ORDERDATE ASC,
O_ORDERKEY ASC,
O_CUSTKEY ASC);
```

Example 7-26 shows the catalog information for these indexes.

*Example 7-26 Details for indexes involved in index only scan*

-----	-----	-----	-----	-----	-----	-----	-----	-----
IDX	COMP	UNIQ	COLS	AVGKEYL	PGSIZE	SPACE	NLEAF	NLEVELS
-----	-----	-----	-----	-----	-----	-----	-----	-----
ORDERS4	N	D	3	17	4	313200	74627	4
ORDERS4_COMPR	Y	D	3	17	8	151200	37038	4
ORDERS4_COMPR16	Y	D	3	17	16	151200	35964	4
ORDERS4_COMPR32	Y	D	3	17	32	151200	35964	4

Table 7-2 summarizes the performance measurements.

*Table 7-2 Index compression performance compared for index scan operations*

	No compression	Compress 8K	Compress 16K	Compress 32K
Getpages	74630	37041	35967	35967
CPU	4.22	4.07	4.06	4.17
Elapsed time	4.92	5.82	5.61	6.49
Suspend time	0.01	0.01	0.08	1.39
Total prefetch	2346	2330	4533	9219
SQL statements	2409	2409	2409	2409

## 7.1.4 Considerations on index compression

Use index compression on an index space if the index space has the following characteristics:

- It is large enough to save significant disk space, for example larger than 10 MB, and to check the benefit of pages saved versus the cost of compress or decompress.
- It compresses by at least 35% and can fit at least one more rows on a page (index length < compression ratio x page size).

Watch the trade off of CPU and memory overhead versus savings on disk. Keep in mind that, with index compression, the CPU overhead is incurred at read and write I/O execution time only. This is quite different from data compression where the CPU overhead is incurred while each row is transferred to and from the buffer pool to the application.

With the larger index page sizes required, probably less free space is required, but free space is still useful to avoid page splits on inserts.

A larger buffer pool space for random processing avoids more frequent I/O and reduces compression and decompression.

## 7.2 Table space compression

Data warehouse environments that run on DB2 9 for z/OS can use the table space compression option. This option uses the hardware compression feature of System z, which is described in 3.2.3, "Hardware data compression" on page 36. For details, see *DB2 for OS/390 and Data Compression*, SG24-5261.

In our data warehouse scenario, we used the table space compression option. To demonstrate its value, we started with uncompressed table space. This uncompressed table space contains 1 million rows and is approximately 1.4 GB in size. To estimate the effect of compression on this table space, we ran the DSN1COMP utility on the uncompressed table space to find the percentage of space savings after compression. To save time, we ran the utility to scan the first 100,000 rows only to provide the compression report. The DSN1COMP compression report showed a 55 percent space savings. See Example 7-27.

*Example 7-27 DSN1COMP compression report*

---

```

DSN1998I INPUT DSNNAME = DSN912.DSNDBD.COMPRESS.LINEITEM.I0001.A001 ,VASM
DSN1944I DSN1COMP INPUT PARAMETERS
      4,096 DICTIONARY SIZE USED
      0 FREEPAGE VALUE USED
      5 PCTFREE VALUE USED
100,000 ROWLIMIT REQUESTED
      1 NUMBER OF PARTITIONS
      ESTIMATE BASED ON DB2 REORG METHOD
      255 MAXROWS VALUE USED

DSN1940I DSN1COMP COMPRESSION REPORT
      14,190 KB WITHOUT COMPRESSION
      6,055 KB WITH COMPRESSION
      57 PERCENT OF THE BYTES WOULD BE SAVED

      793 ROWS SCANNED TO BUILD DICTIONARY
100,000 ROWS SCANNED TO PROVIDE COMPRESSION ESTIMATE
      4,096 DICTIONARY ENTRIES

      148 BYTES FOR AVERAGE UNCOMPRESSED ROW LENGTH
      65 BYTES FOR AVERAGE COMPRESSED ROW LENGTH

      16 DICTIONARY PAGES REQUIRED
      3,847 PAGES REQUIRED WITHOUT COMPRESSION
      1,711 PAGES REQUIRED WITH COMPRESSION
      55 PERCENT OF THE DB2 DATA PAGES WOULD BE SAVED

DSN1994I DSN1COMP COMPLETED SUCCESSFULLY,      3,879 PAGES PROCESS

```

---

The table space in Example 7-27 is a good candidate for compression because it is giving a 55% savings in disk space. The compression of this table space will also result in shorter I/O and paging times, with more data instantly accessible in caches and in buffer pools. To use the hardware compression for this table space, we must issue the SQL ALTER TABLESPACE statement with the COMPRESS YES clause as shown in Example 7-28.

*Example 7-28 Altering a table space for using compression*

---

```

ALTER TABLESPACE COMPRESS.LINEITEM COMPRESS YES

```

---

After the ALTER statement completes, the REORG utility must be run on the table space to compress the existing data. Example 7-29 on page 120 shows the compression report of the REORG utility that we ran against the table space. It shows a 56% disk space savings due to compression. In this case, we saved approximately 800 MB of physical disk space because of compression.

*Example 7-29 Compression Report of REORG utility*

---

```
-D912 123 17:11:38.11 DSNURWT - COMPRESSION REPORT FOR TABLE SPACE COMPRESSION REPORT FOR
TABLE SPACE COMPRESS.LINEITEM, PARTITION 1
1418278 KB WITHOUT COMPRESSION
611473 KB WITH COMPRESSION
    56 PERCENT OF THE BYTES SAVED FROM COMPRESSED DATA ROWS

    100 PERCENT OF THE LOADED ROWS WERE COMPRESSED

    148 BYTES FOR AVERAGE UNCOMPRESSED ROW LENGTH
    65 BYTES FOR AVERAGE COMPRESSED ROW LENGTH

382556 PAGES REQUIRED WITHOUT COMPRESSION
167999 PAGES REQUIRED WITH COMPRESSION
    56 PERCENT OF THE DB2 DATA PAGES SAVED USING COMPRESSED DATA
```

---

**Note:** The percentage of disk savings due to compression may vary from case to case because it depends on multiple factors such as data type definitions, data repetition, and data quality.

## 7.2.1 Considerations on data compression

Use data compression on a table space if the table space has the following characteristics:

- ▶ It is large enough to provide a significant space savings (for example, larger than 10 MB) or pages saved are greater than 5 MB.
- ▶ It compresses by at least 30%.
- ▶ It can fit at least one more rows on a page, where  $\text{row length} < \text{compression ratio} \times \text{page size value}$ .

There is a maximum number of rows per page:

$\text{row length} > \text{page size} / 255 - 6$

If the row lengths are shorter than the following values, then there is no savings:

- ▶ 4K → 10 bytes
- ▶ 8K → 26 bytes
- ▶ 16k → 58 bytes
- ▶ 32K → 122 bytes

With data compression, buffer pool memory use generally improves, because buffers contain compressed data, and getpages are reduced for sequential access.

The processor overhead must be considered, especially with older processors, such as G4, and when there are many updates or every data row is examined. Index access is generally inexpensive.

Generally you want to increase free space with compression to handle the variability of the rows and avoid off page pointers

You must consider the space requirements for dictionaries (one per partition), which are loaded in memory when open and can cause virtual storage constraint, prior to DB2 V8.



## 7.3 Index-use tracking by using real-time statistics

With DB2 9 for z/OS, the date when the index is last used for SELECT, FETCH, searched UPDATE, searched DELETE, or enforce referential integrity constraints with dynamic SQL is maintained in the RTS column SYSINDEXSPACESTATS.LASTUSED. As a reminder, if an index is used by a static SQL statement, it is displayed as BTYPE= ' I ' in SYSPACKDEP (package) or SYSPLANDEP (database request module (DBRM) bound directly into a plan).

## 7.4 Not logged table spaces

Logging can become a bottleneck for INSERT-intensive operations. In the context of a data warehouse environment, this can apply to the following scenarios:

- ▶ Creation of intermediate summary tables
- ▶ Insert processing in parallel
- ▶ Seasonal massive treatments
- ▶ Extract, load, and transform processes
- ▶ Massive batch updates due to confidentiality requirements (scrambling)

DB2 9 for z/OS introduces the notion of logging volume reduction capabilities by allowing logging to be suppressed for table spaces and associated indexes.

**Important:** Indiscriminate use of not logged operations can cause a loss of data. Use NOT LOGGED on an exception basis and generally as a last resort after checking that your system is optimally configured for high logging demand operations.

Before implementing not logged table spaces, consider the following elements:

- ▶ DB2 9 for z/OS has introduced the following additional logging improvements:
  - Log record sequence number (LRSN) increment wait reduction in data sharing environments
  - VSAM data striping and compression now available for archive logs
  - Dual buffering for archive log reads
  - Support for up to 4 GB archive logs
- ▶ The existence of faster devices, controllers, and paths associated with Modified Indirect Data Address Word (MIDAW) improves dramatically the logging capabilities of System z.  
*Refer to [How does the MIDAW Facility Improve the Performance of FICON Channels Using DB2 and other workloads?](#), REDP-4201, and [Disk Storage Access with DB2 for z/OS](#), REDP-4187, for more details about DB2 and I/O infrastructure*
- ▶ The LOAD utility can be used with the LOG NO option as a way of skipping logging. LOG NO is used in most of the cases in a data warehouse environment.
- ▶ If elapsed time is increased because of logging, VSAM data striping can provide relief for non-synchronous logging operations.
- ▶ By avoiding logging, RECOVER is not possible.
- ▶ DATA CAPTURE CHANGES is not compatible with the NOT LOGGED option.

Example 7-30 on page 122 shows an extract of the DB2 9 for z/OS SQL guide, indicating how you disable or enable logging for a table space.

*Example 7-30 Alter table space NOT LOGGED syntax*

```
>>-ALTER TABLESPACE--+-----+table-space-name----->
                           '-database-name.-'

>-----+LOGGED-----+----->
        '-NOT LOGGED-'
```

You also must consider the impact to your backup policy that is caused by the use of the NOT LOGGED option. Figure 7-7 highlights again the fact that a not logged operation should be exceptional and image copies before and after the exceptional processing are needed. With NOT LOGGED, if transactions are updating the database while the table is not logged, and a problem occurs in a table, then the table space is reset. You also need to start again from the copy. You relinquish the option of database recovery by replacing it with application recovery.

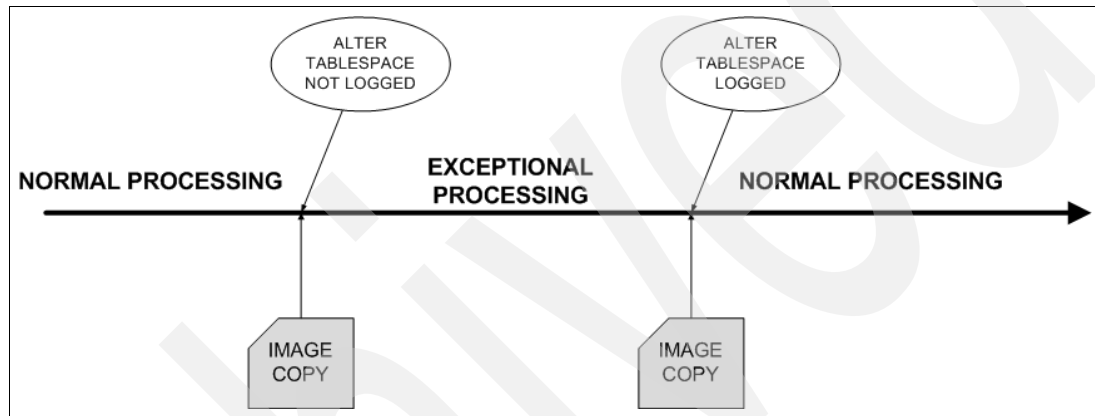


Figure 7-7 Exceptional processing and the NOT LOGGED option

Updating operations against a NOT LOGGED object puts the table space in RW,ICOPY status. For example, we execute the query in Example 7-31.

*Example 7-31 Sample query against a not logged table space*

```
UPDATE WORK.CUSTOMER_DIM
SET LAST_UPDATE = CURRENT_TIMESTAMP
```

Example 7-32 shows the output of the display table space after the query execution.

*Example 7-32 ICOPY status of a not logged table space after an update operation*

```
DSNT360I  -D912 *****
DSNT361I  -D912 *  DISPLAY DATABASE SUMMARY
              *  GLOBAL
DSNT360I  -D912 *****
DSNT362I  -D912      DATABASE = STARJOIN  STATUS = RW
              DBD LENGTH = 12104
DSNT397I  -D912
NAME      TYPE PART  STATUS              PHYERRLO PHYERRHI CATALOG  PIECE
-----
ORDERRTR TS    0001 RW,ICOPY
*****  DISPLAY OF DATABASE STARJOIN ENDED  *****
DSN9022I  -D912 DSNTDDIS 'DISPLAY DATABASE' NORMAL COMPLETION
```

ICOPY is an informational COPY pending advisory status and indicates that the affected object should be copied.

**Important:** ICOPY is not a restricted status, and a -DIS DB() SPACENAM() RESTRICT does not show any ICOPY table space. This technique cannot be used for detecting non-recoverable objects.

Unless defined as COPY NO, indexes are also in the ICOPY status after an update operation. An attempt of copying the index independently of the table space produces a DSNU449I error, as shown in Example 7-33.

*Example 7-33 Index space must be copied with not logged table spaces*

---

```
DSNU000I 116 10:20:34.55 DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = COPYX
DSNU1044I 116 10:20:34.58 DSNUGTIS - PROCESSING SYSIN AS EBCDIC
DSNU050I 116 10:20:34.58 DSNUGUTC - COPY INDEXSPACE STARJOIN.CUST18GD DSNUM ALL
COPYDDN(SYSCOPY)
DSNU449I -D912 116 10:20:34.58 DSNUBAII - THE NOT LOGGED INDEXSPACE STARJOIN.CUST18GD
IN ICOPY PENDING STATE MUST BE COPIED WITH ITS TABLESPACE IN THE SAME COPY
INVOCATION
DSNU012I 116 10:20:34.59 DSNUGBAC - UTILITY EXECUTION TERMINATED, HIGHEST RETURN CODE=8
```

---

A NOT LOGGED index that is in ICOPY pending state must be copied together with its table space in the same COPY invocation. An image copy of such an index made without its table space cannot be used for recovery of the index to a point that is consistent with its table space. The recovery requires applying log records, which were not written.

The load utility places a NOT LOGGED table space on RW,ICOPY status, even when using the NOCOPYPEND option as shown in Example 7-34.

*Example 7-34 Not logged table space and load nocopypend*

---

```
DSNU000I 116 08:55:20.79 DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = LOAD
DSNU050I 116 08:55:20.83 DSNUGUTC - LOAD INDDN SYSREC LOG NO RESUME YES NOCOPYPEND
DSNU650I -D912 116 08:55:20.83 DSNURWI - INTO TABLE "WORK"."CUSTOMER_DIM"
....
DSNU570I -D912 116 08:55:21.15 DSNUGSRX - TABLESPACE STARJOIN.CUSTOMER PARTITION 1 IS IN
INFORMATIONAL COPY PENDING
DSNU010I 116 08:55:21.15 DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```

---

## 7.5 Exploiting the DB2 STATEMENT CACHE

A typical data warehouse can run most of its reports by using dynamic SQL statements. To analyze the most intensive or expensive statements, the information in the DB2 STATEMENT cache can be used.

The IFCID 316, 317, 318 must be active in your system to exploit this feature. You can do this by selectively activating them with the trace statement:

```
-STA TRACE(P) CLASS(30) IFCID(316,317,318)
```

The sample member *HLQ.SDSNSAMP(DSNTESC)* contains the necessary DDL for the creation of a DB2 9 for z/OS STATEMENT CACHE table called *DSN\_STATEMENT\_CACHE\_TABLE*.

The SQL command in Example 7-35 copies the dynamic SQL statements from the DB2 cache to the table.

*Example 7-35 Populating the DSN\_STATEMENT\_CACHE\_TABLE*

```
EXPLAIN STMTCACHE ALL;
```

**Important:** The authorization rules are different if the STMTCACHE keyword is specified to have a cached statement explained. The privilege set must have SYSADM authority or the authority that is required to share the cached statement. For more information about the authority to use the dynamic statement cache, see the *DB2 Version 9.1 for z/OS Application Programming and SQL Guide*, SC18-9841.

After the DSN\_STATEMENT\_CACHE\_TABLE is populated, it can be queried by any method to identify the most consuming or more frequently executed queries. Among others, the following columns provide information that may be of interest in a data warehouse environment:

<b>STMT_ID</b>	The statement ID. This value is the EDM unique token for the statement.
<b>PROGRAM_NAME</b>	The name of the package or DBRM that performed the initial PREPARE for the statement.
<b>STAT_EXEC</b>	The number of times this statement has been run. For a statement with a cursor, this is the number of OPENs.
<b>STAT_GPAG</b>	The number of getpage operations that are performed for the statement.
<b>STAT_INDX</b>	The number of index scans that are performed for the statement.
<b>STAT_RSCN</b>	The number of table space scans that are performed for the statement.
<b>STAT_ELAP</b>	The accumulated elapsed time that is used for the statement.
<b>STAT_CPU</b>	The accumulated CPU time that is used for the statement.
<b>STAT_SUS_SYNO</b>	The accumulated wait time for synchronous I/O operations for the statement.
<b>STAT_RIDLIMT</b>	The number of times a record identifier (RID) list was not used because the number of RIDs might exceed DB2 limits.
<b>STAT_RIDSTOR</b>	The number of times a RID list was not used because not enough storage is available to hold the list of RIDs.

**Tip:** DB2 Optimization Service Center and DB2 Optimization Expert provide extensive support for the analysis of the information on the DB2 STATEMENT CACHE. Refer to *IBM DB2 9 for z/OS: New Tools for Query Optimization*, SG24-7421, for details.

## 7.6 Star schema processing

A *star schema* is a logical database design that is often found in a data warehouse implementation. It is composed of a fact table and a number of dimension tables that are connected or related to it. A *dimension table* contains several values that are given an ID, which is used in the fact table instead of all the values. The *fact table* is typically much larger

than the dimension tables. The graphical representation of the logical relationship among these tables resembles a star formation, as shown in Figure 7-8.

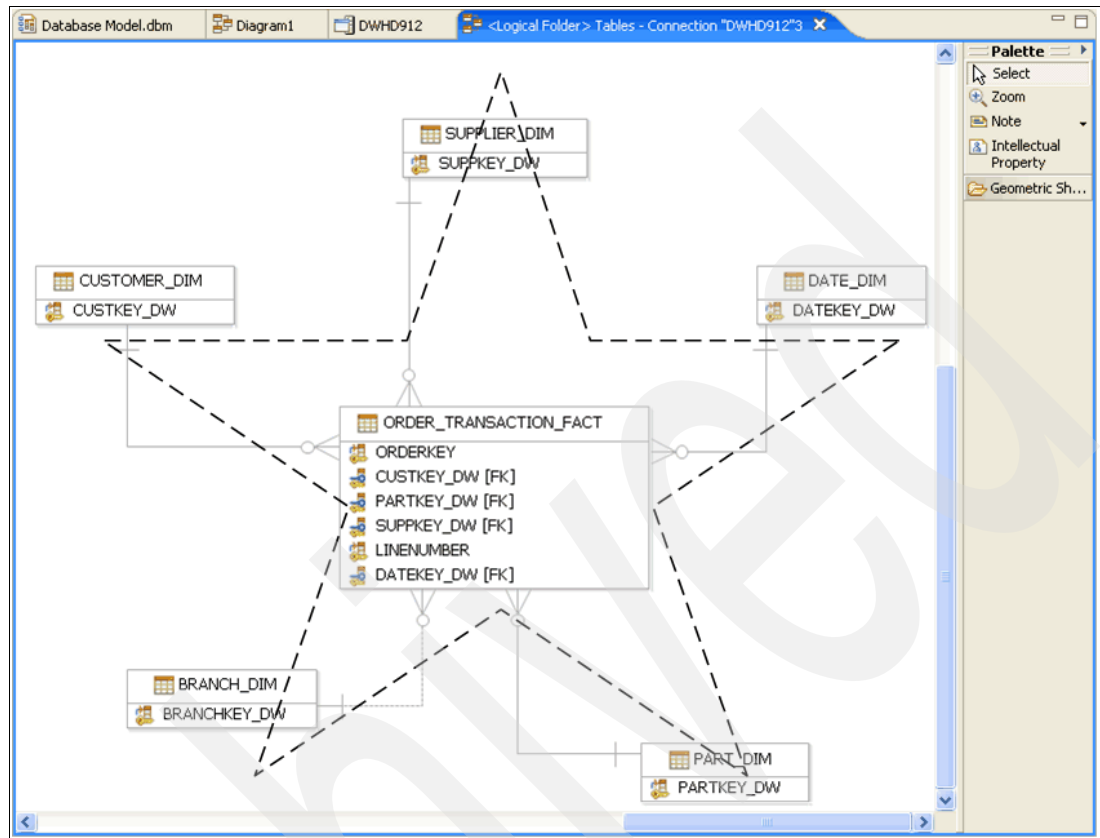


Figure 7-8 Star schema representation

A snowflake is an extension of a star schema, where a dimension table is normalized into several tables and only one of the tables in the snowflake joins directly with the fact table.

DB2 for z/OS can use special access methods, called a star join and pair-wise join, to solve queries against a star schema:

- ▶ A *star join* (JOIN\_TYPE = 'S' in PLAN\_TABLE) consolidates the dimensions to form a Cartesian product for matching to a single fact table index. This access method requires a single multicolumn index in the fact table.
- ▶ A *pair-wise join* (JOIN\_TYPE = 'P' in PLAN\_TABLE) considers each dimension as an independent object to match the fact table on corresponding single column indexes. This access method requires an index in the fact table for each dimension key column.

In this section, we discuss these methods and the changes that are introduced in DB2 9 to improve performance of star schema-related queries.

### 7.6.1 Star schema access methods

Join methods, such as nested-loop join, merge-scan join, and hybrid join, involve only two tables in each step. A single step in a star join can involve three or more tables to consolidate filtering before accessing the large fact table. For this reason, a star join can be a more efficient choice for processing a star schema than traditional join methods. The optimizer

considers star schema access methods based on costs, and a star join method is not necessarily always used.

To have a query be a candidate or eligible for a star join, the following conditions are required:

- ▶ The query must respect some SQL restrictions.
- ▶ Two specific DSNZPARMs must be setup accordingly: STARJOIN and SJTABLES.
- ▶ Proper index design must be present in the star schema tables.

These requirements are described in the following paragraphs.

## SQL restrictions for star join

In order for a query to be a candidate for star schema processing, the following conditions must be met:

- ▶ The query references at least two dimensions.
- ▶ All join predicates are between the fact table and the dimension tables, or within tables of the same snowflake. If a snowflake is connected to the fact table, only one table in the snowflake (the central dimension table) can be joined to the fact table.
- ▶ All join predicates between the fact table and dimension tables are equi-join predicates.
- ▶ All join predicates between the fact table and dimension tables are Boolean term predicates. A *Boolean term predicate* is a simple or compound predicate that, when it is evaluated false for a particular row, makes the entire WHERE clause false for that particular row.
- ▶ If predicates contain OR conditions that involve multiple tables, then at least one common table is referenced in each OR branch.
- ▶ There are no correlated subqueries across dimensions.
- ▶ After DB2 simplifies join operations, no outer join operations exist.
- ▶ The data type and length of both sides of a join predicate are the same.

The query shown in Example 7-36 is based on our data model. This query is an example of a candidate for star schema processing.

*Example 7-36 Sample query candidate for star join*

---

```
SELECT
    SUM(FACT.QUANTITY)
FROM
    WORK.ORDER_TRANSACTION_FACT FACT,
    WORK.DATE_DIM DATE,
    WORK.BRANCH_DIM BRANCH,
    WORK.CUSTOMER_DIM CUST,
    WORK.PART_DIM PART,
    WORK.SUPPLIER_DIM SUPP
WHERE
    FACT.CUSTKEY_DW = CUST.CUSTKEY_DW
    AND FACT.BRANCHKEY_DW = BRANCH.BRANCHKEY_DW
    AND FACT.DATEKEY_DW = DATE.DATEKEY_DW
    AND FACT.PARTKEY_DW = PART.PARTKEY_DW
    AND FACT.SUPPKEY_DW = SUPP.SUPPKEY_DW
    AND BRANCH.BRANCHKEY_DW = 1
    AND DATE.DATE_VAL = '2007-01-01 '
    AND CUST.CUST_REGION_NAME = 'EUROPE'
```

---

## System parameters for star schema processing

To have star schema processing considered, the following DSNZPARMs must be correctly set up:

### ► STARJOIN

It specifies whether star join processing is enabled. It can have the following values:

- **DISABLE**: Star schema processing is disabled. This is the default value.
- **1**: Star schema processing is **ENABLED**. The table with the largest cardinality is considered as the fact table. However, if more than one table has this cardinality, the star join is not enabled.
- **ENABLE**: Star schema processing is enabled if the cardinality of the fact table is at least 25 times the cardinality of the largest dimension that is a base table that is joined to the fact table.
- **2 to 32768**: Star schema processing is enabled if the cardinality of the fact table is at least *n* times the cardinality of the largest dimension that is a base table that is joined to the fact table.

### ► SJTABLES

This value indicates the minimum number of tables in the star schema query block, including the fact table, dimensions tables, and snowflake tables that are needed for enabling star schema processing.

These two parameters can be changed online via the **-SET SYSPARM** command.

The following parameters are other star schema-related DSNZPARM parameters:

### ► MXQBCE

MXQBCE limits how many different join sequences the DB2 optimizer will consider. The lower the number is, the fewer the number of join sequences are considered. By reducing the number of join sequences that are considered, you can reduce the time that DB2 spends in bind processing. Of course, spending less time can also mean that a less appropriate access path might be chosen.

### ► SJMISSKY

By specifying SJMISSKY = ON, DB2 has a better chance to choose a more efficient access path for star join queries when the star join outside-in phase involves missing keys, the dimension table is not highly normalized, or both.

Example 7-37 shows an Optimization Service Center report of the default values for the system parameters STARJOIN and SJTABLES on a DB2 9 for z/OS subsystem.

#### *Example 7-37 Optimization Service Center report - Default values for STARJOIN and SJTABLES*

---

\*\*\*\*Optimization Service Center Parameter Browser \*\*\*\*

Parameter: STARJOIN Macro: DSN6SPRM Installation panel: DSNTIP8

Value: DISABLE

Description: STARJOIN- Disable (default) = no star join. Enable = enable star join. 1 = The fact table will be the largest table in the star join query. No fact/dimension ratio checking. 2-32768 = Star join fact table and the largest dimension table ratio.

\*\*\*\*Optimization Service Center Parameter Browser \*\*\*\*

Parameter: SJTABLES Macro: DSN6SPRM Installation panel: Installation field:

Value: 00010

Description: SJTABLES- Effective only when star join is enabled (see starjoin parm, below). Sets the total number of tables in a query block as the threshold to enable star join for the query block. Valid settings range from 0-32767, as follows: 0: indicates that the

default value is used, that is, star join is considered for a query block having 10 or more tables. 1,2,3: Star join is always considered. 4-225: Star join is considered for a query block having at least the specified number of tables 226-32767: star join will never be enabled.

---

Follow these general guidelines for setting the value of SJTABLES and STARJOIN:

- ▶ If you have queries that reference fewer than 10 tables in a star schema database and you want to make the star schema methods applicable to all qualified queries, set the value of SJTABLES to the minimum number of tables used in queries that you want to be considered for star schema processing.
- ▶ If you want to use star schema processing for relatively large queries that reference a star schema database but are not necessarily suitable for star join or pair-wise join, use the default. The star schema methods are considered for all qualified queries that have 10 or more tables.
- ▶ If you have queries that reference a star schema database but, in general, do not want to use star schema processing, consider setting STARJOIN to DISABLE to disable the star schema join methods from being considered.

You have to consider that, even if star schema processing can reduce bind time significantly, when the query contains a large number of tables, it does not provide optimal performance in all cases.

The performance of a star join and pair-wise join depends on several factors, such as the available indexes on the fact table, the cluster ratio of the indexes, and the selectivity of rows through local and join predicates. You might consider a strategy to quickly disable star schema processing in your system in case of performance issues, for example, changing online the related DSNZPARMs to its default values (DISABLED).

## Index design for star schema processing

Star join and pair-wise join are index-based access methods. Proper indexes are necessary to have these techniques considered by the optimizer.

To design indexes to enable pair-wise join, create an index for each key column in the fact table that corresponds to a dimension table. For further performance improvement for some star schema queries, consider the following index design recommendations to encourage DB2 to use star join access:

- ▶ Define a multicolumn index on all key columns of the fact table. *Key columns* are fact table columns that have corresponding dimension tables.
- ▶ If you do not have information about the way that your data is used, first try a multicolumn index on the fact table that is based on the cardinality of the key columns. Order the index key columns from lowest to highest cardinality.

Define this multicolumn index as clustering on the fact table.

- ▶ Define indexes on dimension tables to improve access to those tables.
- ▶ When you have executed a number of queries and have more information about the way that data is used, follow these recommendations:
  - Place more selective columns at the beginning of the multicolumn index.
  - If a number of queries do not reference a dimension, place the column that corresponds to that dimension at the end of the index, or remove it completely.



## 7.6.2 Star schema processing implementation example

To show the steps for implementing and monitoring star schema processing, a query was executed against the star schema that we implemented in our scenario.

Figure 7-9 shows the star schema representation of the data model that is used in the following examples. During this implementation, the fact table, which is represented in the middle of the figure, contains at least 35 times more rows than any of the dimensions tables. No referential integrity is established among the tables.

The optimizer determines parent child relationships by the existence of unique indexes on one side of the join, implying that this is the parent. If the optimizer cannot decide which one of the tables is the fact table based on this, it refers to the cardinality of the objects.

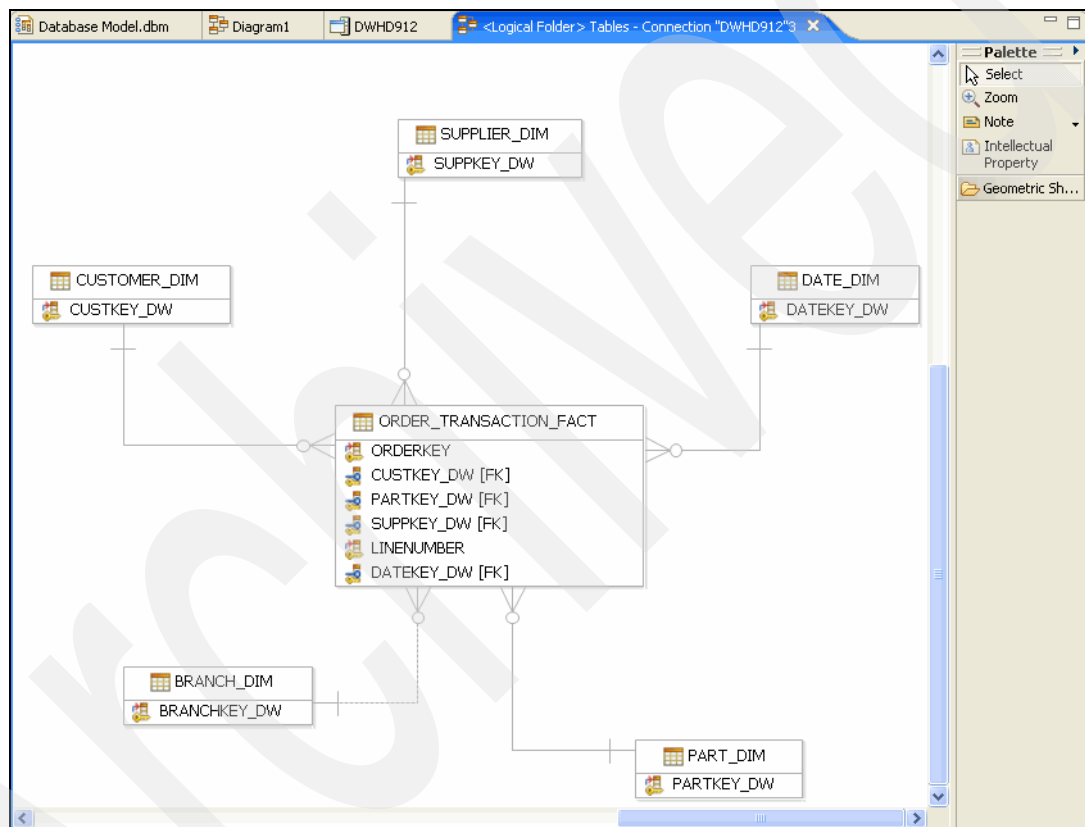


Figure 7-9 Star schema implementation sample

In our example, the star schema is built as follows:

- ▶ Fact table, named **ORDER\_TRANSACTION\_FACT**, contains additive information about orders and line items.
- ▶ Dimension tables contain descriptive information:
  - **DATE\_DIM**
  - **BRANCH\_DIM**
  - **CUSTOMER\_DIM**
  - **PART\_DIM**
  - **SUPPLIER\_DIM**

We used the query shown in Example 7-38 as a candidate for star join access methods.

*Example 7-38 Sample query candidate for a star join*

---

```
SELECT
    SUM(FACT.QUANTITY)
FROM
    WORK.ORDER_TRANSACTION_FACT FACT,
    WORK.DATE_DIM DATE,
    WORK.BRANCH_DIM BRANCH,
    WORK.CUSTOMER_DIM CUST,
    WORK.PART_DIM PART,
    WORK.SUPPLIER_DIM SUPP
WHERE
    FACT.CUSTKEY_DW      = CUST.CUSTKEY_DW
    AND FACT.BRANCHKEY_DW = BRANCH.BRANCHKEY_DW
    AND FACT.DATEKEY_DW  = DATE.DATEKEY_DW
    AND FACT.PARTKEY_DW  = PART.PARTKEY_DW
    AND FACT.SUPPKEY_DW  = SUPP.SUPPKEY_DW
    AND BRANCH.BRANCHKEY_DW = 1
    AND DATE.DATE_VAL    = '2007-01-01 '
    AND CUST.CUST_REGION_NAME = 'EUROPE'
```

---

## Implementing indexes

The following indexes were created before activating star schema processing in our DB2 9 for z/OS subsystem:

- A multicolumn index on all key columns of the fact table

In our example, it includes five columns from the five dimension tables. The DDL is described in Example 7-39. This index can help the optimizer to select a star join access method for eligible queries.

*Example 7-39 Creating a multicolumn index in the fact table for star join processing*

---

```
CREATE INDEX WORK.SJOIN_ALL_KEY
ON WORK.ORDER_TRANSACTION_FACT
(DATEKEY_DW      ASC,
 BRANCHKEY_DW   ASC,
 CUSTKEY_DW      ASC,
 PARTKEY_DW      ASC,
 SUPPKEY_DW      ASC);
```

---

In our test scenario, the index described in Example 7-39 must be defined as CLUSTER in order to be considered by DB2 for star join.

- An index for each key column in the fact table that corresponds to a dimension table

Indexes that contain the same keys were created in the dimension tables. Example 7-40 shows a DDL sample for the creation of these indexes on the fact table. They can support the pair-wise access method.

**Tip:** To determine the best order of columns in an index for a star schema, refer to the method described in Chapter 27, “Tuning your queries” in the DB2 Version 9.1 of the *DB2 Version 9.1 for z/OS Application Programming and SQL Guide*, SC18-9841.

*Example 7-40 Defining indexes for each dimension key column in the fact table*

---

```
CREATE INDEX WORK.SJOIN_DIM_X1
  ON WORK.ORDER_TRANSACTION_FACT
  (DATEKEY_DW          ASC);

CREATE INDEX WORK.SJOIN_DIM_X2
  ON WORK.ORDER_TRANSACTION_FACT
  (BRANCHKEY_DW ASC);

CREATE INDEX WORK.SJOIN_DIM_X3
  ON WORK.ORDER_TRANSACTION_FACT
  (CUSTKEY_DW ASC);

CREATE INDEX WORK.SJOIN_DIM_X4
  ON WORK.ORDER_TRANSACTION_FACT
  (PARTKEY_DW ASC);

CREATE INDEX WORK.SJOIN_DIM_X5
  ON WORK.ORDER_TRANSACTION_FACT
  (SUPPKEY_DW ASC);
```

---

**Tip:** Partitioning cluster data according to commonly used dimension keys can reduce the I/O that is required on the fact table for a pair-wise join.

To better appreciate the impact of the star join, we used Optimization Service Center to graphically represent the access paths that are chosen by the optimizer during the execution of our tests. Figure 7-10 on page 132 shows the original access path for the query in Example 7-38 on page 130.

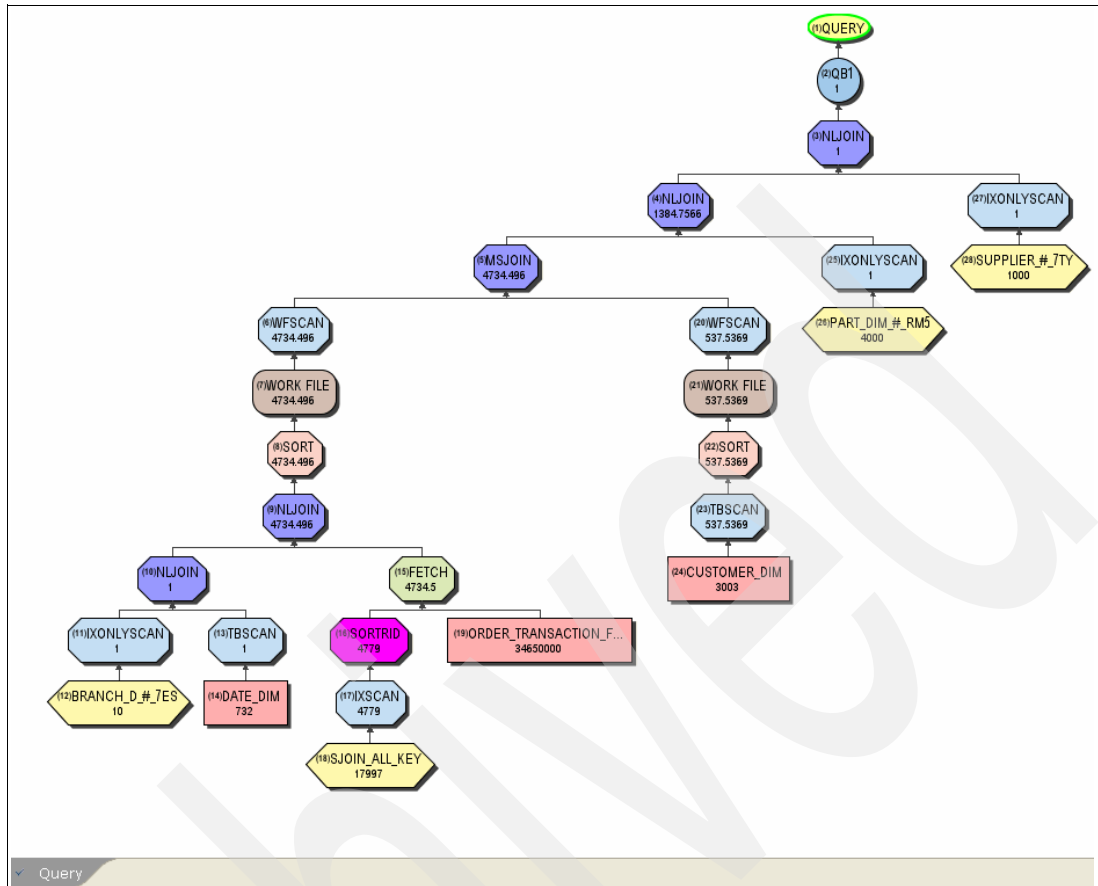


Figure 7-10 Query access path before activating star schema processing

Table 7-3 shows an extract of the PLAN\_TABLE for this query, including the columns of interest for this analysis.

Table 7-3 Sample PLAN\_TABLE extract before activating star schema processing

PLANNO	METHOD	TNAME	JOIN TYPE	SORTN JOIN	ACCESS TYPE	PRIMARY ACCESS TYPE
1	0	BRANCH_DIM		N	I	
2	1	DATE_DIM		N	R	
3	1	ORDER_TRANSACTION_FACT		N	I	
4	2	CUSTOMER_DIM		Y	R	
5	1	PART_DIM		N	I	
6	1	SUPPLIER_DIM		N	I	

The optimizer uses the SJOIN\_ALL\_KEY index to access the table ORDER\_TRANSACTION\_FACT, which is the fact table in our example, with an index scan. The resolution of this query involves six steps executed in series.

The column JOIN\_TYPE indicates the type of join. It has the following possible values:

- F** Full outer join
- L** Left outer join (RIGHT OUTER JOIN converts to a LEFT OUTER JOIN)
- S** Star join
- Blank** INNER JOIN or no join

PRIMARY\_ACCESSTYPE provides indication of whether direct row access is attempted first. It has the following possible values:

- D** DB2 tries to use direct row access with a rowid column.
- T** The base table or result file is materialized into a work file, and the work file is accessed via sparse index access. If a base table is involved, then ACCESSTYPE indicates how the base table is accessed.

**Blank** DB2 does not try to use direct row access by using a rowid column or sparse index access for a work file.

## System parameters

To activate star schema processing, we changed the default DSNZPARMs as shown in Example 7-41.

*Example 7-41 System parameters implementation for star join*

---

```
STARJOIN=1
SJTABLES=3
```

---

Changing this setting has the following effects:

- ▶ STARJOIN=1 enables star schema processing, and the optimizer considers the table with the highest cardinality as the fact table. In our example, the table ORDER\_TRANSACTION\_FACT is, by far, bigger than any of the dimension tables.
- ▶ SJTABLES=3 indicates that star schema processing will be considered when at least three tables, including the fact table, are included in a query block candidate to star join. This value is acceptable for our purposes, but you can find it too aggressive because star join or pair-wise access methods do not always provide the better performance. In mixed workloads, you might consider using a higher value for your installation.

Because these parameters can be changed online, we assembled a specific DSNZPARM load module to activate and deactivate these two parameters as desired, as shown in Example 7-42.

*Example 7-42 Online activation of star schema processing parameters*

---

```
-D912 SET SYSPARM LOAD(STARPARM)
DSN9022I -D912 DSNZCMD0 'SET SYSPARM' NORMAL COMPLETION
DSNZ006I -D912 DSNZCMD1 SUBSYS D912 SYSTEM PARAMETERS LOAD MODULE NAME STARPARM
IS BEING LOADED
DSNZ007I -D912 DSNZCMD1 SUBSYS D912 SYSTEM PARAMETERS LOAD MODULE NAME STARPARM
LOAD COMPLETE
```

---

During our testing, we were able to fall back from these changes, that is to deactivate these parameters, by reloading the startup DSNZPARM as shown in Example 7-43 on page 134. You might consider this as a quick method to fix a performance issue related to star schema processing.

---

*Example 7-43 Deactivating star schema processing parameters by reloading startup zparms*

---

```
-D912 SET SYSPARM STARTUP
DSN9022I  -D912 DSNZCMD0 'SET SYSPARM' NORMAL COMPLETION
DSNZ011I  -D912 DSNZCMD1 SUBSYS D912 SYSTEM PARAMETERS SET TO STARTUP
```

---

We used Optimization Service Center to verify the changes on these system parameters, as shown in Example 7-44.

---

*Example 7-44 Optimization Service Center report on changed values for STARJOIN, SJTABLES*

---

```
****Optimization Service Center Parameter Browser ****
Parameter: SJTABLES
Value: 00003

****Optimization Service Center Parameter Browser ****
Parameter: STARJOIN
Value: 00001
```

---

### **Observed changes on access path**

After verification of the system parameters changes, the query shown in Example 7-38 on page 130 is explained again. The changes on access path as reported by the Optimization Service Center are shown in Figure 7-11 on page 135. We numbered some of the elements of this graph that are the indication of a star join processing being implemented:

- Point 1** STARJOIN node. This node indicates the execution of a star join operation.
- Point 2** Star join dimension work file node. A star join dimension work file node represents a work file that results from the sorting of a dimension table or another star join dimension work file during a data manager pushdown star join.
- Point 3** This part of the graph indicates that the dimension table BRANCH\_DIM is accessed by using one of its indexes.
- Point 4** This part shows that the access to dimension tables, for local qualification or rows, is not exclusively done by index access. The table CUSTOMER\_DIM is read by a table space scan. We might consider adding a proper index to improve performance.

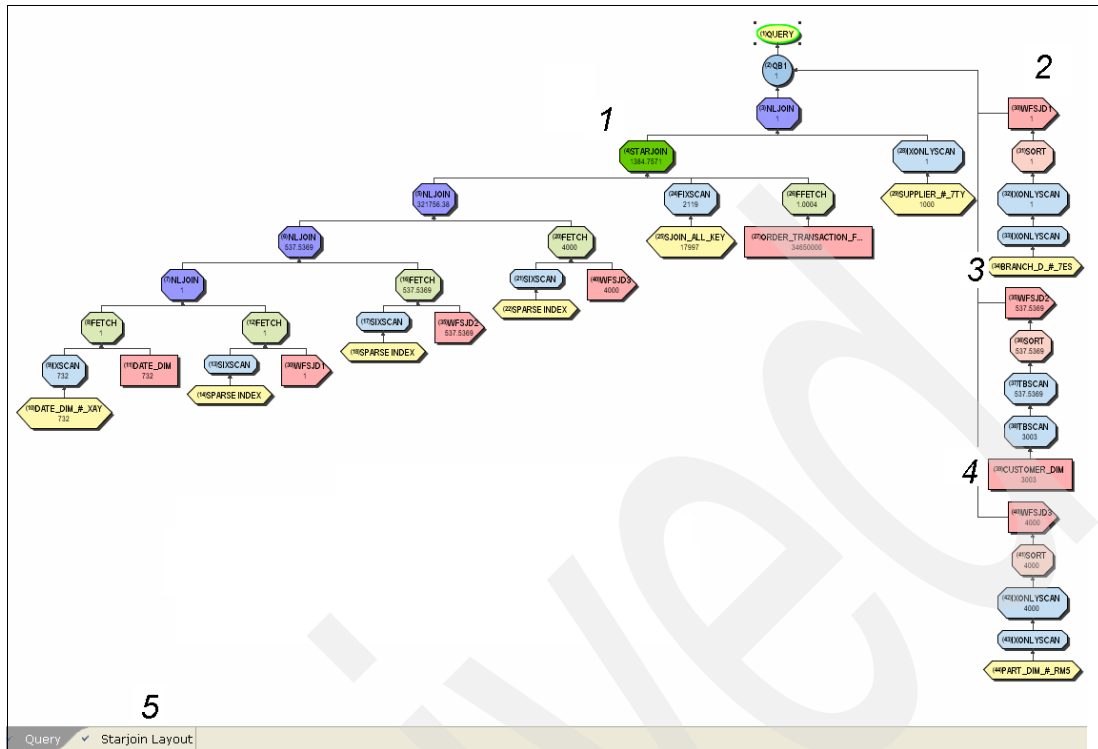


Figure 7-11 Access path changes after enabling a star join

**Point 5** A new section is activated in the Optimization Service Center access path graph when a star join is involved, star join layout. This section is shown in Figure 7-12.

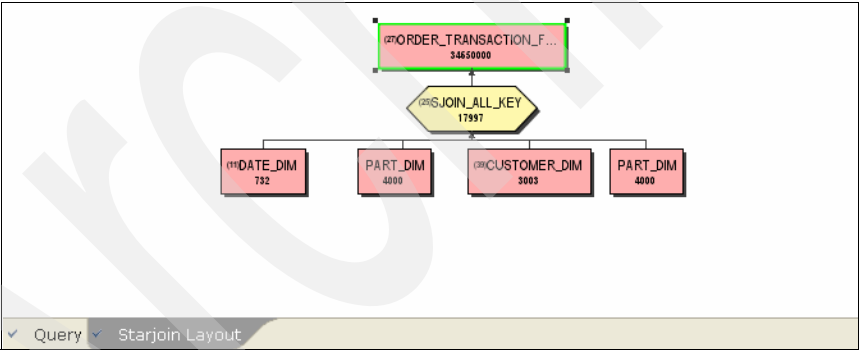


Figure 7-12 Star join layout in Optimization Service Center

Table 7-4 shows relevant PLAN\_TABLE information for this explanation.

Table 7-4 Sample PLAN\_TABLE extract after activating star schema processing

PLANNO	METHOD	TNAME	JOIN TYPE	SORTN JOIN	ACCESS TYPE	PRIMARY ACCESS TYPE
1	0	DATE_DIM	S	N	I	
2	1	BRANCH_DIM	S	Y	I	T
3	1	CUSTOMER_DIM	S	Y	R	T
4	1	PART_DIM	S	Y	I	T

PLANNO	METHOD	TNAME	JOIN TYPE	SORTN JOIN	ACCESS TYPE	PRIMARY ACCESS TYPE
5	1	ORDER_TRANSACTION_FACT	S	N	I	
6	1	SUPPLIER_DIM		N	I	

The PLAN\_TABLE shows JOIN\_TYPE='S' for this query, meaning that it chooses to execute a star join. It will access four of five dimensions before accessing the fact table, as shown in Figure 7-12 on page 135. This decision is based on costs. The optimizer can select another strategy involving one, two, or all dimensions before accessing the fact table.

The value PRIMARYACCESS\_TYPE = 'T' in the PLAN\_TABLE indicates that there is materialization into a work file, and the work file is accessed via sparse index access. Figure 7-13 shows a graphical representation of this work file and the use of a sparse index.

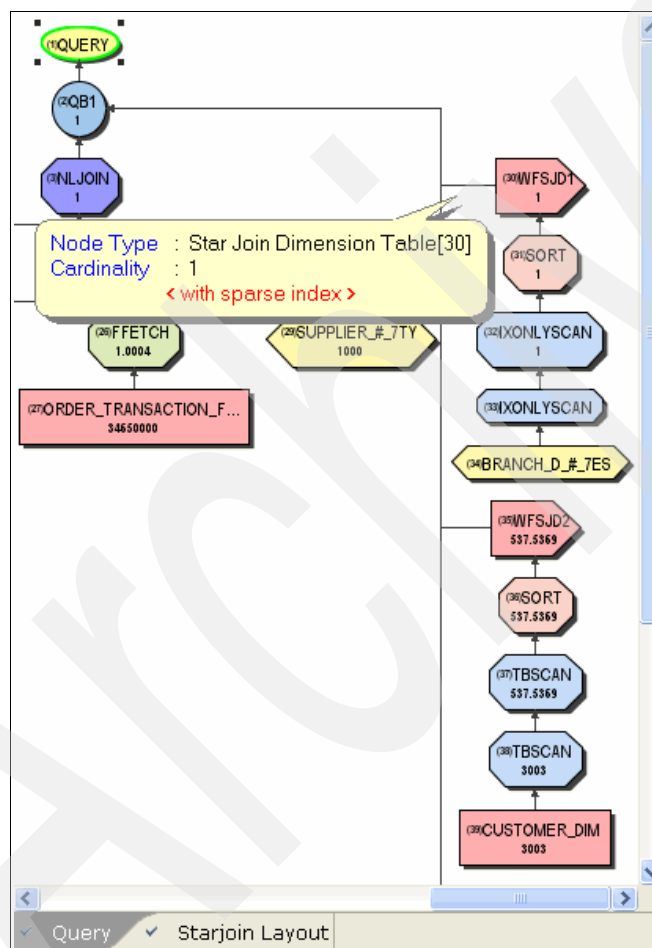


Figure 7-13 Optimization Service Center star join dimension node type

The optimizer chooses a fact table clustering index that qualifies the most number of join columns. In this example, it is the index that is created in Example 7-39. Based on this index, DB2 performs a Cartesian join on the corresponding dimension tables in the index key order. For each record in the resulting virtual table, DB2 uses a matching index scan on the fact table to find qualifying records. During run time, DB2 does not create the virtual table. However, it sorts each dimension table in the fact table index key order, and uses a variant of a nested loop



join to join them together. DB2 also uses a special technique, called *index skipping*, to further reduce the number of rows that it needs to retrieve from the dimension tables.

This star schema does not contain snowflakes. If they exist, they are processed before the central part of the star join as individual query blocks and are materialized into work files, one for each snowflake.

During our tests, we provided the optimizer the choice of having twice the same index described in Example 7-39. It is the multicolumn index that is created to support the star join. One of the indexes was compressed. Example 7-45 shows some of its characteristics.

*Example 7-45 Compression on a star schema index*

IDX	COMP	COLS	AVGKEYL	PGSIZE	SPACE	NLEAF	NLEVELS
SJOIN_ALL_KEY	N	5	20	4	234000	57870	4
SJOIN_ALL_KEY_COMP	Y	5	20	8	151200	34687	4

Index compression can be a good idea for this kind of index because they can be big specially when the star schema contains a large number of dimension keys. In our tests, we observed that the optimizer takes the compressed version of the index, instead of the non-compressed one, which is the opposite of what we observed during non-star join tests. The assumption is that the optimizer is more favorable to compressed versions of the same index when it is deployed in a star join context, but you have to validate this observation on your own environment.

## Dynamic index ANDing

Dynamic index ANDing (also known as *pair-wise join*) is one of the new features on DB2 9 for z/OS for data warehousing. It provides a simpler solution to previous index design challenges for star schemas.

With star schema, the filtering is done generally on the dimension tables and not on the fact table. These dimensions are generally independent. Creating fact table indexes to support the various combination of filtering dimensions was one of the challenges of a star schema before the introduction of dynamic index ANDing.

Dynamic index ANDing only requires a single column index for each filtering dimension on the fact table. Each filtering dimension is accessed independently and in parallel. Each dimension is joined with the fact table indexes independently, building separate RID lists.

During run time, each of these parallel processes is checked for completion. In case a long running process is still executing after all the others have finished, DB2 can decide to terminate its processing to avoid waiting until it ends. This provides the advantage of being able to recover, at run time, out of a less than perfect access path choice made at bind or prepare time.

The obtained RIDs are then combined and intersected before accessing the fact table. This ability of dynamically ANDing RIDs avoids the need of defining fact table indexes with the combination of the dimension keys.

A RID pool failure during RID list processing causes a fall back to the table space scan in DB2 for z/OS V8, which may have huge impacts on performance. In DB2 9 for z/OS, within the dynamic index ANDing access method, a RID pool failure does not fall back to a table space scan. The current RID list is written into a work file, and processing continues by writing into that work file.

In summary, dynamic index ANDing provides the following improvements:

- ▶ Pre-fact table filtering to filter dimensions that are accessed concurrently
- ▶ Runtime optimization, which terminates poorly filtering processing on dimension tables at run time
- ▶ More aggressive parallelism
- ▶ Fallback to a work file for RID pool failure instead of table space scan

## Enabling data caching for star schema queries

You can enable data caching to improve the performance of queries on star schemas. When data caching is enabled for star schemas, DB2 caches data from work files that are used by star schema queries.

Data caching provides the following advantages:

- ▶ Immediate data availability  
During a star join operation, work files might be scanned many times. If the work file data is cached in the dedicated virtual memory pool, that data is immediately available for join operations.
- ▶ Reduced buffer pool contention  
Because the virtual memory space for data caching is separated from the work file buffer pool, contention with the buffer pool is reduced. Reduced contention improves performance particularly when sort operations are performed concurrently.

**Tip:** With DB2 9 for z/OS, a local pool above the 2 GB bar is used instead of a global pool. DSNZPARM MXDTCACH specifies the maximum size in MB (default of 20 MB) of virtual memory for data caching each thread. To set the pool size, use the MXDTCACH DSNTIP8 installation panel. To determine the best setting for the size of this pool, refer to the *DB2 Version 9.1 for z/OS Performance Monitoring and Tuning Guide*, SC18-9851.

If the dedicated memory pool is not large enough to contain the materialized work file, DB2 uses a sparse index on the work file instead.

OMEGAMON XE for DB2 Performance Expert on z/OS, in combination with IFCID 002, can be used to monitor this pool performance, as shown in Example 7-46.

*Example 7-46 DB2 OMEGAMON/PE Record trace - IFCID 002 - STARJOIN pool statistics*

---

STARJOIN POOL STATISTICS	
CURRENT POOL SIZE (MB)	102
POOL FULL FAILURES	101
ALLOCATION REQUESTS	100
MAX POOL SIZE (MB)	103

---

In DB2 9 for z/OS, in-memory data caching is extended to joins other than the star join. Since DB2 9 uses a local pool above the bar instead of a global pool, data caching storage management is associated with each thread and, therefore, potentially reduces storage contention.

## 7.7 Index on expressions

DB2 9 for z/OS introduces indexes on expressions. This new feature can enhance performance for queries if the optimizer decides to use the index on expression.

In contrast to simple indexes, where index keys consist of a concatenation of one or more table columns, the index key values on this new kind of index are not the same as the values in the table columns. The values have been transformed or precalculated by the specified expressions called *key expressions*. An index can mix regular columns and key expressions.

**Important:** One of the most important applications of index on expression is the ability to make stage 2 predicates indexable, providing potentially large performance improvements.

The key expression specifies an expression that returns a scalar value and cannot be specified with the GENERATE KEY USING clause. Each key expression must contain at least one reference to a column.

A key expression cannot contain the following items:

- ▶ A subquery
- ▶ An aggregate function
- ▶ An undeterministic function
- ▶ A function that has an external action
- ▶ A user-defined function
- ▶ A sequence reference
- ▶ A host variable
- ▶ A parameter marker
- ▶ A special register
- ▶ A CASE expression
- ▶ An online analytical processing (OLAP) specification

Example 7-47 shows an extract of the DB2 V9 for z/OS SQL reference, which includes how to integrate the *expression* into the index definition.

*Example 7-47 Index creation DDL - Index on expression*

```
>>--CREATE--+-----+--INDEX--index-name---->
          '-UNIQUE--+-----+
>--ON----->
>--+table-name--(-----+column-name-----+-----+-----+-----+>
|               '-key expression-'  +-DESC---+
|               '-RANDOM-'
| -aux-table-name-----|
```

Keep in mind the following design considerations and limitations:

- ▶ An index on expression cannot be a clustering index.
- ▶ If the index is created by using an expression, EXECUTE privilege is required on any user-defined function that is invoked in the index expression.
- ▶ All references to columns of table name must be unqualified. Referenced columns cannot be large object (LOB), XML, or DECFLOAT data types or a distinct type that is based on one of these data types. Referenced columns cannot include any FIELDPROCs or a SECURITY LABEL.

- ▶ If key expression references a cast function, the privilege set must implicitly include EXECUTE authority on the generated cast functions for the distinct type.
- ▶ If key expression references the LOWER or UPPER functions, the input string expression cannot be FOR BIT DATA, and the function invocation must contain the locale name argument.
- ▶ The maximum length of the text string of each key expression is 4000 bytes after conversion to UTF-8. The maximum number of key expressions in an extended index is 64.

Refer to DB2 documentation for further details about limitations and restrictions for index on expressions.

### Design example

Use index on expression when you want an efficient evaluation of queries with a column expression. You can consider the creation of an index on expression to improve performance. The precalculated values are stored on the index. This saves the runtime evaluation of the expression. It can also provide index only access paths that are not possible otherwise.

The query in Example 7-48 is considered stage 2 and cannot use the index OLTP.CUSTOMER\_X1 because of the UPPER function.

#### *Example 7-48 Stage 2 predicate query example*

---

```
CREATE TABLE OLTP.CUSTOMER
(  CUST_KEY INTEGER NOT NULL,
   NAME VARCHAR(25),
   ADDRESS VARCHAR(40),
   COUNTRY CHAR(25),
   PHONE CHAR(15),
   ACCTBAL DECIMAL(12,2),
   MKTSEGMENT CHAR(10)
);

CREATE INDEX OLTP.CUSTOMER_X1
ON
OLTP.CUSTOMER
(NAME, ACCTBAL);

SELECT
ACCTBAL
FROM OLTP.CUSTOMER
WHERE UPPER(NAME) = 'BOEHLER'
;
```

---

Example 7-49 shows how to implement a simple index on expression that makes the predicate stage 1 and indexable.

#### *Example 7-49 Index on expression creation example*

---

```
CREATE INDEX OLTP.CUSTOMER_XE1
ON
OLTP.CUSTOMER
(UPPER(NAME), ACCTBAL);
```

---

All the information needed for the key expression evaluation must be contained in a single row.

Example 7-50 shows a DDL for index creation failing with return code -120, which means that an aggregate function cannot be used for a key expression.

*Example 7-50 Index on expression cannot contain aggregate functions*

---

```
CREATE INDEX
OLTP.CUSTOMER_XE2
ON
OLTP.CUSTOMER
(SUM(ACCTBAL));
-----+-----+-----+-----+-----+-----+-----+
DSNT408I  SQLCODE = -120, ERROR:  AN AGGREGATE FUNCTION IS NOT VALID IN THE
        CONTEXT IN WHICH IT WAS INVOKED
-----+-----+-----+-----+-----+-----+-----+

```

---

Example 7-51 shows the successful creation of an index including a multiplication expression. SQLCODE 807 is a warning message that indicates that the runtime evaluation of the expression can cause overflow. This message does not prevent index creation.

*Example 7-51 Creating an index on expression*

---

```
CREATE INDEX OLTP.CUSTOMER_XE3
ON OLTP.CUSTOMER
(ACCTBAL*1000);
-----+-----+-----+-----+-----+-----+-----+
DSNT404I  SQLCODE = 807, WARNING:  THE RESULT OF DECIMAL MULTIPLICATION MAY
        CAUSE OVERFLOW
-----+-----+-----+-----+-----+-----+-----+
DSNE616I  STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+-----+-----+

```

---

Some changes were introduced in the DB2 catalog to support index on expression. Example 7-52 shows an extract of the RUNSTATS utility executed after the index creation. It includes the update of the related new catalog tables SYSKEYTARGETS and SYSKEYTGTDIST. Notice that SYSCOLUMNS is not updated.

*Example 7-52 Runstats output for an index on expression*

---

```
DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = CRIS.CRISD
DSNUGUTC - RUNSTATS TABLESPACE STARJOIN.LINEITEM TABLE(ALL) INDEX(ALL)
DSNUSUTP - SYSTABLEPART CATALOG UPDATE FOR STARJOIN.LINEITEM SUCCESSFUL
.....
DSNUSUIP - SYSINDEXPART CATALOG UPDATE FOR WORK.LINEITEM_X1 SUCCESSFUL
DSNUSUIP - SYSINDEXPART CATALOG UPDATE FOR WORK.LINEITEM_XEXP_01 SUCCESSFUL
DSNUSUCO - SYSCOLUMNS CATALOG UPDATE FOR WORK.LINEITEM_X1 SUCCESSFUL
DSNUSUIX - SYSINDEXES CATALOG UPDATE FOR WORK.LINEITEM_X1 SUCCESSFUL
DSNUSUKT - SYSKEYTARGETS CATALOG UPDATE FOR WORK.LINEITEM_XEXP_01 SUCCESSFUL
DSNUSUIX - SYSINDEXES CATALOG UPDATE FOR WORK.LINEITEM_XEXP_01 SUCCESSFUL
DSNUSUCD - SYSCOLDIST CATALOG UPDATE FOR WORK.LINEITEM_X1 SUCCESSFUL
DSNUSUKD - SYSKEYTGTDIST CATALOG UPDATE FOR WORK.LINEITEM_XEXP_01 SUCCESSFUL
DSNUSEOF - RUNSTATS CATALOG TIMESTAMP = 2008-04-25-15.52.50.656740
DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0

```

---

The new catalog tables are impacted by RUNSTATS in an index that contains an expression as follows:

- ▶ SYSIBM.SYSKEYTARGETS contains one row for each key target that is participating in an extended index definition.
- ▶ SYSIBM.SYSKEYTGTDIST contains one or more rows for the first key target of an extended index key.

## Taking advantage of index on expressions

**Important:** You do not need to change your query to help the optimizer to use an index that contains an expression. However, further changes to the query make the optimizer skip the index on expression. On dynamic SQL environments, this can cause the index not to be used.

We observed that the optimizer does not use the index on expression in some circumstances where, for example, an aggregate function and a GROUP BY are involved. The optimizer prefers to use the traditional index and make the calculations before the resolution of the GROUP BY.

Complex expressions that are good candidates for materialized query tables (MQTs) can also be implemented as index on expressions. You may consider this possibility if your application requirements are not compatible with an MQT because, for example, keeping the data up to date requires a high frequency of REFRESH operations. Expressions on indexes are calculated and kept in sync with the base data as soon as the data is updated.

Example 7-53 shows a query including a predicate that can be made indexable by the implementation of the index shown in Example 7-54.

*Example 7-53 Sample query from TCP-H for index on expression implementation*

---

```
SELECT
  L_LINENUMBER,
  SUM(L_EXTENDEDPRI* (1-L_DISCOUNT) * (1+L_TAX)) AS SUM_CHARGE
FROM
  WORK.LINEITEM
GROUP BY
  L_LINENUMBER;
```

---

*Example 7-54 Creating an index on expression*

---

```
CREATE INDEX
  WORK.LINEITEM_XEXP_01
ON
  WORK.LINEITEM
  ( L_LINENUMBER ASC,
    L_EXTENDEDPRI*(1-L_DISCOUNT)*(1+L_TAX)
  );
```

---

## 7.8 Working with the ADD CLONE SQL command

The service-level agreement (SLA) of data warehouse systems becomes more and more aggressive, and a 24x7x365 SLA is not exaggerated for operational data warehouses.

Many installations have data warehouse processing that involves LOAD REPLACE utilities that make the objects unavailable longer than desired or allowed. Other installations might have an online process that includes the massive updating or summarization of data that might encounter or produce concurrency issues.

The new cloned table concept provides the ability to generate a table with the exact same attributes as a table that already exists in the data warehouse. While the original object remains operational, the clone can be accessed by applications, SQL, or utilities. It can also be used as the target of the offline or disreputable processing without impacting the operations on the base table. After the update processing is finished, there is an exchange of data, and the base table becomes the container of the information of the clone, and vice versa.

The creation or drop of a clone table does not impact applications that access base table data.

**Important:** The cloning process does not apply to data. No rows are copied from the base table to the clone, only the definition of the table.

Figure 7-14 on page 144 illustrates the steps that are involved in a cloning operation. The steps are described as follows:

1. The process starts with the ADD CLONE SQL command against the base table. The clone table is created identical to the base table in every way, with the same columns, check constraints, indexes, and before triggers, for example. Security privileges are not cloned, and privileges need to be applied on the clone table, unless using Resource Access Control Facility (RACF)-based security with an appropriate profile.
2. After the clone objects creation, you can access them directly without impacting the base table. Processes that require exclusive locking on the objects or that can be non-compatible with an operational data warehouse can be executed while the base table is accessible all the time to users and applications.
3. After the update processing is achieved on the clone table, it contains the most updated information. The EXCHANGE DATA SQL command is run. You can imagine this operation like an extended FAST SWITCH that is applied to all the VSAM data sets involved in the object definitions. Table spaces, indexes, and auxiliary table spaces for LOBs have their definitions changed in the DB2 catalog. This quick operation makes available the new data to the users in a nondisruptive and fast way. The clone table, which now contains the data that was previously available for the users, remains available for another cloning cycle.

**Important:** No base object quiesce is necessary. This process does not invalidate plans, packages, or the dynamic statement cache.

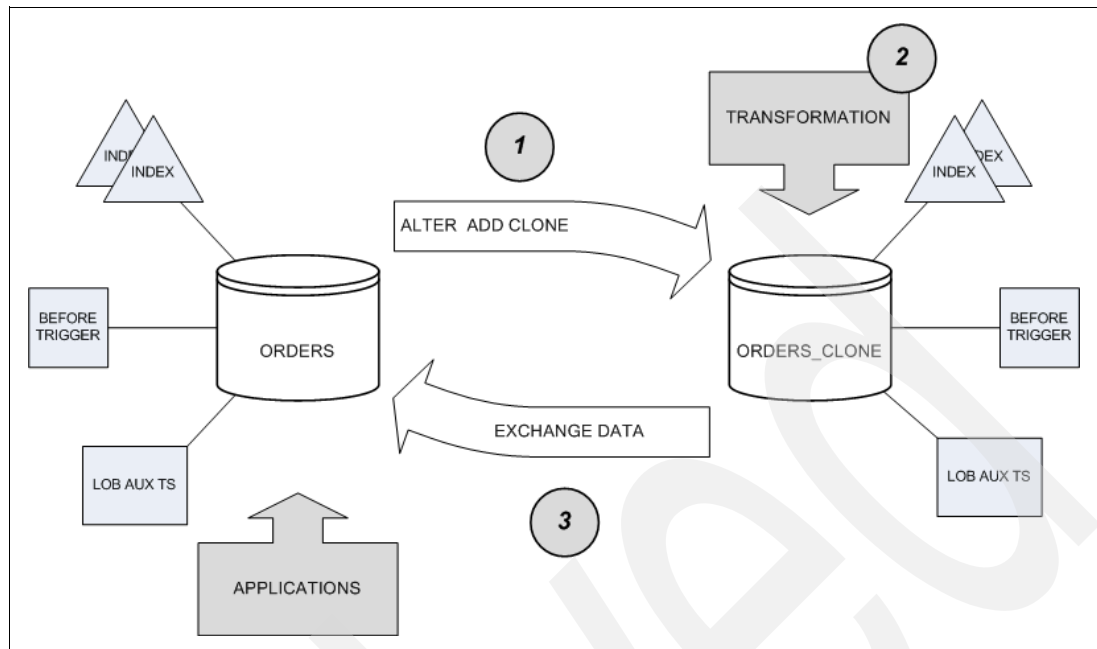


Figure 7-14 ADD CLONE process schematic representation

### 7.8.1 Operating cloned objects

A base table must be compliant with some restrictions *before* it can accept a clone:

- ▶ It cannot participate in a referential constraint.
- ▶ The base table cannot have any AFTER triggers (only BEFORE triggers) defined against it.
- ▶ Only one clone relationship can be created for a table.
- ▶ It can only create a clone in a DB2 managed table space. The table space is a universal table space, and there is only a table in the table space.
- ▶ Creating a clone against a table defined in a non-universal table space will not fail and a SQL error is not received. However the table is not defined within the complete scope of the ADD CLONE command. That is, indexes will be missing.
- ▶ No clones can be defined on MQTs.

The following considerations apply *after* the clone creation:

- ▶ A base table cannot receive a referential integrity constraint if the table has been cloned.
- ▶ After a table enters a cloned relationship, you can create a BEFORE trigger on the base table and that trigger will be created for base and clone tables.
- ▶ Index creation is allowed against the base table after that table has been cloned, and that index will be created for the base and cloned tables. If the clone table should happen to have data in it when the index is created, the index on the clone table will be placed in RBDP/PSRBD status.
- ▶ The RUNSTATS utility cannot be run against a clone table.
- ▶ FASTSWITCH is not allowed.
- ▶ No online schema changes for base and clone table are allowed.
- ▶ Real-time statistics for the base table are invalidated.



Example 7-55 shows the DDL that is used for the clone creation of three tables of our data model. The name for the clone objects follows the naming rules for tables.

*Example 7-55 SQL sample of creating a CLONE table of CUSTOMER*

---

```
ALTER TABLE EXAMPLE.CUSTOMER ADD CLONE EXAMPLE.CUSTOMER_CLONE;
ALTER TABLE EXAMPLE.ORDERS ADD CLONE EXAMPLE.ORDERS_CLONE;
ALTER TABLE EXAMPLE.LINEITEM ADD CLONE EXAMPLE.LINEITEM_CLONE;
```

---

Example 7-56 shows a sample query against the DB2 catalog that can be used for getting information about cloned objects.

*Example 7-56 Sample query for inquiring clone tables information from the DB2 catalog*

---

```
SELECT
    SUBSTR(NAME,1,15) AS NAME
    ,TYPE
    ,SUBSTR(TSNAME,1,8) AS TSNAME
    ,CARD
    ,NPAGES
    ,PCTPAGES
    ,CREATEDTS
    ,HEX(RBA1) AS RBA1
    ,HEX(RBA2) AS RBA2
    ,STATTIME
FROM
    SYSIBM.SYSTABLES
WHERE DBNAME = 'CLONING';
```

---

Example 7-57 shows a sample output.

*Example 7-57 Sample DB2 catalog information about cloned tables*

---

NAME	TYPE	TSNAME	CARD	NPAGES	PCTPAGES	CREATEDTS	RBA1	RBA2
ORDERS	T	ORDERS	-1	153	85	2008-04-21-14.26.26.492153	00030842BC51	000308ABA074
ORDERS_CLONE	C	ORDERS	0	-1	-1	2008-04-21-14.53.44.435564	000308ABA074	000308ABA074
CUSTOMER	T	CUSTOMER	-1	220	40	2008-04-21-14.26.57.889064	00030844A20E	000308AB2BCE
CUSTOMER_CLONE	C	CUSTOMER	0	-1	-1	2008-04-21-14.53.44.194213	000308AB2BCE	000308AB2BCE
LINEITEM	T	LINEITEM	-1	186	34	2008-04-21-14.27.27.970241	000308465437	000308AC4726
LINEITEM_CLONE	C	LINEITEM	0	-1	-1	2008-04-21-14.53.44.675273	000308AC4726	000308AC4726

---

Example 7-57 shows the TYPE=C column value for cloned objects. This value is new in DB2 9 and is used to inform you that the object is a clone. The column TBCREATOR of SYSTABLES for a clone table is the name of the creator of the base table.

There is no statistical information on cloned tables. RUNSTATS cannot be executed against a clone table. When the exchange of data is executed, the clone table inherits the statistical information from the base table. In that way, packages and plans are not invalidated, but you can consider running RUNSTATS to update the statistics information if the data on the clone and base table are too different.

The CREATEDTS column of a clone table indicates the time stamp at which it was created. The RBA1 column indicates the log RBA when the table was created, while RBA2 indicates the log RBA when the table was last altered. RBA1 is different between a base table and its clone, because they can only be created at different moments. However, RBA2 is the same for both tables, as seen in Example 7-57.

Example 7-58 shows the display of a database that contains cloned objects.

*Example 7-58 Display of a database that contains cloned objects*

```

DSNT360I  -D912 *****
DSNT361I  -D912 *   DISPLAY DATABASE SUMMARY
              *   GLOBAL
DSNT360I  -D912 *****
DSNT362I  -D912      DATABASE = CLONING  STATUS = RW
              DBD LENGTH = 12104
DSNT397I  -D912
NAME      TYPE PART  STATUS          PHYERRLO PHYERRHI CATALOG  PIECE
-----
ORDERS    TSB1  0001  RW
ORDERS    TSC2  0001  RW
ORDE1LEQ  IXB1  L*    RW
ORDE1LEQ  IXC2  L*    RW

```

The message DSNT397I in Example 7-58 shows the data header TYPE = TS for a table space or IX for an index space. If the object is involved in cloning, the object type is appended with a B for “base” or a C for “clone” along with a data set instance number.

Example 7-59 shows the VSAM data sets that re created during the cloning process of the objects displayed in Example 7-58. The data set instance number is in bold.

*Example 7-59 View of VSAM data sets underlying a cloned object*

Command - Enter "/" to select action	Message	Volume
DSN912.DSNDBC.CLONING.ORDERS.I0001.A001		*VSAM*
DSN912.DSNDBC.CLONING.ORDERS.I0002.A001		*VSAM*
DSN912.DSNDBC.CLONING.ORDE1LEQ.I0001.A001		*VSAM*
DSN912.DSNDBC.CLONING.ORDE1LEQ.I0002.A001		*VSAM*

Refer to the column CLONE in the SYSTABLESPACE catalog table to determine if a table space and all its related objects are involved with cloning. A CLONE='Y' value indicates that cloning is active. The INSTANCE column value indicates the data set instance number for the current base objects.

The cloned indexes are not visible in the DB2 Catalog, but the VSAM data sets are defined. When defined, you can execute SQL against a cloned table, as shown in Example 7-60. The DB2 Control Center (v9.5.0.808) and DB2 Admin Tool (DB2 Administration Menu 7.2.0) cannot be used to perform a SELECT against a CLONE. Instead, use SPUFI.

Example 7-60 shows how you can execute an EXCHANGE DATA operation by using SPUFI. This process is reversible. You can exchange data between the base table and its clone an unlimited number of times.

*Example 7-60 Exchange data execution sample*

```

SELECT COUNT(*) FROM EXAMPLE.ORDERS      ;
5000
-----+-----+-----+-----+-----+-----+-----+
SELECT COUNT(*) FROM EXAMPLE.ORDERS_CLONE;
1000
-----+-----+-----+-----+-----+-----+-----+
EXCHANGE DATA BETWEEN TABLE EXAMPLE.ORDERS_CLONE AND EXAMPLE.ORDERS;
-----+-----+-----+-----+-----+-----+-----+
COMMIT;
-----+-----+-----+-----+-----+-----+-----+

```

```

SELECT COUNT(*) FROM EXAMPLE.ORDERS      ;
      1000
-----+-----+-----+-----+-----+-----+
SELECT COUNT(*) FROM EXAMPLE.ORDERS_CLONE;
      5000

```

---

Example 7-61 shows that an exchange data operation can be rolled back if needed.

*Example 7-61 Rollback of an exchange data operation*

```

SELECT COUNT(*) FROM EXAMPLE.ORDERS      ;
      5000
-----+-----+-----+-----+-----+-----+
SELECT COUNT(*) FROM EXAMPLE.ORDERS_CLONE;
      1000
-----+-----+-----+-----+-----+-----+
EXCHANGE DATA BETWEEN TABLE EXAMPLE.ORDERS_CLONE AND EXAMPLE.ORDERS;
-----+-----+-----+-----+-----+-----+
ROLLBACK;
-----+-----+-----+-----+-----+-----+
SELECT COUNT(*) FROM EXAMPLE.ORDERS      ;
      5000
-----+-----+-----+-----+-----+-----+
SELECT COUNT(*) FROM EXAMPLE.ORDERS_CLONE;
      1000

```

---

When executing several EXCHANGE DATA operations within the same logical unit of work (LUW), a rollback operation affects all of them. You must consider the scope of the LUW for this kind of operations to keep the related objects in synchrony in case one of the operations fails. Refer to Example 7-62.

*Example 7-62 Exchange data and operations involving several clones*

```

SELECT COUNT(*) FROM EXAMPLE.ORDERS      ;
      1000
-----+-----+-----+-----+-----+-----+
SELECT COUNT(*) FROM EXAMPLE.ORDERS_CLONE;
      5000
-----+-----+-----+-----+-----+-----+
SELECT COUNT(*) FROM EXAMPLE.CUSTOMER    ;
      1000
-----+-----+-----+-----+-----+-----+
SELECT COUNT(*) FROM EXAMPLE.CUSTOMER_CLONE;
      5000
-----+-----+-----+-----+-----+-----+
SELECT COUNT(*) FROM EXAMPLE.LINEITEM    ;
      1000
-----+-----+-----+-----+-----+-----+
SELECT COUNT(*) FROM EXAMPLE.LINEITEM_CLONE;
      5000
-----+-----+-----+-----+-----+-----+
EXCHANGE DATA BETWEEN TABLE EXAMPLE.ORDERS_CLONE AND EXAMPLE.ORDERS;
-----+-----+-----+-----+-----+-----+
EXCHANGE DATA BETWEEN TABLE EXAMPLE.CUSTOMER_CLONE AND EXAMPLE.CUSTOMER;
-----+-----+-----+-----+-----+-----+
EXCHANGE DATA BETWEEN TABLE EXAMPLE.LINEITEM_CLONE AND EXAMPLE.LINEITEM;
-----+-----+-----+-----+-----+-----+
ROLLBACK;
-----+-----+-----+-----+-----+-----+
SELECT COUNT(*) FROM EXAMPLE.ORDERS      ;

```

```

1000
-----+-----+-----+-----+-----+-----+-----+
SELECT COUNT(*) FROM EXAMPLE.ORDERS_CLONE;
5000
-----+-----+-----+-----+-----+-----+-----+
SELECT COUNT(*) FROM EXAMPLE.CUSTOMER    ;
1000
-----+-----+-----+-----+-----+-----+-----+
SELECT COUNT(*) FROM EXAMPLE.CUSTOMER_CLONE;
5000
-----+-----+-----+-----+-----+-----+-----+
SELECT COUNT(*) FROM EXAMPLE.LINEITEM    ;
1000
-----+-----+-----+-----+-----+-----+-----+
SELECT COUNT(*) FROM EXAMPLE.LINEITEM_CLONE;
5000

```

---

Example 7-63 shows the procedure for dropping a clone object, which is done with the ALTER command.

*Example 7-63 Drop clone*

---

```
ALTER TABLE ORIG.CUSTOM DROP CLONE TABLE;
```

---

## 7.9 Table space partitioning

In a data warehouse environment, the size, growth estimates, or both of certain tables, such as fact tables and history tables, pose a challenge in designing and administering the table space. They also present challenges in ensuring query performance, and loading and deleting data. Partitioned table spaces, as they have evolved with the implementation in DB2 9 for z/OS, help resolve most of these problems.

The partitioned table spaces have the following characteristics among others:

- ▶ There is a single table per table space.
- ▶ One table space consists of multiple physical data sets called *partitions*.
- ▶ Each partition can be considered as a unit of storage.
- ▶ They have a maximum of 4096 partitions.
- ▶ Each partition can reside in a different storage group.
- ▶ Partitions are based on the boundary values of data columns or on the size threshold of physical data sets or a partition.
- ▶ They are highly scalable. The size limit of a partitioned table space is 128 TB.
- ▶ A utility job can work on a part of data, while allowing applications to access other parts.
- ▶ A partition can be added dynamically.
- ▶ A partition that contains old data can be reused by using rotation.
- ▶ The maximum size of each partition can be 64 GB (64 x 4096, not 128 TB).
- ▶ They can run a mass update, delete, or insert of jobs at partition level to reduce the elapse time.
- ▶ DB2 can do parallel processing on more than one partition simultaneously.

## 7.9.1 Universal table space

As a follow-on to all of the changes regarding partitioning in DB2 V8, which centered on the unbundling of the partitioning index functions, DB2 9 has introduced *universal table space* as a new partitioned table space type. Universal table spaces combine the advantages of both partitioned and segmented table spaces. The new characteristics help with scalability and growth of data.

There are two different types of universal table spaces:

- ▶ Partition by growth
- ▶ Partition by range

### Partition by growth

Partition by growth simplifies manageability of table spaces that may exceed existing size limitations that are associated with segmented table spaces. Partition-by-growth table spaces have the following features:

- ▶ The creation of partitions is managed by DB2.
- ▶ DB2 automatically adds a new partition when an insert cannot be added to the current partition.
- ▶ They are best used when a table is expected to exceed 64 GB and does not have a suitable partitioning key for the table.
- ▶ They combine the size and growth capability of partitions along with space management, mass delete, and insert performance characteristics of segmented table space.
- ▶ They are useful in a data warehouse environment to store high volume growth data because the space management is done by DB2.
- ▶ They are created by specifying the MAXPARTITIONS, DSSIZE, and SEGSIZE parameters in the CREATE TABLESPACE statement. This type of table space can grow up to 128 TB with a maximum size determined by the values of MAXPARTITIONS, DSSIZE, and page size.
  - Like segmented table spaces, the SEGSIZE value must be a multiple of 4K between 4 and 64 (inclusive) and cannot be changed.
  - The value of MAXPARTITIONS can range from 1 to 4096 (default 256), but the maximum value depends upon the DSSIZE and page size value for the table space.
  - DSSIZE can have one of the following values and cannot be changed:
    - 1 G - 1 GB
    - 2 G - 2 GB
    - 4 G - 4 GB
    - 8 G - 8 GB
    - 16 G - 16 GB
    - 32 G - 32 GB
    - 64 G - 64 GB

**Attention:** For a DSSIZE of more than 4G, the following conditions are required:

- ▶ DB2 is running with DFSMST<sup>™</sup> Version 1 Release 5.
- ▶ The data sets for the table space are associated with a DFSMS data class that has been specified with extended format and extended addressability.

- ▶ They have better space management as it relates to varying-length rows because a segmented space map page has more information about free space than a partitioned space map page.
- ▶ They have improved mass delete performance and have immediate reuse of the segments after the mass delete.
- ▶ A compression dictionary is copied from the previous partition to the new partition.
- ▶ The Freespace, caching, define, logging and trackmod attributes are same for each partition.
- ▶ All utilities can operate at the partition level, except the LOAD utility.
- ▶ The APPEND clause on CREATE TABLE statement can be specified to insert the data at the end of table space instead of searching for the space from the first partition. This clause is useful in improving the performance when doing a bulk load in a data warehouse environment.
- ▶ Example 7-64 shows a sample DDL for partition by growth. The DDL was used in our scenario to create a table space for the ORDER\_TRANSACTION\_FACT table.

*Example 7-64 DDL for partition-by-growth table space*

---

```
CREATE TABLESPACE TSTRANFT IN DBTENG B USING STOGROUP TENG B
DSSIZE 2G MAXPARTITIONS 48 SEGSIZE 4 LOCKSIZE ANY BUFFERPOOL BP3
```

---

- ▶ They can also be created by using CREATE TABLE PARTITION BY SIZE EVERY nG statement. This is only available when you do not specify a table space name in the CREATE TABLE statement.

**Note:** The partition-by-growth table space cannot be used in a work file database and LOB table spaces.

## Partition by range

Partition-by-range table spaces have the following features:

- ▶ They require a partitioning column. Range-partitioned table spaces are based on partitioning ranges for the partitioning column.
- ▶ They can spread a large table over several DB2 storage groups.
- ▶ They let a utility job work on part of the data, while allowing other applications to concurrently access data on other partitions.
- ▶ They can break mass update, delete, or insert operations into separate jobs. Each works on a different partition that is helpful in a data warehouse environment when doing mass load, delete, or a table scan.
- ▶ The maximum size of a range-partitioned universal table space is 128 TB.
- ▶ All current types of indexes are allowed on partition-by-range table spaces.
- ▶ They combine the size and data distribution capability of the traditional partitioned table space with space management, mass delete, and insert performance characteristics of the segmented table space.
- ▶ They take advantage of parallelism for certain read-only queries. When DB2 determines that processing will be extensive, it can begin parallel processing of more than one partition at a time. Parallel processing (for read-only queries) is most efficient when the partitions are spread over different disk volumes.
- ▶ They can take advantage of query parallelism in a DB2 data sharing group.

- They are created by specifying the Numparts, DSSize, and Segsize parameters in the CREATE TABLESPACE statement as shown in Example 7-65. This DDL was used to create a table space for the PART table in our scenario.

*Example 7-65 DDL for creating a partition-by-range table space*

---

```
CREATE TABLESPACE TSPART IN DBTENG
USING STOGROUP TENG PRIQTY -1 SECQTY -1 SEGsize 4 DSSize 4G
Numparts 10 COMPRESS YES BUFFERPOOL BP2
```

---

The PART table was created by using the PARTITION BY RANGE clause on the CREATE TABLE statement. Ten partition ranges were defined as shown in Example 7-66.

*Example 7-66 Partition-by-range table definition*

---

```
PARTITION BY RANGE (P_PARTKEY)
PARTITION 1 ENDING AT (1250),
PARTITION 2 ENDING AT (2500),
PARTITION 3 ENDING AT (3750),
PARTITION 4 ENDING AT (4000),
PARTITION 5 ENDING AT (5250),
PARTITION 6 ENDING AT (6500),
PARTITION 7 ENDING AT (7750),
PARTITION 8 ENDING AT (8000),
PARTITION 9 ENDING AT (9250),
PARTITION 10 ENDING AT (10500))
```

---

- They provide better space management as it relates to varying-length rows because a segmented space map page has more information about free space than a partitioned space map page.
- They have improved mass delete performance and immediate reuse of the segments after the mass delete.
- No index controlled partitioning can be defined. Only table controlled partitioning is acceptable.
- They can grow up to 128 TB depending upon Numparts, DSSize, and page size values.
- Similar to partition-by-growth table spaces, the DSSize value can vary from 1G to 64 G. The maximum DSSize value depends on the value of Numparts that is specified. If Numparts is greater than 256, the maximum DSSize value also depends on the page size of the table space.
- This table space can be used for doing a load and scan at the partition level.
- Data in the table is spread across the multiple partitions as per the table definition in the partitioning key or the partitioning index on the table.

Data can be re-distributed between existing partitions by using ALTER TABLE with the ALTER PARTITION clause and then doing a REORG on the table space. In the partition-by-range table space, a partition can be added dynamically (as we describe in the following section), or an existing partition that has old data can be reused.

## Adding a partition

In a partition-by-range table space, a partition can be added dynamically by using the ALTER TABLE statement with the ADD PARTITION clause. It adds a partition to the table and to each partitioning index on the table. The existing table space PRIQTY and SECQTY attributes of the previous logical partition are used for the space attributes of the new partition. For each partitioned index, the existing PRIQTY and SECQTY attributes of the previous partition are

used. This feature is extremely useful in a data warehouse environment where a partition can be added dynamically on new range of values or on the basis of a time period. For example, on our PART table, if we want to add a new partition for new values ending at 12000, we can use the SQL statement shown in Example 7-67. It adds another partition dynamically for the p\_partkey range from 10500 to 11999.

*Example 7-67 Adding a partition*

---

```
ALTER TABLE PART ADD PARTITION ENDING AT (12000)
```

---

### Rotating a partition

The first logical partition in a partition-by-range table space can be reused by using the ALTER TABLESPACE statement with the ROTATE PARTITION FIRST TO LAST clause. It specifies that the first logical partition should be rotated to become the last logical partition. A new range for is specified by using ENDING AT clause for this partition, which is now logically last. This can be done dynamically, and the data in the first partition is deleted. A partition rotation can be applicable when the data in the first partition is no longer of use to the application.

For example, an application that requires only the last five years of data has a partitioned-by-range table space that is partitioned on the year value. After five years, the first partition can be reused by using the rotation-of-partition feature, which helps in reusing the existing space and definitions.

## 7.10 Materialized query tables

MQTs offer a way to speed up response times on well-known and repeatable queries against large volumes of data, where results are often expressed as summaries or aggregates. An MQT is a real (materialized) table that is built from the result set of a query. Like any other table, an MQT can have indexes. You can use RUNSTATS to collect and store table statistics.

MQTs primarily exist for performance. Properly selected, MQTs perform a set of calculations that can be used over and over by subsequent queries, but without repeatedly going through the same operations. The qualifying rows do not need to be fetched again, saving input time. Also, the calculations do not need to be redone, which saves CPU time.

With informational referential constraints (NOT ENFORCED option), users can declare a referential constraint and avoid the overhead of enforcing the referential constraints by DB2. However, DB2 can use the data association information for query rewrite of MQTs when extra tables are in the MQT definition.

In this section, we provide simple examples of MQTs as they relate to our scenario. For a more comprehensive description, see Chapter 11, “Using materialized query tables to improve SQL performance,” in *DB2 Version 9.1 for z/OS Performance Monitoring and Tuning Guide*, SC18-9851.

### 7.10.1 When to consider an MQT

The design of good materialized query tables requires adequate up-front planning and analysis. You need to be familiar with the query workload to identify patterns for accessing



tables and frequently performed aggregation and summarization. When deciding whether to create a materialized query table, consider:

- ▶ Does the MQT significantly increase performance?
- ▶ Do many queries benefit? Do the most frequent, most critical, or most expensive and longest running queries benefit?
- ▶ Does the MQT offer resource savings: communication, I/O, and CPU?
- ▶ Is the loss of disk space that contains the MQT and its indexes a worthwhile trade for the performance gained?
- ▶ What is the cost of updating or refreshing the MQT?
- ▶ What are the patterns for accessing groups of tables, for aggregation, and for grouping requirements?
- ▶ How current does the data in the MQT need to be? Does it need to be up to the minute?
- ▶ For MQTs that are maintained in real time, are automatic updates too slow?
- ▶ What is the logging requirement when large MQTs are refreshed?
- ▶ Does the MQT need to be system-maintained or user-maintained?

## 7.10.2 MQTs used in our scenario

Example 7-68 defines an MQT in the ORDERS table that counts the orders by the customer key and the order date with the following options:

### **DATA INITIALLY DEFERRED**

Specifies that the data is not inserted into the MQT when it is created. Use the REFRESH TABLE statement to populate the MQT.

### **REFRESH DEFERRED**

Specifies that the data in the table can be refreshed at any time by using the REFRESH TABLE statement.

### **MAINTAINED BY SYSTEM**

Specifies that the MQT is maintained by the system. Only the REFRESH statement is allowed on the table.

### **ENABLE QUERY OPTIMIZATION**

Specifies that the MQT can be used for query optimization. In real life, we recommend that you *initially* DISABLE QUERY OPTIMIZATION until after the first refresh. Otherwise optimizer can redirect queries to an empty MQT.

#### *Example 7-68 Creating MQT*

```
CREATE TABLE ORDERS_MQT (CUSTKEY, ORDERDATE, CNT) AS
(SELECT O_CUSTKEY, O_ORDERDATE, COUNT(*))
FROM DWHODS.ORDERS
GROUP BY O_CUSTKEY, O_ORDERDATE)
DATA INITIALLY DEFERRED
REFRESH DEFERRED
MAINTAINED BY SYSTEM
ENABLE QUERY OPTIMIZATION;
```

```
-----+-----+-----+-----+-----+-----+-----
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+-----+-----
```

```
-----+-----+-----+-----+-----+-----+-----
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
```

```
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+-----+-----
```

```
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 1
DSNE621I NUMBER OF INPUT RECORDS READ IS 34
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 46
```

---

The MQT can be dropped like any other table, as shown in Example 7-69.

*Example 7-69 Dropping MQT*

```
DROP TABLE ORDERS_MQT;
-----+-----+-----+-----+-----+-----+-----+-----
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+-----+-----+-----
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+-----+-----+-----
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 1
DSNE621I NUMBER OF INPUT RECORDS READ IS 35
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 47
```

---

Another example of an MQT is in 12.4, “Improving response times by using materialized query tables” on page 300, where we discuss operation business intelligence (BI) implementation.

## 7.11 OLAP functions

DB2 9 for z/OS offers new SQL enhancements for improving OLAP functionalities in a data warehouse solution. The following OLAP expressions were introduced in DB2 9 for z/OS:

- ▶ RANK and DENSE\_RANK
- ▶ ROW\_NUMBER

These functions provide the ability to return ranking and row numbering information as a scalar value in the result query. The result of a RANK, DENSE\_RANK, and ROW\_NUMBER specification is BIGINT, and the result cannot be null. When an OLAP function is invoked, a window is specified that defines the rows and the order of rows over which the function is applied.

### 7.11.1 RANK and DENSE\_RANK

RANK or DENSE\_RANK specifies the ordinal rank of a row that is computed within the specified window. Rows that are not distinct, with respect to the ordering within the specified window, are assigned the same rank. The value for the ranking can be with or without gaps in the sequence depending upon the function that is used (RANK or DENSE\_RANK) and non-distinct values of the columns specified in the window.

RANK specifies that the row is 1 plus the number of rows preceding it in the result set. All the rows that are not distinct have the same rank values. RANK and DENSE\_RANK are useful expressions in finding the top *n* rows in certain business scenarios, for example, in finding the top 10 suppliers.

The PARTITION BY clause defines the window within which the ranking occurs. Upon changing the value of the PARTITION BY column or columns, the ranking is reset to 1. Without the PARTITION BY clause, the window is the entire qualified result.

Refer to the query shown in Example 7-70 for the use of the RANK function. This query finds the top four customers for each quarter in the year 2007 based on the highest quantity value.

*Example 7-70 RANK expression*

```
SELECT ORDER_YEAR_VAL, ORDER_QTR_CODE, CUST_NAME, RANK_VAL, TOTAL_QUANT
FROM
  (SELECT B.ORDER_YEAR_VAL, B.ORDER_QTR_CODE, C.CUST_NAME, RANK() OVER ( PARTITION BY B.ORDER_QTR_CODE
ORDER BY SUM(A.QUANTITY) DESC) AS RANK_VAL,
    SUM(A.QUANTITY) AS TOTAL_QUANT
  FROM
    DWHDDS.ORDER_TRANSACTION_FACT AS A INNER JOIN DWHDDS.ORDERDATE_DIM_VW AS B ON A.ORDERDATEKEY
= B.ORDERDATEKEY INNER JOIN DWHDDS.CUSTOMER_DIM AS C ON A.CUSTKEY_DW = C.CUSTKEY_DW
    WHERE B.ORDER_YEAR_VAL = 2007
    GROUP BY B.ORDER_YEAR_VAL, B.ORDER_QTR_CODE, C.CUST_NAME) AS QTR_RANKING
WHERE RANK_VAL < 5
ORDER BY ORDER_QTR_CODE, RANK_VAL
```

The query in Example 7-70 does the SUM of quantity bought by customers in each quarter and then uses the RANK function to rank them for each quarter based on the total quantity bought in each quarter. The quantity data is read from the ORDER\_TRANSACTION\_FACT table and then ranked per quarter with the help of the PARTITION BY clause in the RANK expression. Refer to the Figure 7-15 for the results from the query in Example 7-70.

RANK.sql				
Messages Parameters Results				
ORDER_YEAR_VAL	ORDER_QTR_CODE	CUST_NAME	RANK_VAL	TOTAL_QUANT
2007	QTR1	Gaurav K G	1	95.00
2007	QTR1	Sam C I	2	92.00
2007	QTR1	Gaurav L P	3	91.00
2007	QTR1	Frank R H	4	90.00
2007	QTR2	Peter R L	1	121.00
2007	QTR2	Sam C I	2	98.00
2007	QTR2	Paul D V	3	83.00
2007	QTR2	Gaurav Q K	3	83.00
2007	QTR2	Andrew R S	3	83.00
2007	QTR3	Paul W Z	1	104.00
2007	QTR3	Sreeni P P	2	91.00
2007	QTR3	Khadija S M	3	84.00
2007	QTR3	Khadija F Z	4	82.00
2007	QTR4	Frank R H	1	114.00
2007	QTR4	Sam B G	2	112.00
2007	QTR4	Cris T N	3	96.00
2007	QTR4	Peter R L	4	95.00

*Figure 7-15 Results for the top four customers for each quarter*

If you notice the results for QTR 2, three customers have the third RANK because they were identical in the total number of items bought by them in QTR2. As per the RANK expression definitions, if we use a similar query to get the top 10 customers, then the next RANK after these three customers is sixth. Therefore the RANK function leaves holes in the rank values that are retrieved.

The DENSE\_RANK function specifies that the rank of a row is defined as 1 plus the number of preceding rows that are distinct with respect to the ordering. Therefore, there will be no gaps in the sequential rank numbering. To describe DENSE\_RANK functionality in more

detail, we use this expression for the previous scenario and then compare the results with RANK. Example 7-71 shows the SQL statement that uses DENSE\_RANK.

**Example 7-71 DENSE\_RANK expression**

```
SELECT ORDER_YEAR_VAL, ORDER_QTR_CODE, CUST_NAME, RANK_VAL, TOTAL_QUANT
FROM
    (SELECT B.ORDER_YEAR_VAL, B.ORDER_QTR_CODE, C.CUST_NAME, DENSE_RANK() OVER ( PARTITION BY
B.ORDER_QTR_CODE ORDER BY SUM(A.QUANTITY) DESC) AS RANK_VAL,
    SUM(A.QUANTITY) AS TOTAL_QUANT
    FROM
        DWHDDS.ORDER_TRANSACTION_FACT AS A INNER JOIN DWHDDS.ORDERDATE_DIM_VW AS B ON A.ORDERDATEKEY
= B.ORDERDATEKEY INNER JOIN DWHDDS.CUSTOMER_DIM AS C ON A.CUSTKEY_DW = C.CUSTKEY_DW
        WHERE B.ORDER_YEAR_VAL = 2007
        GROUP BY B.ORDER_YEAR_VAL, B.ORDER_QTR_CODE, C.CUST_NAME) AS QTR_RANKING
WHERE RANK_VAL < 5
ORDER BY ORDER_QTR_CODE, RANK_VAL
```

Figure 7-16 shows the results of the DENSE\_RANK expression.

ORDER_YEAR_VAL	ORDER_QTR_CODE	CUST_NAME	RANK_VAL	TOTAL_QUANT
2007	QTR1	Gaurav K G	1	95.00
2007	QTR1	Sam C I	2	92.00
2007	QTR1	Gaurav L P	3	91.00
2007	QTR1	Frank R H	4	90.00
2007	QTR2	Peter R L	1	121.00
2007	QTR2	Sam C I	2	98.00
2007	QTR2	Paul D V	3	83.00
2007	QTR2	Gaurav Q K	3	83.00
2007	QTR2	Andrew R S	3	83.00
2007	QTR2	Robert Q C	4	80.00
2007	QTR2	John H A	4	80.00
2007	QTR3	Paul W Z	1	104.00
2007	QTR3	Sreeni P P	2	91.00
2007	QTR3	Khadija S M	3	84.00
2007	QTR3	Khadija F Z	4	82.00
2007	QTR4	Frank R H	1	114.00
2007	QTR4	Sam B G	2	112.00
2007	QTR4	Cris T N	3	96.00
2007	QTR4	Peter R L	4	95.00

Figure 7-16 Results from DENSE\_RANK expression

Between the results for QTR2 when using the RANK expression and when using the DENSE\_RANK expression, the difference is in the results for the QTR2. In RANK, we did not have any customer with a RANK value of four because there were three customers with a RANK value equal to three. Therefore, the next RANK value was six. Because the query only looks for rows that have a RANK value up to four, it did not have the next RANK in the result set. Alternatively when using DENSE\_RANK, the next RANK assigned, after the three customers who received RANK value three, was RANK value four. Therefore, they were included in the result set.

## 7.11.2 ROW\_NUMBER

ROW\_NUMBER specifies that a sequential row number is computed for the row that is defined by the ordering, starting with 1 for the first row. If the ORDER BY clause is not specified in the window, the row numbers are assigned to the rows in an arbitrary order, because the rows are returned (not according to any ORDER BY clause in the select-statement). ROW\_NUMBER prompts DB2 to provide a sequential counter based on the expression specified in the OVER keyword. To demonstrate the result by using ROW\_NUMBER from the previous scenario, we ran the SQL statement shown in Example 7-72 on page 157.

*Example 7-72 ROW\_NUMBER*

---

```

SELECT ORDER_YEAR_VAL, ORDER_QTR_CODE, CUST_NAME, ROW_VAL, TOTAL_QUANT
FROM
    (SELECT B.ORDER_YEAR_VAL, B.ORDER_QTR_CODE, C.CUST_NAME, ROW_NUMBER() OVER ( PARTITION BY
B.ORDER_QTR_CODE ORDER BY SUM(A.QUANTITY) DESC) AS ROW_VAL,
    SUM(A.QUANTITY) AS TOTAL_QUANT
    FROM
        DWHDDS.ORDER_TRANSACTION_FACT AS A INNER JOIN DWHDDS.ORDERDATE_DIM_VW AS B ON A.ORDERDATEKEY
= B.ORDERDATEKEY INNER JOIN DWHDDS.CUSTOMER_DIM AS C ON A.CUSTKEY_DW = C.CUSTKEY_DW
        WHERE B.ORDER_YEAR_VAL = 2007
        GROUP BY B.ORDER_YEAR_VAL, B.ORDER_QTR_CODE, C.CUST_NAME) AS QTR_RANKING
WHERE ROW_VAL < 5
ORDER BY ORDER_QTR_CODE, ROW_VAL

```

---

Figure 7-17 shows the results.

Messages	Parameters	Results			
ORDER_YEAR_VAL	ORDER_QTR_CODE	CUST_NAME	ROW_VAL	TOTAL_QUANT	
2007	QTR1	Gaurav K G	1	95.00	
2007	QTR1	Sam C I	2	92.00	
2007	QTR1	Gaurav L P	3	91.00	
2007	QTR1	Frank R H	4	90.00	
2007	QTR2	Peter R L	1	121.00	
2007	QTR2	Sam C I	2	98.00	
2007	QTR2	Paul D V	3	83.00	
2007	QTR2	Andrew R S	4	83.00	
2007	QTR3	Paul W Z	1	104.00	
2007	QTR3	Sreeni P P	2	91.00	
2007	QTR3	Khadija S M	3	84.00	
2007	QTR3	Khadija F Z	4	82.00	
2007	QTR4	Frank R H	1	114.00	
2007	QTR4	Sam B G	2	112.00	
2007	QTR4	Cris T N	3	96.00	
2007	QTR4	Peter R L	4	95.00	

*Figure 7-17 Results from ROW\_NUMBER expression*

Notice that the result set ROW\_NUMBER expression generates a unique sequence number for each customer in all four quarters based on the highest quantity value.

Archived

## Q replication and event publishing

In this chapter, we discuss Q replication and event publishing as solutions for near real-time, but asynchronous, replication of data and propagation of changes from the operational system to the operational data warehouse. We also show how to implement Q replication and event publishing for an operational data warehouse.

This chapter contains the following sections:

- ▶ 8.1, “Introduction to replication functions” on page 160
- ▶ 8.2, “Implementation of Q replication and Event Publisher” on page 162
- ▶ 8.3, “Operating Q replication and event publishing” on page 195

## 8.1 Introduction to replication functions

*Replication* involves the copying of changes from one location, which is the source, to another location, which is the target, and synchronizing the data in both locations. The source and the target can be in servers that are on the same machine or on different machines in the same network.

*Publishing* is a technique that captures changes in a location, which is the source, and makes the changes available for applications or processes. These processes can transform this information before saving it in an operational data warehouse, for example.

The IBM WebSphere Replication Server is a collection of smaller programs that track changes to source databases and replicate some or all of the changes to target databases. To detect these changes, a capture process continuously reads the database recovery log. You can use WebSphere Replication Server to help maintain your data warehouse and facilitate real-time business intelligence. WebSphere Replication Server provides the flexibility to distribute, consolidate, and synchronize data from many locations by using differential replication or extract, transform, and load (ETL) processes.

### 8.1.1 Q replication

Two different types of replication are available in WebSphere Replication Server: Q replication and SQL replication. The main difference is the repository of changes and how these changes are transmitted:

- ▶ Q replication is a WebSphere MQSeries-based product, and the captured changes are stored and moved using MQ queues.
- ▶ SQL replication keeps track of changes in DB2 tables.

Because of its low latency and high throughput capabilities, we implemented Q replication in our scenario.

The Q Capture program replicates transactions from a source table and puts those transactions on a send queue in compact format. Then the Q Apply program gets the compact messages from a receive queue and applies the transactions to a target table (either a DB2 table or a nickname on a DB2 federated server) or passes them to a stored procedure. Figure 8-1 on page 161 shows a schematic representation of the Q replication process.

The following replication scenarios are available within Q replication:

- ▶ *Unidirectional replication*, which replicates changes in a single direction, from source to target
- ▶ *Bidirectional replication*, in which the data replication occurs in two senses  
This configuration requires source and target tables to be identical. Also, it is not possible to replicate a subsets of rows.
- ▶ *Peer-to-peer replication*, under which replication occurs between tables on two or more servers



For our scenario we used unidirectional replication. With unidirectional replication, you replicate data from source tables to target tables or stored procedures. Unidirectional Q replication has the following characteristics:

- ▶ Transactions that occur at a source table are replicated over WebSphere MQ queues to a target table or are passed as input parameters to a stored procedure to manipulate the data.
- ▶ Transactions that occur at the target table are not replicated back to the source table.
- ▶ The target table typically is read only or is not updated by applications other than the Q Apply program.

From any source table, you can replicate either all of the columns and rows or only a subset of the columns and rows. If you want to transform the data, you can specify for the Q Apply program to pass the transactions from a source table as input parameters to a stored procedure that you provide. The stored procedure can update data in either a DB2 or non-DB2 server.

You must create at least one replication queue map to transport data from the Q Capture program on the source server to the Q Apply program on the target server (or the DB2 federated server if you are replicating to a non-DB2 target table). There is one Q subscription for every pair of source and target tables or every pair of source tables and stored procedures.

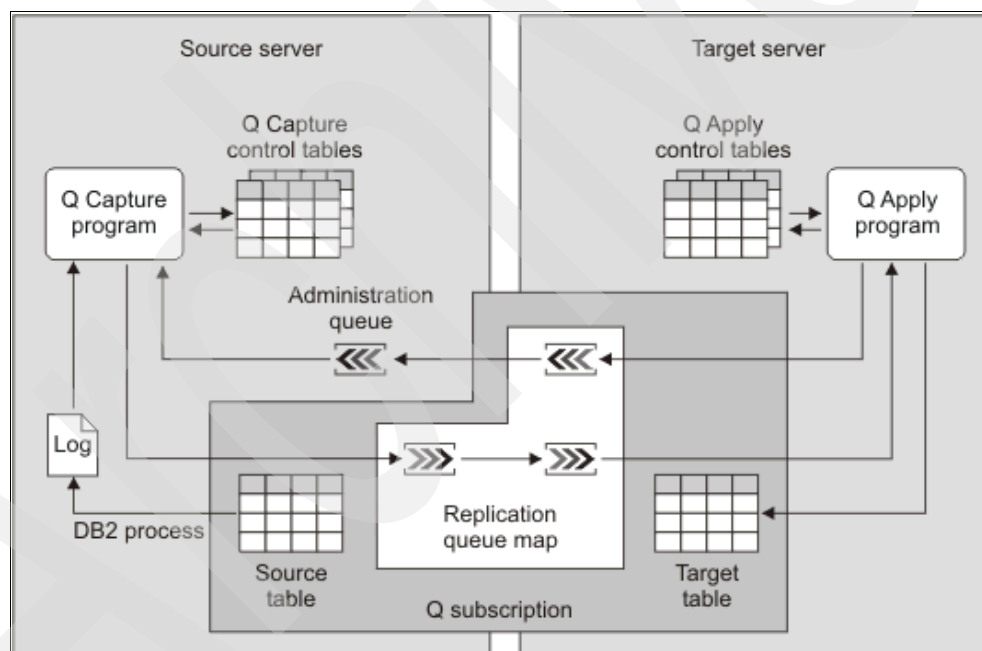


Figure 8-1 Q replication overview

Refer to the following Redbooks publications for more information:

- ▶ *WebSphere Information Integrator Q Replication: Fast Track Implementation Scenarios*, SG24-6487
- ▶ *WebSphere Replication Server for z/OS Using Q Replication: High Availability Scenarios for the z/OS Platform*, SG24-7215

## 8.1.2 Event publishing

Event publishing captures changes on source tables and converts committed transactional data to messages in XML or delimited format. Each message can contain an entire transaction or only a row-level change. These messages are put on an MQ queue and read by an application.

You can use event publishing for a variety of purposes that require published data, including feeding central information brokers and Web applications, and triggering actions based on insert, update, or delete operations at the source tables. Source tables can be relational tables in DB2 for z/OS and DB2 for Linux, UNIX, and Windows.

In our data warehouse implementation, the messages are taken by DataStage, which updates the operational data warehouse system after data transformation.

Figure 8-2 shows a schematic representation of event publishing.

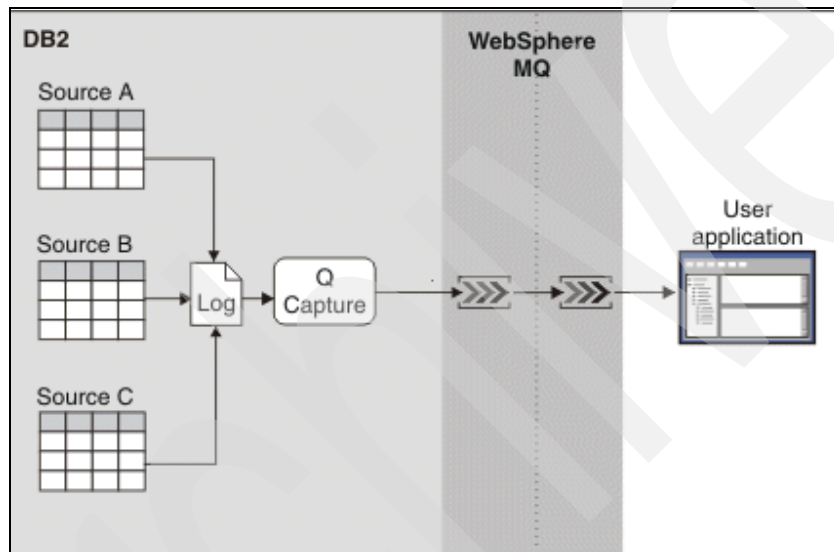


Figure 8-2 Event publishing overview

## 8.2 Implementation of Q replication and Event Publisher

In this section, we explain how we implemented Q replication and Event Publisher in our operational data warehouse environment:

- Q replication is used for the tables ORDER and LINEITEMS. The information of these tables is required in the operational data warehouse and changes must be propagated with low latency. By having these tables, current information can be available for operational reporting without the need of accessing the production tables. The replication is done unidirectionally from the production system to the operational data warehouse. No change is done to the data, and the source and target tables have the same structure.
- Event publishing is used for the table CUSTOMER. Changes are made available to DataStage, which reads the event publishing messages that are stored in a delimited format from an MQSeries queue. The latency requirements are not relevant, and it is more important to transform the data. DataStage updates the information that comes from event publishing, making aggregations and cross-reference checks before storing the data into the data warehouse.

This section is by no means a full description of all the possibilities and problems that you may encounter, but a quick guide through the process. In the following descriptions, we assume that the System Modification Program/Extended (SMP/E) installation of the products were done successfully.

Refer to *WebSphere Information Integration Version 9.1 Replication Installation and Customization Guide for z/OS*, SC19-1025, for more information.

## 8.2.1 Common infrastructure

Both Q replication and event publishing use the Q Capture program to capture changes from the source objects. It converts the transaction data to messages and place the messages on a WebSphere MQ queue.

Q replication and event publishing can share the following objects:

- ▶ Q Capture started task
- ▶ Q Capture user
- ▶ WebSphere MQ queues

For our implementation, we created a new user ID, QREP, to be used in the started tasks. Example 8-1 shows the Resource Access Control Facility (RACF) definitions for its creation. This example also shows the definitions of the started tasks for Q Capture and Q Apply.

*Example 8-1 Defining a user and started tasks for replication server*

---

```
ADDUSER
  QREP DFLTGRP(SYS1)
  OMVS(UID(xxx)      ----> define UID
  HOME(/u/qrep)
  PROGRAM(/bin/sh))
+ NAME('QREP STARTED TASKS')
NOPASSWORD
NOIDCARD
SETOPTS
CLASSACT(STARTED)

RDEFINE
  STARTED CAPT*.*
  STDATA(USER(QREP)
  GROUP(SYS1)
  TRACE(YES))

RDEFINE
  STARTED APPL*.*
  STDATA(USER(QREP)
  GROUP(SYS1)
  TRACE(YES))

SETOPTS
  RACLIST(STARTED)
  GENERIC(STARTED)
  REFRESH
```

---

This user ID must have DB2 and WebSphere MQ privileges defined. It also needs a UNIX System Service (USS) profile and access to the /TMP directory in OMVS.

Example 8-2 shows how Q Apply reports a DB2 authority failure in the system log and stops processing.

*Example 8-2 Q Apply DB2 authority failure report*

---

```
IEF695I START APPLDWH1 WITH JOBNAME APPLDWH1 IS ASSIGNED TO USER QREP , GROUP STCGROUP
2008-04-11-02.43.32.002229 ASN8999D "Q Apply" : "N/A" :
ASN0531E "Q Apply" : "REDDWH" : "Initial" : The program could
not open the plan. The SQL return code is "8", the reason code is
"f30034", the subsystem name is "D912", and the plan name is
"ASNQA910".
ASN0573I "Q Apply" : "REDDWH" : "Initial" : The program was stopped.
IEF142I APPLDWH1 APPLDWH1 - STEP WAS EXECUTED - COND CODE 0012
```

---

In our implementation, we used the same Q Capture started task for Q replication and event publishing. Example 8-3 shows our procedure and its key elements:

- ▶ CAPTURE\_SERVER=D911 indicates the DB2 subsystem on which Q Capture works and must contain the Q Capture replication control tables.
- ▶ capture\_schema=REDDWH is the schema of the Q Capture control tables as defined during the creation of the subscription.

We do not describe all the working parameters of Q Capture in this book. Refer to *WebSphere Information Integrator Q Replication: Fast Track Implementation Scenarios*, SG24-6487, for further details.

During this example, the same Q Capture schema was used for Q replication and event publishing, but this is not mandatory. They can be defined into two different schemas. A Q Capture started task can only serve a single schema.

*Example 8-3 Procedure Sample for Q Capture*

---

```
//CAPTDWH1 JOB NOTIFY=CRIS,
//          MSGCLASS=H,MSGLEVEL=(1,0),
//          REGION=OM,TIME=NOLIMIT
//*****
//*
//*   Sample JCL for QCapture for schema REDDWH
//*
//*****
//QCAP      EXEC PGM=ASNQCAP,
// PARM='CAPTURE_SERVER=D911 capture_schema=REDDWH startmode=warmsi'
//STEPLIB DD DISP=SHR,DSN=SYS1.REP.V910.SASNLOAD
//          DD DISP=SHR,DSN=SYS1.DSN.V910.SDSNLOAD
//CAPSPILL DD DSN=&&CAPSPL,DISP=(NEW,DELETE,DELETE),
//          UNIT=VIO,SPACE=(CYL,(50,70)),
//          DCB=(RECFM=VB,BLKSIZE=6404)
//MSGSD    DD PATH='/local/REP910/db2rep1_09_01/msg/En_US/db2asn.cat'
//CEEDUMP DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD DUMMY
```

---

Example 8-4 shows our Q Apply procedure. Notice that we used another DB2 subsystem as the apply server and the same apply schema name, which is defined at the target server.

*Example 8-4 Procedure sample for Q Capture*

---

```
//APPLDWH1 JOB NOTIFY=CRIS,
//          MSGCLASS=H,MSGLEVEL=(1,0),
//          REGION=OM,TIME=NOLIMIT
//*****
//*
//*   Sample JCL for Q Apply for schema REDDWH
//*
//*****
//APPLDWH1 EXEC PGM=ASNQAPP,
//  PARM='APPLY_SERVER=D912 APPLY_SCHEMA=REDDWH logstdout=y'
//STEPLIB DD DISP=SHR,DSN=SYS1.REP.V910.SASNLOAD
//          DD DISP=SHR,DSN=SYS1.DSN.V910.SDSNLOAD
//CEEDUMP DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD DUMMY
//*
```

---

Example 8-5 shows both started tasks running in our system. Both started tasks are needed only for a Q replication solution. Event publishing requires only Q Capture.

*Example 8-5 Started tasks for Q replication*

---

```
DEST=(ALL) OWNER=QREP SORT=Pos/D
JOBNAME StepName ProcStep JobID Owner
APPLDWH1 APPLDWH1 APPLDWH1 STC10334 QREP
CAPTDWH1 CAPTDWH1 QCAP STC10337 QREP
```

---

## Creating the WebSphere MQ objects

In this section, we show examples of how to setup a *local* Q replication and event publishing infrastructure.

**Note:** The description applies to a local replication. That is when all the WebSphere MQ objects, including the MQ Managers, are local.

When a Q Capture program replicates data to a Q Apply program on the same system, you need only one queue manager. You can use the same local queue for the send queue and receive queue, and the two programs can share one local administration queue. You do not need remote queue definitions, transmission queues, or channels.

The following WebSphere MQ objects are required for our Q replication and event publishing implementation:

- ▶ Common objects:
  - One queue manager to which both the Q Capture and Q Apply program connects
  - One local queue to serve as the administration queue for both the Q Capture and Q Apply program
  - One local queue to serve as the restart queue
  - A spill queue

- For Q replication, one local queue to serve as both the send and receive queue
- For event publishing, one local queue to serve as the receive queue

Example 8-6 shows the queues that we defined for our implementation.

*Example 8-6 MQSeries queues used in our implementation*

Name	Type	Disposition
<> RED*	QUEUE	PRIVATE Q801
REDDWH.ADMINQ	QLOCAL	QMGR Q801
REDDWH.D911.OLTP.EVPUB	QLOCAL	QMGR Q801
REDDWH.D911.OLTP.EVPUB.DEL	QLOCAL	QMGR Q801
REDDWH.D911.OLTP.EVPUB.XML	QLOCAL	QMGR Q801
REDDWH.D911.TO.D912.QREP	QLOCAL	QMGR Q801
REDDWH.RESTARTQ	QLOCAL	QMGR Q801
REDDWH.SPILLQ	QMODEL	QMGR Q801

The utilization of the queues is listed as follows:

**REDDWH.ADMINQ** Used by the Q Apply program to communicate with the Q Capture program at the source.

**REDDWH.D911.OLTP.EVPUB**  
Event publishing target queue, default message definition.

**REDDWH.D911.OLTP.EVPUB.DEL**  
Event publishing target queue, delimited format for messages.

**REDDWH.D911.OLTP.EVPUB.XML**  
Event publishing target queue, XML message format.

**REDDWH.D911.TO.D912.QREP**  
Transmission and receive queue for Q replication.

**REDDWH.RESTARTQ**  
Stores restart information for the Q Capture program.

**REDDWH.SPILLQ** A model queue definition. Spill queues are created dynamically to hold any transactions that arrive from the source while the target table is loaded.

Example 8-7 shows a job control language (JCL) sample for the creation of the common MQ queues.

*Example 8-7 Creating common MQSeries objects*

```
//CRISQDEF JOB (),'QUEUE DEF SAMPLE',
//      REGION=OK,NOTIFY=CRIS,
//      MSGCLASS=X,
//      CLASS=C
//*****
//*      WS Replication Server for z/OS
//*****
//CRTQ    EXEC PGM=CSQUTIL,PARM='Q801'
//STEPLIB DD DSN=SYS1.MQM.Q801.CSQLOAD,DISP=SHR
//      DD DSN=SYS1.MQM.V600.SCSQANLE,DISP=SHR
//      DD DSN=SYS1.MQM.V600.SCSQAUTH,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN   DD *
//      COMMAND DDNAME(CMDINP)
//*
//CMDINP DD *
```

```

DEFINE REPLACE                                     +
    QLOCAL('REDDWH.ADMINQ')                       +
    DESCR('SAMPLE ADMIN LOCAL QUEUE')             +
    PUT(ENABLED)                                   +
    GET(ENABLED)                                   +
    SHARE                                           +

DEFINE REPLACE                                     +
    QLOCAL('REDDWH.RESTARTQ')                     +
    DESCR('SAMPLE RESTART LOCAL QUEUE')           +
    PUT(ENABLED)                                   +
    GET(ENABLED)                                   +
    SHARE                                           +

DEFINE REPLACE                                     +
    QMODEL('REDDWH.SPILLQ')                       +
    DEFTYPE(PERMDYN) +                             +
    DEFSOPT(SHARED)                               +
    MAXDEPTH(500000)                              +
    MAXMSGL(4194304)                              +
    MSGDLVSQ(FIFO)                                +

/*

```

---

The creation jobs may return RC=0 even if a syntax errors is in the define commands. Carefully check the job output shown in Example 8-8 for failures.

*Example 8-8 MQ queue creation output sample*

```

DEFINE REPLACE                                     +
    QLOCAL(REDDWH.RESTARTQ')                       +
    DESCR('SAMPLE RESTART LOCAL QUEUE')           +
    PUT(ENABLED)                                   +
    GET(ENABLED)                                   +
    SHARE                                           +

CSQN205I  COUNT=      8, RETURN=00000008, REASON=FFFFFFFF
CSQ9002E <Q801 Unbalanced parentheses following 'QLOCAL'
CSQ9003E <Q801 'QLOCAL' parameter contains unbalanced apostrophes
CSQ9031E <Q801 Syntax error following 'QLOCAL'
CSQ9003E <Q801 'REDDWH.RESTARTQ' parameter contains unbalanced
apostrophes
CSQ9026E <Q801 'QLOCAL' parameter does not satisfy name rules
CSQ9023E <Q801 CSQ9SCND 'DEFINE QLOCAL' ABNORMAL COMPLETION

```

---

## MQSeries requirements for Q replication

In addition to the common administrative and restart queue, Q replication needs one local queue to serve as both the send queue (Q Capture) and receive queue (Q Apply).

This queue must be defined with the following characteristics:

- PUT (ENABLED)** Allows Q Capture to PUT messages in the queue
- MAXMSGL(n)** *n* defines the maximum message length
- DEFPSIST(YES)** The queue uses persistent messages. That is, they are logged and then recoverable. The MQSeries log can become a performance bottleneck on stressed systems. You might consider techniques for improving logging throughput, such as VSAM data striping for logging. We recommend that you use persistent messages for recoverability.

<b>INDXTYPE(MSGID)</b>	Allows the queue manager to maintain an index of message identifiers.
<b>SHARE</b>	More than one application instance can get messages from the queue.
<b>GET(ENABLED)</b>	Allows Q Apply to GET messages from the queue.
<b>MSGDLVSQ(PRIORITY)</b>	Messages on the queue are delivered in FIFO order within priority. It is the recommended default for the receive queue even though Q replication messages are not prioritized.

Example 8-9 shows the JCL that we used for the creation of this queue.

*Example 8-9 JCL sample for the creation of the Q replication queue*

---

```
//CRISQDEF JOB (), 'QUEUE DEF SAMPLE',
//          REGION=OK, NOTIFY=CRIS,
//          MSGCLASS=X,
//          CLASS=C
//*****
/*      WS Replication Server for z/OS V91      */
//*****
//CRTQ     EXEC PGM=CSQUTIL, PARM='Q801'
//STEPLIB  DD DSN=SYS1.MQM.Q801.CSQLOAD, DISP=SHR
//          DD DSN=SYS1.MQM.V600.SCSQANLE, DISP=SHR
//          DD DSN=SYS1.MQM.V600.SCSQAUTH, DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
//          COMMAND DDNAME(CMDINP)
/*
//CMDINP   DD *
DEFINE REPLACE
    QLOCAL(REDDDWH.D911.TO.D912.QREP)
    DESCR('LOCAL QUEUE - QREPLICATION')
    PUT(ENABLED)
    GET(ENABLED)
    MAXMSGL(4194304)
    SHARE
    DEFSOPT(SHARED)
    DEFPSIST(YES)
    INDXTYPE(MSGID)
/*
```

---

## MQSeries requirements for event publishing

In addition to the common queues, one local queue to serve as the target for the publishing is needed. In our scenario, this queue is the input to InfoSphere DataStage. Example 8-10 shows the JCL that is used for the creation of these queues.

*Example 8-10 Creating MQSeries queues for Q replication and event publishing*

---

```
//CRISQDEF JOB (), 'QUEUE DEF SAMPLE',
//          REGION=OK, NOTIFY=CRIS,
//          MSGCLASS=X,
//          CLASS=C
//*****
/*      WS Replication Server for z/OS      */
//*****
//CRTQ     EXEC PGM=CSQUTIL, PARM='Q801'
//STEPLIB  DD DSN=SYS1.MQM.Q801.CSQLOAD, DISP=SHR
//          DD DSN=SYS1.MQM.V600.SCSQANLE, DISP=SHR
//
```

---



```
//          DD DSN=SYS1.MQM.V600.SCSQAUTH,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
          COMMAND DDNAME(CMDINP)
/*
//CMDINP   DD *
DEFINE REPLACE
          QLOCAL(REDDBWH.D911.OLTP.EVPUB)
          DESCR('LOCAL QUEUE - EVENT PUBLISHING')
          PUT(ENABLED)
          GET(ENABLED)
          MAXMSGL(4194304)
          SHARE
          DEFSOPT(SHARED)
          DEFPSIST(YES)
          INDXTYPE(MSGID)
/*
```

---

## DB2 Replication Center

The replication and event publishing definitions are created and updated by using the DB2 Replication Center. The Replication Center supports administration for DB2-to-DB2 replication environments and administration for replication between DB2-and-non-DB2 relational databases. The DB2 Replication Center is part of the DB2 Control Center set of tools. The process described in this section was done by using version DB2 v9.5.0.808 of the DB2 Replication Center.

The subscription definitions and the running parameters of the replication server are stored in a series of control tables. You create these control tables early in the configuration process from the DB2 Replication Center, or you can use the DDL samples provided by IBM in *hlq.SASNSAMP(ASNQCTLZ)*. The updates of the contents of these tables are executed by the Replication Center using SQL. This code is always shown before execution.

Figure 8-3 shows the Replication Center launchpad. This panel contains a brief description of each of the replication techniques that you can control from this tool:

- ▶ Q replication
- ▶ Event publishing
- ▶ SQL replication

Some of the operations that can be done from the Replication Center require that a DB2 Administration Server instance be installed on your target z/OS system. This installation is beyond the scope of this section.

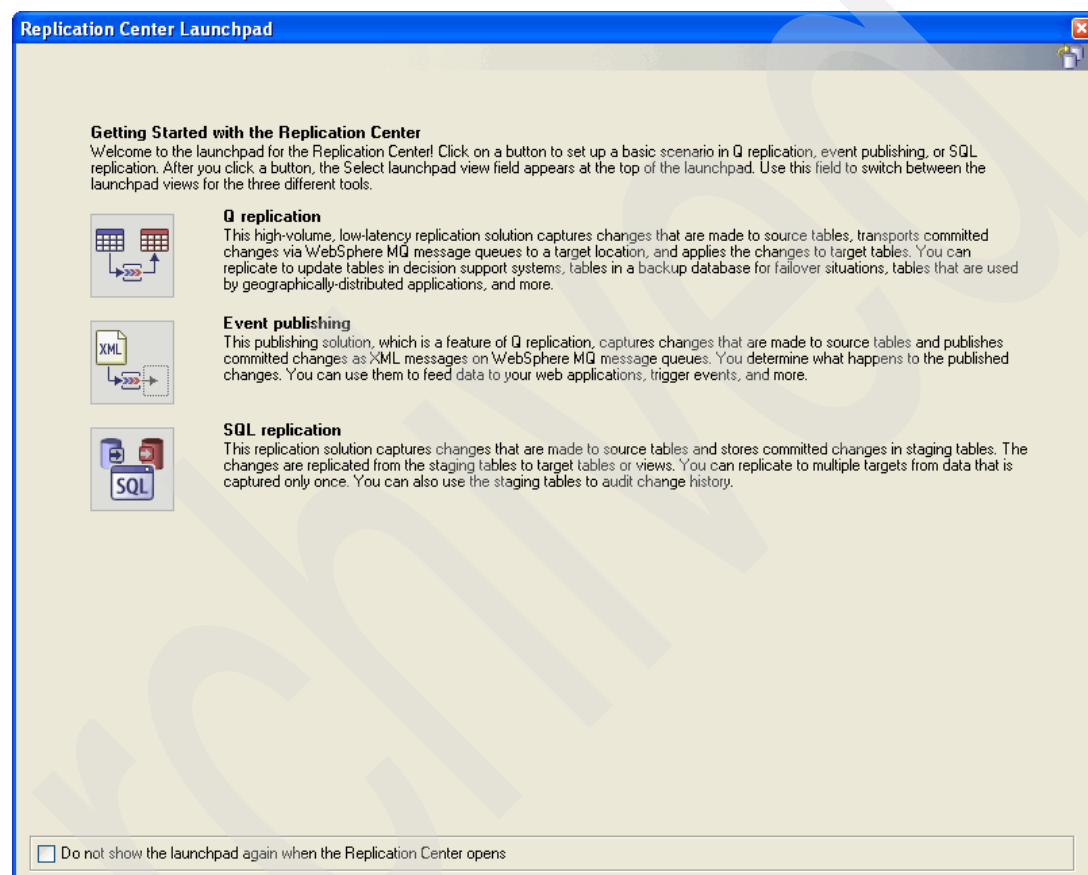


Figure 8-3 DB2 Replication Center launchpad

Figure 8-4 shows the main panel of the tool. Here we can choose one of the possible solutions.

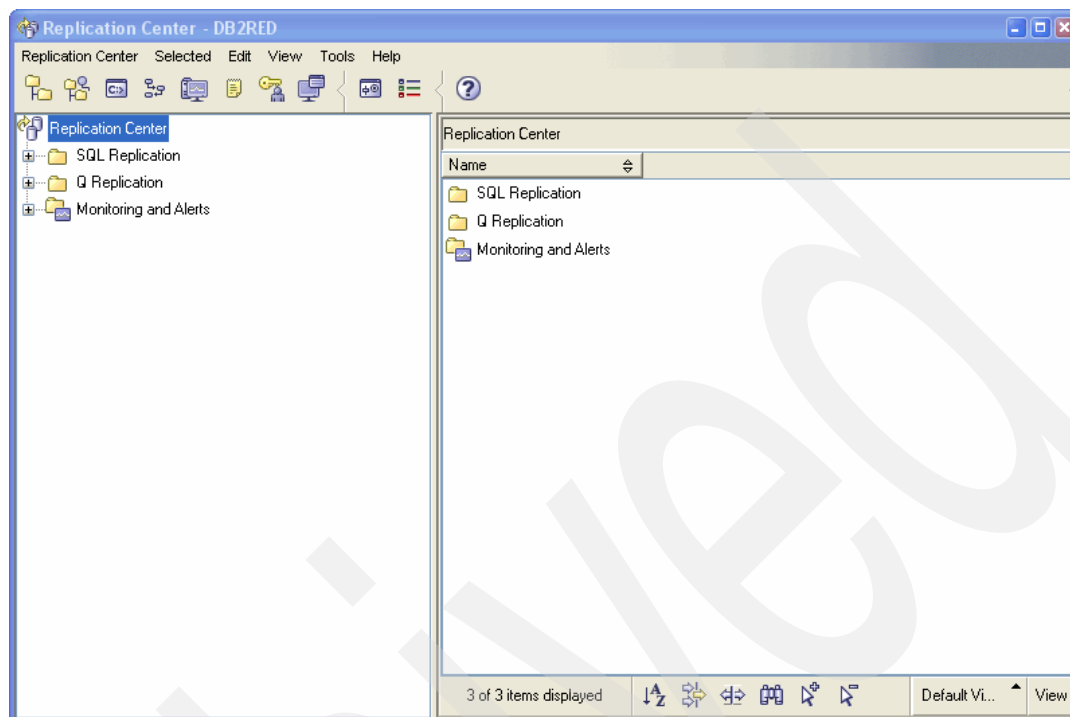


Figure 8-4 DB2 Replication Center main panel

## 8.2.2 Configuring Q replication subscriptions

A unidirectional Q replication scenario requires the following definitions:

- Replication queue maps

These maps define which MQSeries queues are used for storing the captured data and making it available to an Apply program or a stored procedure. You must define at least one queue map.

- Q subscriptions

You must define one Q subscription for each source and target pair. For performance reasons, consider spreading subscriptions on more than one MQSeries Queue Manager.

In the remainder of this section, we show the definitions of these elements by using the DB2 Replication Center.

Figure 8-5 shows the Replication Center Launchpad for Q replication. Steps 1 and 2 of the Replication Center Launchpad for Q replication create the required Q Capture and Q Apply Control Tables that are needed for the functioning of the product. While the description of these steps is not shown, you must complete these steps before continuing with the remainder of the task. You can create these tables from the launchpad or by using the sample job *hlq.SASNSAMP(ASNQCTLZ)* that is provided with the product.

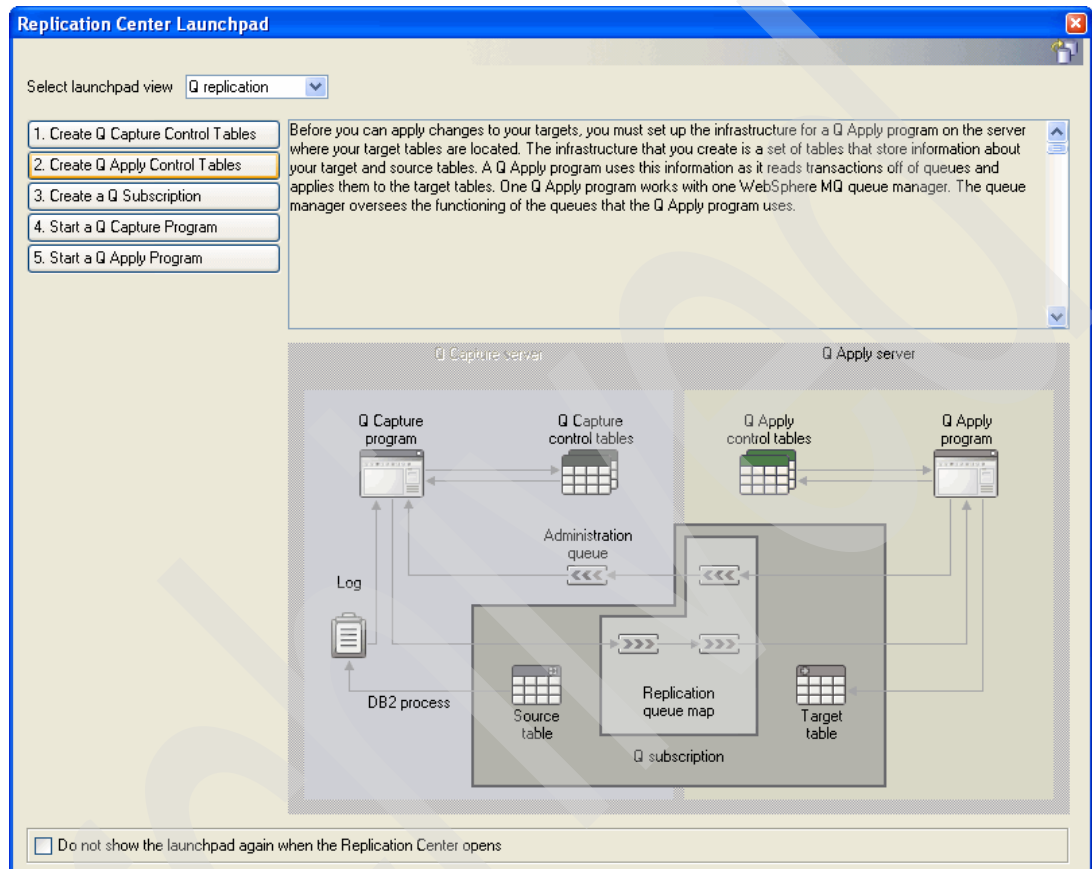


Figure 8-5 Replication Center Launchpad for Q replication

To configure Q replication subscriptions:

1. From the Replication Center Launchpad, click **3. Create a Q Subscription** to start the Create Q Subscriptions launchpad.

2. Click the **2. Replication** button on the left. In the Which type of replication? pane (Figure 8-6) on the right, specify the type of replication to configure:
  - Unidirectional
  - Bidirectional
  - Peer-to-peer, two servers
  - Peer-to-peer, three or more servers

We select **Unidirectional** for our data warehouse. With this type of replication, changes on the source table are replicated to the target unidirectionally.

Click **Next** to continue the configuration process.

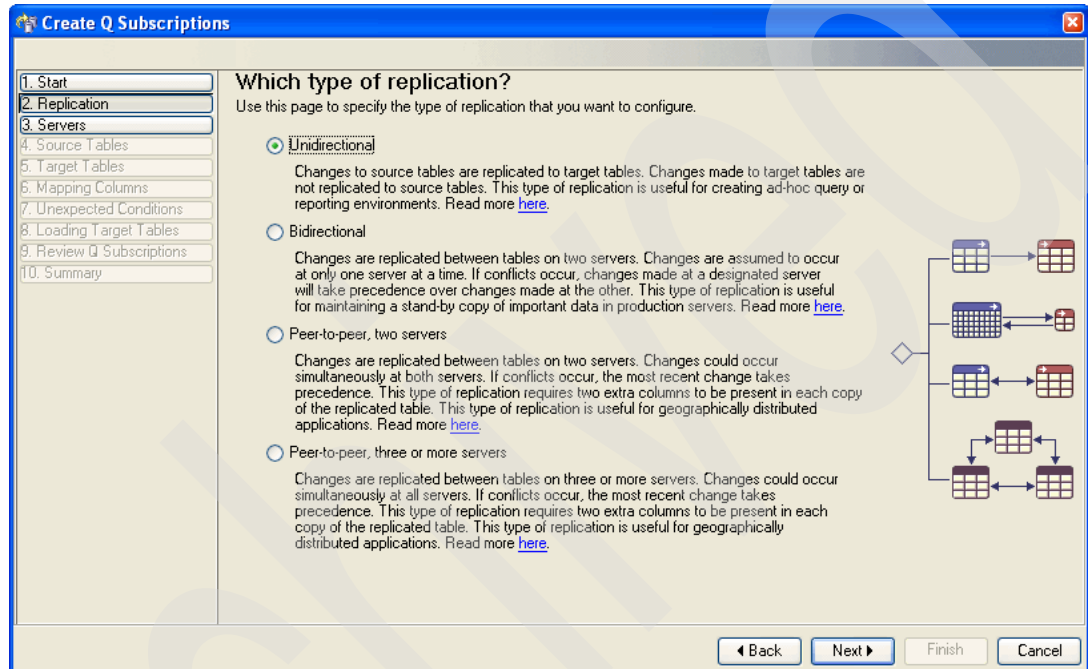


Figure 8-6 Specifying the type of replication

3. Click the **3. Servers** button on the left. In the Which source and target servers? pane (Figure 8-7) on the right, define the source and target servers for the replication configuration.

Our implementation replicates changes from a DB2 9 New Function Mode subsystem to another one, both located in the same LPAR and none in data sharing.

You must catalog these subsystems and create the replication control tables before you can select them as the source of the target for replication. You can do this easily by using the Replication Center Launchpad for Q replication (Figure 8-5 on page 172).

The screenshot shows the 'Create Q Subscriptions' dialog box with the '3. Servers' tab selected in the left-hand pane. The main area is titled 'Which source and target servers?' and contains instructions: 'Use this page to specify the source and target servers. You must also specify the Q Capture and Q Apply programs to use for capturing data at the source and applying it at the target and a replication queue map that the Q Capture and Q Apply program can use to communicate.'

The 'Source' section has a text box: 'This server must contain the tables that you want to replicate.' Below it are two fields: 'Server' with the value 'D\WHD911' and 'Schema' with the value 'REDDWH'.

The 'Target' section has a text box: 'The target tables to which you want to replicate might or might not exist on this server. If they do not, they will be created for you.' Below it are two fields: 'Server' with the value 'D\WHD912' and 'Schema' with the value 'REDDWH'.

The 'Queues' section has a text box: 'Select a replication queue map that the Q Capture program and Q Apply program can use to communicate.' Below it is a field for 'Replication queue map' which is currently empty.

At the bottom right are four buttons: 'Back', 'Next', 'Finish', and 'Cancel'.

Figure 8-7 Defining the source and target servers

For Queues, define and select a *replication queue map*. The replication queue map identifies the MQ queues that a Q Capture program and a Q Apply program use to transport data and communicate. Each replication queue map identifies one of each of the following WebSphere MQ queues:

- Send queue

This is the queue where the Q Capture program sends source data and informational messages. Messages that are sent using a send queue that is identified by a replication queue map must be in compact format, which is the message format that the Q Apply program reads.

- Receive queue

This is the queue where the Q Apply program receives source transactions before applying them to the target table or passing them to a stored procedure.

- Administration queue

This is the queue that the Q Apply program uses to send control messages to the Q Capture program.

You can use a single replication queue map to transport data for one or more Q subscriptions. Also, a single Q Capture program can write data to many send queues. A single Q Apply program can read and apply data from many receive queues. For our implementation, we use the same queue object for the definition of the send and receive queue.

- a. On the **General** page of the Create Replication Queue Map window (Figure 8-8), name the queue map, which in our case, is DWHD911\_TO\_DWHD912. Introduce the send and receive queue name, which is the same in our case, and the administration queue name. The other fields are filled with the name of the objects that we defined at the beginning of this chapter.

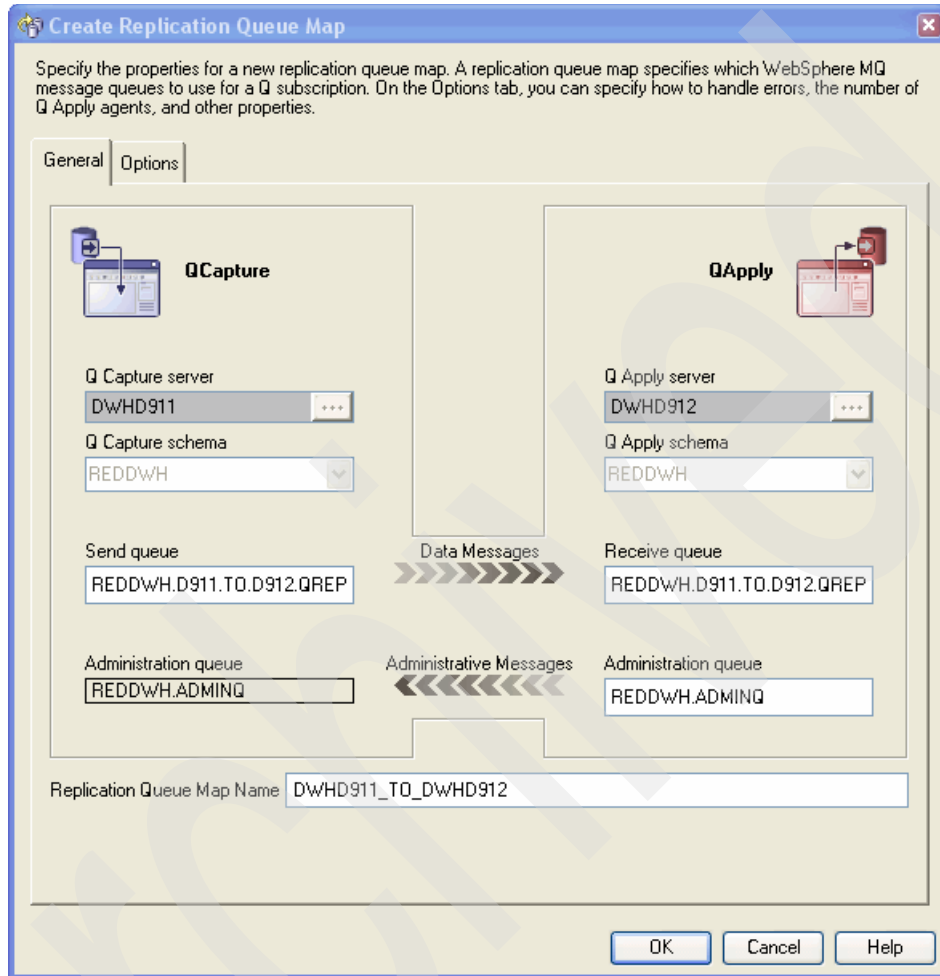


Figure 8-8 Creation of a Q replication queue map

- b. Click the **Options** tab (Figure 8-9) of the Create Replication Queue Map window. Specify the properties to change for a queue map. These replication properties specify how the Q Capture and Q Apply programs process messages that use the send queue and the receive queue. For each replication queue map, you can specify the following properties:
- The maximum size of a message that the Q Capture program can put on the send queue. This limit is independent of the queue maximum message length, but this limit must be equal to or less than the queue maximum message length.
  - How the Q Capture program responds if an error occurs in the queue.
  - The frequency at which the Q Capture program sends messages on this queue to tell the Q Apply program that the Q Capture program is still running when there are no changes to replicate.
  - The number of threads for each Q Apply browser to be created to apply transactions to target tables or pass transactions to stored procedures to manipulate the data.
  - The amount of memory that the Q Apply program can use to process messages from the receive queue.

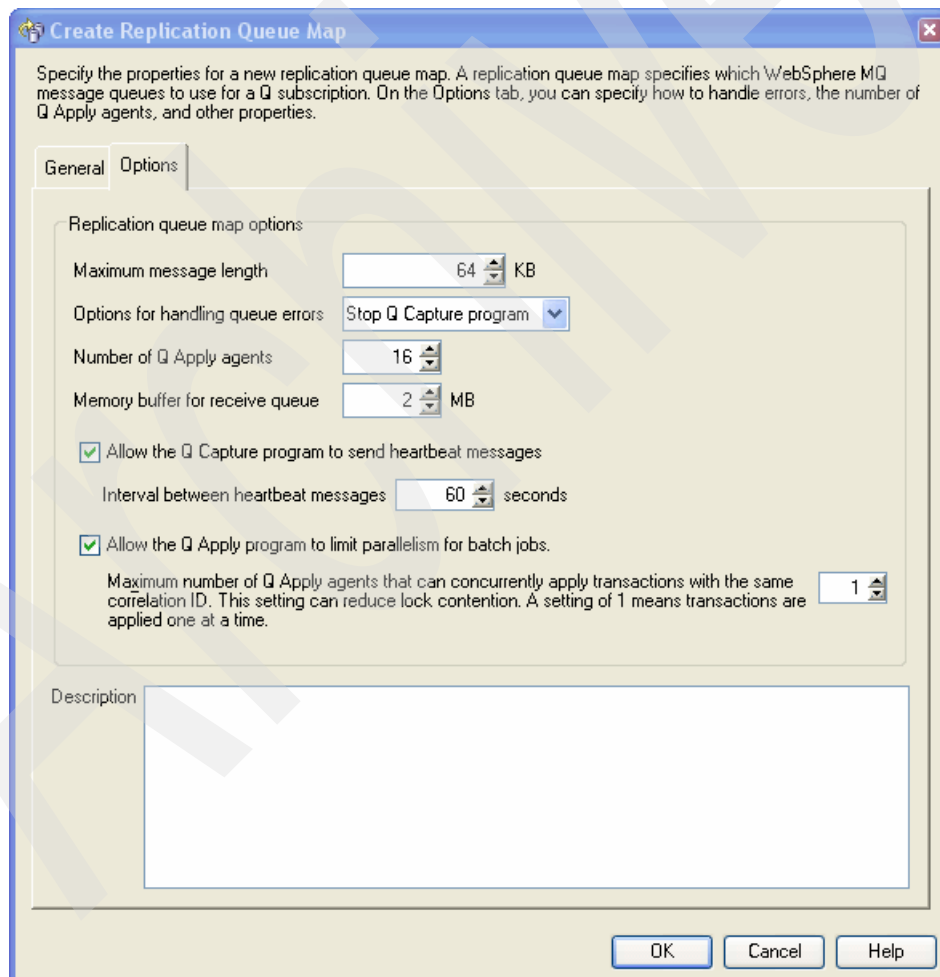


Figure 8-9 Options for the Q replication queue map

Figure 8-10 on page 177 shows the completed Which source and target servers? pane after you provide the servers and queue map definitions.



**1. Start**  
**2. Replication**  
**3. Servers**  
**4. Source Tables**  
5. Target Tables  
6. Mapping Columns  
7. Unexpected Conditions  
8. Loading Target Tables  
9. Review Q Subscriptions  
10. Summary

### Which source and target servers?

Use this page to specify the source and target servers. You must also specify the Q Capture and Q Apply programs to use for capturing data at the source and applying it at the target and a replication queue map that the Q Capture and Q Apply program can use to communicate.

**Source**  
This server must contain the tables that you want to replicate.  
Server: D\WHD911  
Schema: REDDWH

**Target**  
The target tables to which you want to replicate might or might not exist on this server. If they do not, they will be created for you.  
Server: D\WHD912  
Schema: REDDWH

**Queues**  
Select a replication queue map that the Q Capture program and Q Apply program can use to communicate.  
Replication queue map: D\WHD911\_TO\_D\WHD912

◀ Back   Next ▶   Finish   Cancel

Figure 8-10 Defined source and target servers

- Click the **4. Source Tables** button on the left. In the Select Source Tables window, define the source tables. For our data warehouse system, we replicate the LINEITEM and ORDER tables, as shown in Figure 8-11.

**Select Source Tables**

**Search criteria**

Column	Comparison	Values
Creator	LIKE	OLTP_B
Creator	<>	SYSIBM
Database	LIKE	

☒ Meet all conditions   ☐ Meet any conditions

Retrieve All   Retrieve   Count

Name	Creator	Database	Table space	Total
LINEITEM	OLTP_B	DBOLTP	TSLINEIT	
"ORDER"	OLTP_B	DBOLTP	TSORDERS	
SHIPMENT	OLTP_B	DBOLTP	TSSHIP	

OK   Cancel   Help

Figure 8-11 Selecting the source tables for Q replication

5. Click the **5. Target Tables** button on the left. In the Which profile settings for target tables? pane (Figure 8-12), profile the settings for the target tables. You can define one of the following targets:

- Target is a table

We implement this option in our scenario and replicate the changes to a table. As shown in the lines that follow, options can be applied to control how the changes are replicated.

- Target is a consistent-change-data (CCD) table

By using a CCD table as your target type, you can keep a history of source changes. For example, you can track before and after comparisons of the data, when changes occurred, and which user ID updated the source table.

**Create Q Subscriptions**

1. Start  
2. Replication  
3. Servers  
4. Source Tables  
5. Target Tables  
6. Mapping Columns  
7. Unexpected Conditions  
8. Loading Target Tables  
9. Review Q Subscriptions  
10. Summary

### Which profile settings for target tables?

Use this page to review the information that the Replication Center will use to create target tables for your Q subscriptions. If target tables exist that match the profile settings, the Replication Center will use those tables.

These settings are stored in the target object profile for the Q Apply server where the target tables will be located. To change any of the settings or to view more detailed information about a target object profile, click [Change](#).

By default, all columns and rows in the source tables will be replicated. On the Review Q Subscriptions page, you can specify a subset of columns and rows to replicate from a source table by selecting the corresponding Q subscription and clicking Properties.

☒ Target is a table

☐ Target is a consistent-change-data (CCD) table

A CCD table can be used to store changes that have occurred in the source for auditing purposes. [Learn more about using CCDs here](#)

CCD properties:

D:\WHD912

Target tables:

- Schema will be the same as the source table
- Name will be the same as the source table

Target table indexes:

- Schema will be the same as the target table
- Name will be IX + <target table name>

Target table spaces:

- Each table that is created on this server will go in its own table space
- Name will be TS + <target table name>

[Change](#)

◀ Back   Next ▶   Finish   Cancel

Figure 8-12 Profile settings for target tables

You can click the **Change** button to change the properties of the target tables and table spaces. While this process is not shown here, it is straightforward. In the Manage Target Object Profiles window (Figure 8-13), you can specify whether to create target tables in new or existing table spaces. You can also specify operational parameters of the table space, including whether the target table space should use the same partitioning as the source table space.

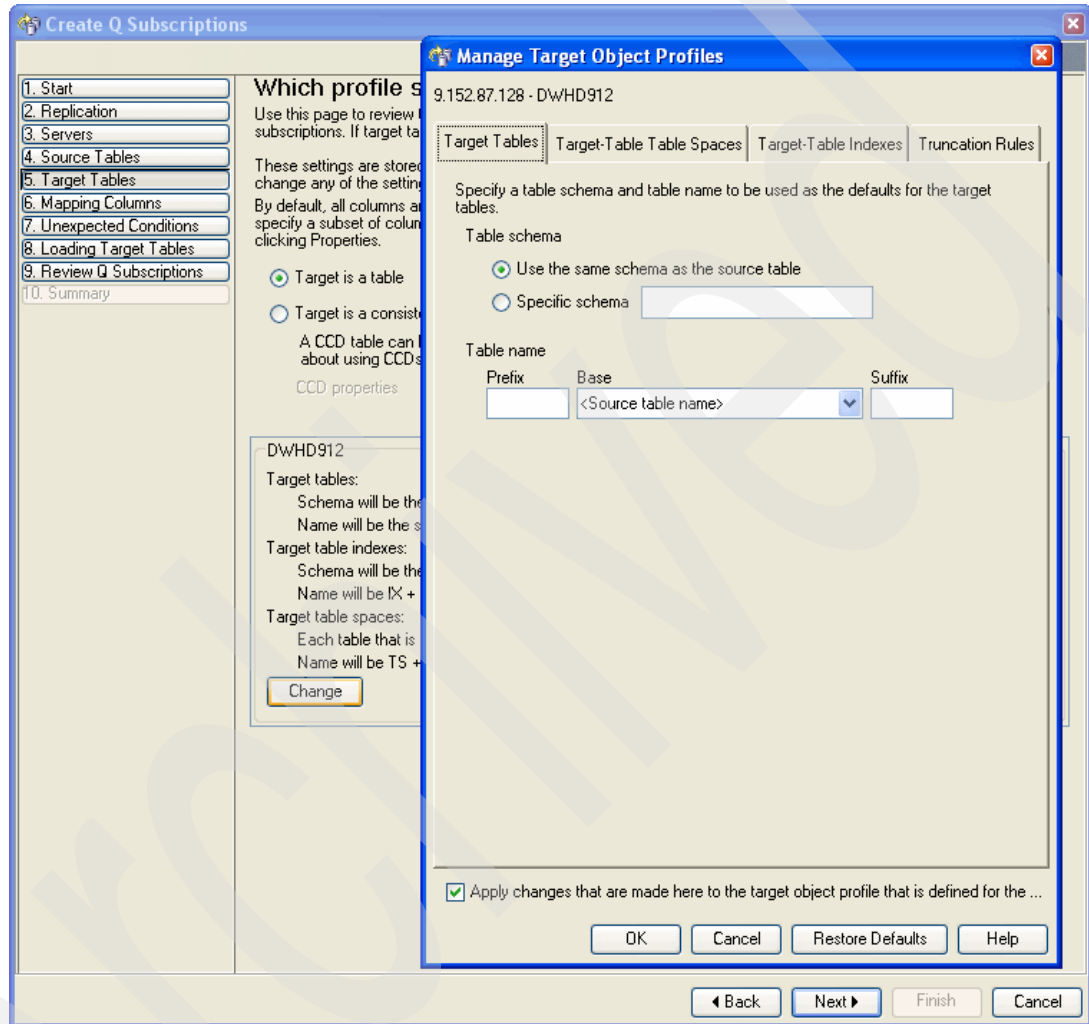


Figure 8-13 Manage Target Object Profiles window

- Click the **6. Mapping Columns** on the left. In the Which source columns map to which target columns? pane (Figure 8-14), map column names from the source to the target table, if their name is not the same.

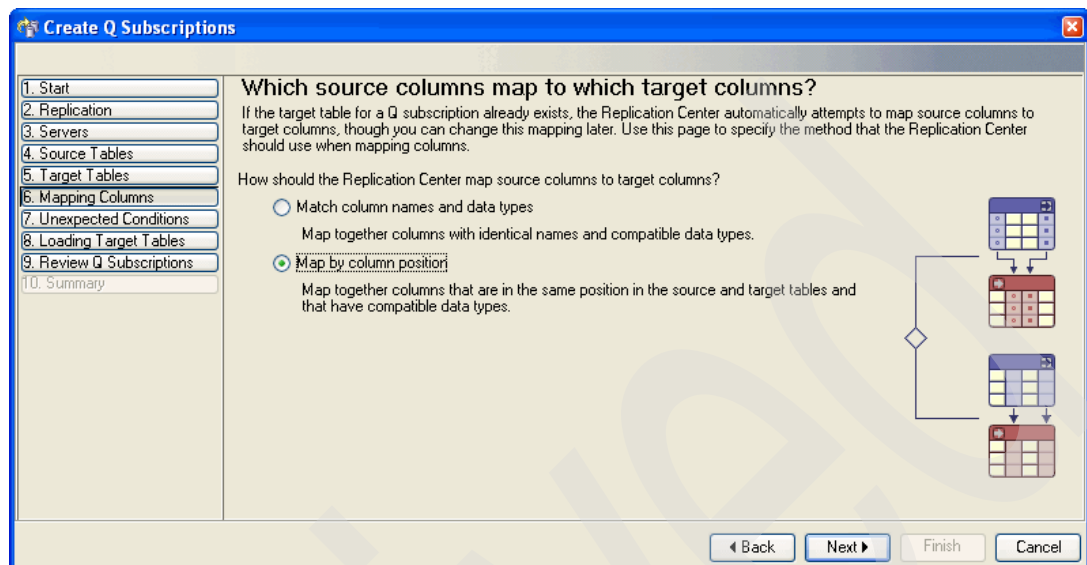


Figure 8-14 Mapping source to target columns

For existing target tables or for stored procedures, specify how you want the data from source columns to map to target columns or to parameters in a stored procedure. If a target table does not exist, then the Replication Center creates the target table with the same columns that the source table has.

The source table cannot replicate more columns than are in the target table or more columns than parameters in the stored procedure. However, the target table can contain more columns than the number of source columns that you selected for replication.

When you create Q subscriptions, you can choose from the following options for mapping source columns to target columns:

- Map by column name and data type  
Each column in the source table is mapped to the column in the target table (or parameter in the stored procedure) that has an identical name and data type.
- Map by column position and data type  
The first column in the source table that you selected for replication is mapped to the first column in the target table or parameter in the stored procedure. Then the second column in the source table that you selected for replication is mapped to the second target column or parameter, and so on. The columns in the target are mapped from left to right. The data types of each pair of mapped columns must be the same.

In our example, both the source and target table columns were named in the same way. We implemented mapping by column name and data type.

- Click the **7. Unexpected Conditions** button. The How should the Q Apply program respond to unexpected conditions? pane (Figure 8-15 on page 181) shows the available options for Q Apply to handle unexpected conditions. The Q Apply program updates the targets with changes that occur at the source table. If other applications are also making changes to the target, then the Q Apply program might encounter rows in the target that are different than expected. For example, the Q Apply program might try to update a row in the target that another application has already deleted. The option that you choose depends on the level of granularity at which you want to isolate and fix the problem.

In many scenarios, you are likely to want the Q Apply program to either force the change or ignore unexpected conditions in target data. However, in some scenarios, you might never expect problems in the target data and, therefore, might choose a different action, depending on the level at which you think you have to troubleshoot problems.

You can specify that the Q Apply program takes one of the following actions when it encounters unexpected conditions in target data:

- Force the change
- Ignore the unexpected condition
- Deactivate the corresponding Q subscription
- Have the Q Apply program stop reading from the corresponding receive queue
- Stop the Q Apply program

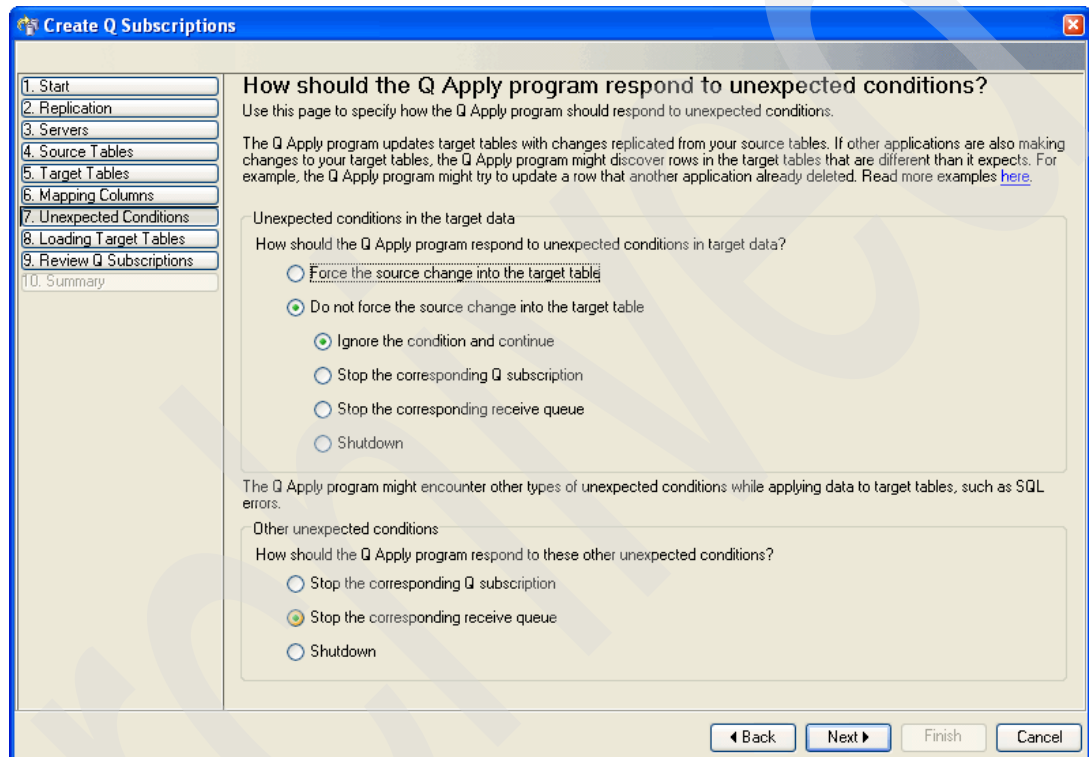


Figure 8-15 Q Apply management of unexpected conditions

8. Click the **8. Loading Target Tables** button on the left. In the How should target tables be loaded? pane (Figure 8-16 on page 182), you see the options for the initial load. When you create a Q subscription, choose among the following options for loading target tables with data from the source:

- Automatic load

The Q Apply program manages the loading of target tables. You can select which load utility the Q Apply program calls, or you can let the Q Apply program choose the best available utility for your operating system and version.

- Manual load

You handle the loading of target tables and then signal the replication programs when loading is done. This option is also known as *external load*.

- No load

You either do not load target tables or you load target tables outside the context of the replication programs.

**Important:** If you instruct APPLY to do the initial load, a Distributed Relational Database Architecture (DRDA) connection between source and target DB2 *must* exist. The load method used is CURSOR.

**Create Q Subscriptions**

1. Start  
2. Replication  
3. Servers  
4. Source Tables  
5. Target Tables  
6. Mapping Columns  
7. Unexpected Conditions  
8. Loading Target Tables  
9. Review Q Subscriptions  
10. Summary

### How should target tables be loaded?

Use this page to specify if and how your target tables will be loaded. When a Q subscription starts, the Q Apply program can initiate a load of the target tables with the utility or utilities that you specify.

Loading the target tables

How will the target table be loaded when a Q subscription starts?

☒ Automatic: The Q Apply program performs the load

How should the Q Apply program choose which load method to use?

☒ Best available: Let the Q Apply program choose a load method

☐ Always use LOAD FROM CURSOR

☐ Always use export and load

☐ Always use export and import

☐ Manual: You perform the load manually and inform the Q Apply program when the load is complete

☐ None: The target table is not loaded

☒ Start all Q subscriptions automatically

Selecting this option causes the Q Capture program to start your new Q subscriptions automatically the next time that the Q Capture program starts or is reinitialized.

◀ Back   Next ▶   Finish   Cancel

Figure 8-16 Q replication options for initial load

- Click the **9. Review Q Subscriptions** button on the left. In the Review and complete Q subscriptions pane (Figure 8-17), review, complete, or modify the definitions.

**Create Q Subscriptions**

1. Start  
2. Replication  
3. Servers  
4. Source Tables  
5. Target Tables  
6. Mapping Columns  
7. Unexpected Conditions  
8. Loading Target Tables  
9. Review Q Subscriptions  
10. Summary

### Review and complete Q subscriptions

Use this page to verify the Q subscriptions you are creating and to provide any information or corrections that are required. Q subscriptions that require additional input are marked with an error icon. To complete or correct a Q subscription, select it and click Properties.

✖ Information or correction required   ⚠ Warning   ✔ Valid

Name	Source Server	Source Owner	Source Name	Target Server	Target
✖ LINEITEM0001	DWHD911	OLTP_B	LINEITEM	DWHD912	
✖ ORDER0001	DWHD911	OLTP_B	ORDER	DWHD912	DWHD912

Properties  
Remove

2 of 2 items displayed

Default View   View

◀ Back   Next ▶   Finish   Cancel

Figure 8-17 Review and complete subscription definition panel

### 8.2.3 Configuring event publishing using the DB2 Replication Center

As described before, our Q replication and event publishing implementation will share the Q capture schema, and the starting tasks will be the same. You may consider separating the process on different schemas to improve parallelism.

The event publishing launchpad in Figure 8-18 is similar to the Q replication launchpad. As a reminder, an event publishing process publishes changes into a queue from where an application must extract the data. In our implementation, DataStage is used to exploit the information produced by event publishing. No Q Apply process is running for an event publishing implementation.

Step 1, Create Q Capture Control Tables is not necessary to do. We click step **2. Create a Publication**. The Create Publications launchpad opens.

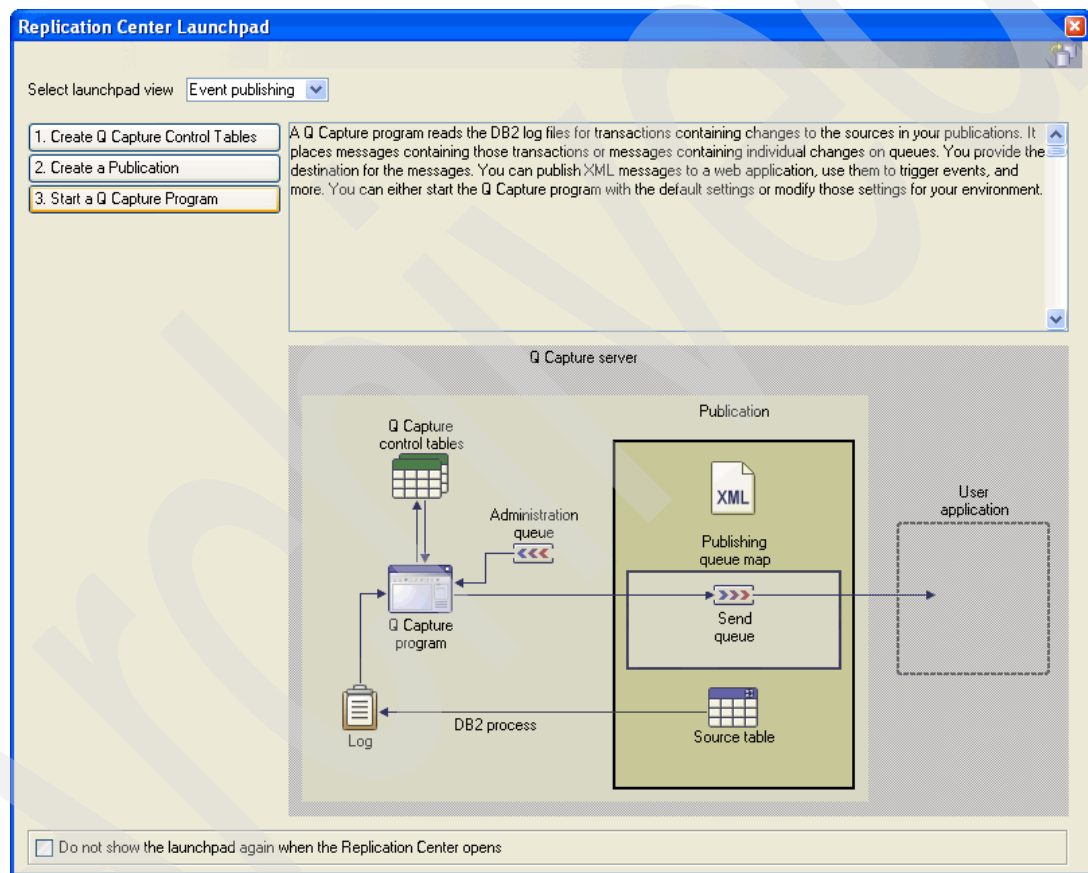


Figure 8-18 Event publishing - Replication Center Launchpad

To create publications:

1. Click the **Start** button to view the first pane, Creating publications (Figure 8-19), of the event publishing process. This pane shows a description of the tasks to perform.

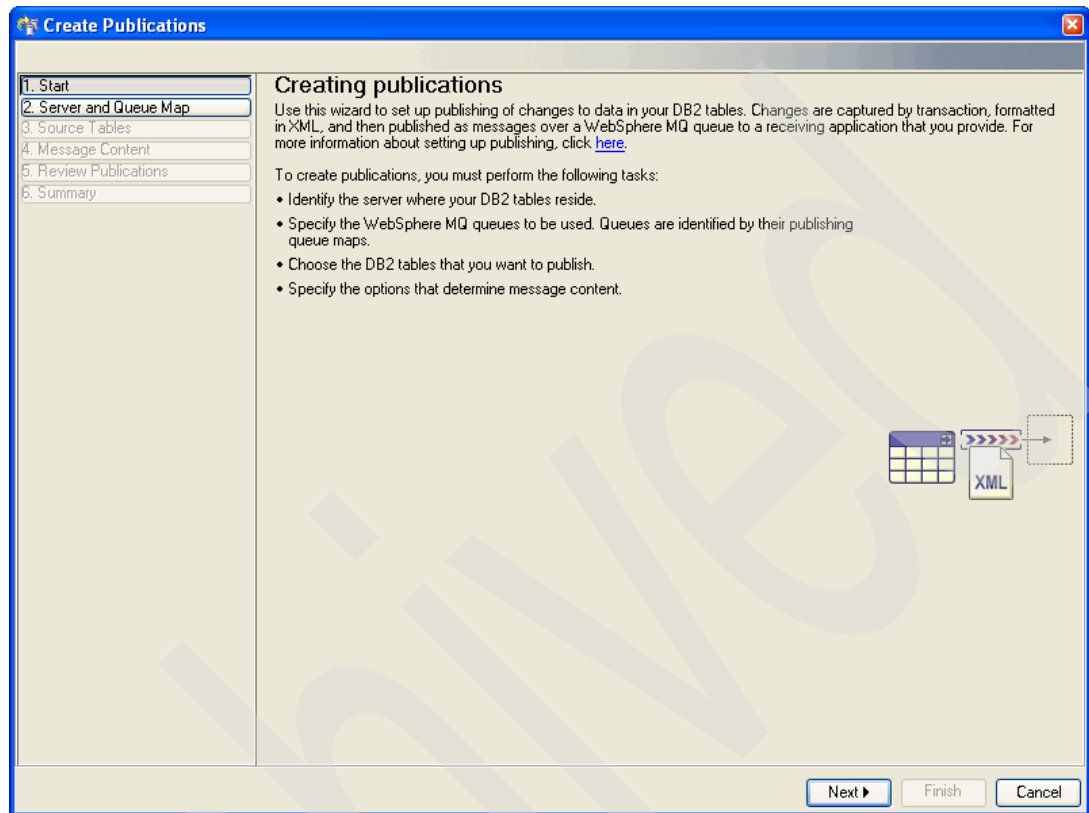


Figure 8-19 Creating publications for event publishing



2. Click the **2. Server and Queue Map** button on the left. In the Creating Publishing Queue Map window (Figure 8-20), as with Q replication, define a queue map for event publishing. However, this process is simpler for event publishing because only a send queue is needed. A receive queue is not defined for event publishing. In this panel, define the send queue and the message format on which event publishing will store the captured information.

**Create Publishing Queue Map - REDDWH**

9.152.87.128 - NDE27804 (DB2) - D\WHD911 - REDDWH

Specify the attributes for a new publishing queue map. The queue map specifies which WebSphere MQ message queue to use for a publication. You can also specify XML or delimited message format, maximum message size, and other attributes.

**General** Properties

Send queue: REDDWH.D911.OLTP.EVPUB

Type of message content: Row operation

Maximum message length: 64 KB

Options for handling queue errors: Stop Q Capture program

☒ Allow the Q Capture program to send heartbeat messages

Interval between heartbeat messages: 60 seconds

**Message Format**

☒ XML format

☐ Add a JMS-compliant (MQRFH2) message header to all messages that are put on this send queue (for JMS applications and message brokers).

☐ Keep Q Capture running after code page conversion errors. Publish the data in error in binary format.

☐ Delimited format

Code page for publishing delimited messages: D

Column delimiter: [dropdown]

Character string delimiter: [dropdown]

New line value: linefeed

Decimal point value: [dropdown]

Default

OK Cancel Help

Figure 8-20 Sample of a publishing queue map creation

The message format can be either XML or delimited. Both formats are supported by DataStage. We used the delimited format in our installation. Refer to “Event publishing message formats” on page 189 for more information and to learn how to work with them.

After the publishing queue map is created, we can continue with the creation of the event publishing subscription creation.

The Which Q Capture server and publishing queue map? pane (Figure 8-21) introduces the Q Capture server name, Q capture schema, and the publishing queue map.

**Create Publications**

1. Start  
2. Server and Queue Map  
3. Source Tables  
4. Message Content  
5. Review Publications  
6. Summary

### Which Q Capture server and publishing queue map?

Changes made to your source tables are captured by a Q Capture program. This program runs on the server where your source tables are located. This server is also called a Q Capture server. The Q Capture program is identified by the schema of the tables that it uses to store information about your source tables. (Note that the term server in Q replication refers either to a DB2 database on Linux, UNIX, or Windows, or to a DB2 UDB subsystem on z/OS.)

**Source**  
Select the Q Capture server where your source tables are located. Then, select the schema that identifies the Q Capture program that you want to publish changes to your source tables.

Q Capture server: DWH911  
Q Capture schema: REDDWH

When you create publications, you specify which WebSphere MQ queue the data will be published to. The queues that are available for publishing and the specific attributes of these queues are defined in publishing queue maps.

**Queues**  
Select a publishing queue map with the attributes that you want for this set of publications.

Publishing queue map: REDDWH

Diagram: A database icon (cylinder) is connected by an arrow to a server icon (box with a screen), which is then connected by an arrow to a queue icon (box with a double arrow).

Buttons: Back, Next, Finish, Cancel

Figure 8-21 Event publishing subscription creation

3. Click the **3. Source Tables** button on the left.
  - a. In the Select Source Tables window (Figure 8-22), continue the process by selecting the source tables.

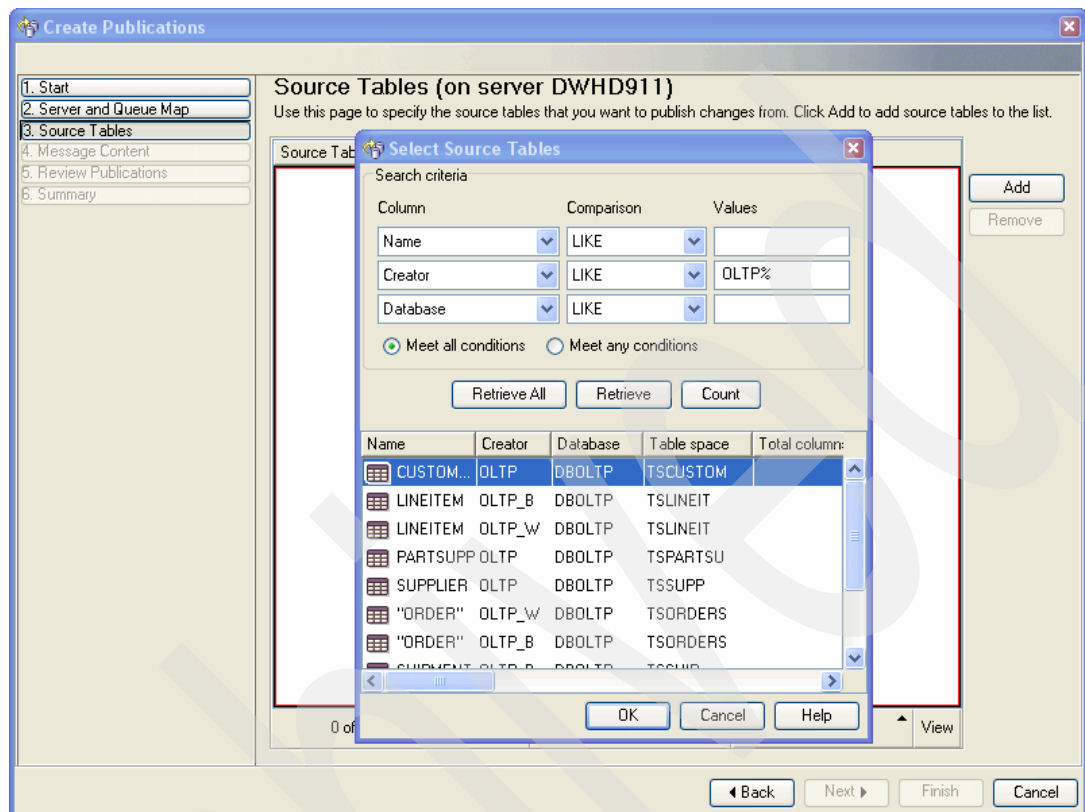


Figure 8-22 Selecting source tables for event publishing

- b. Select which columns to publish. You have the option to publish all the rows or to publish changes of a selection of rows, as shown in Figure 8-23. With this latter option, you can indicate whether a filter on the source table rows must be applied.

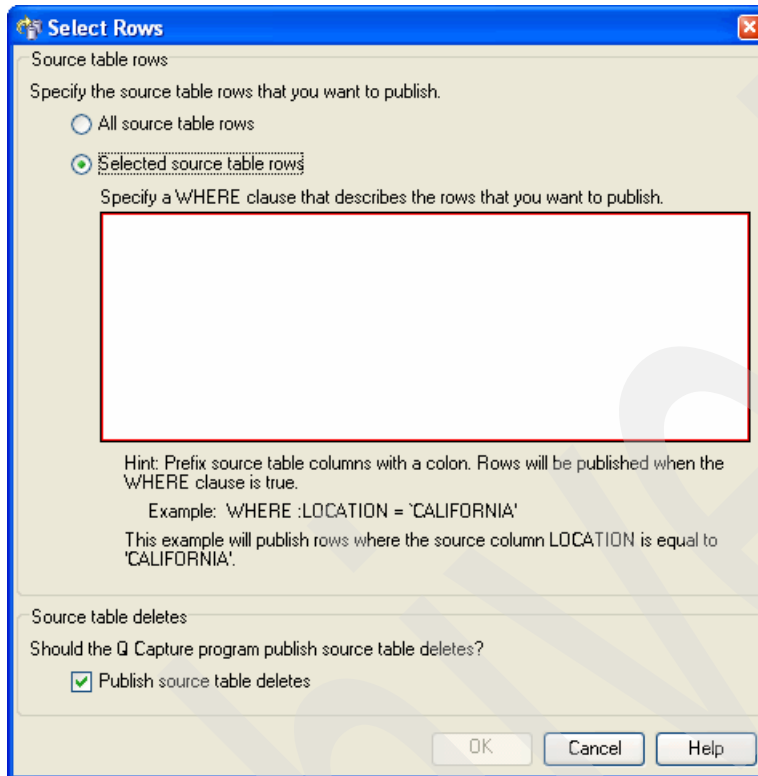


Figure 8-23 Selecting rows for event publishing

The filtering criteria is introduced in the form of a WHERE clause. Example 8-11 shows how a filter criteria can be applied to the source table Customer.

*Example 8-11 WHERE clause on table rows for event publishing*

---

```
WHERE COUNTRY = 'BELGIUM'
```

---

Alternatively, you can select the **All source table rows** option to publish all the rows.

In addition, you can indicate if deletes will be published by Q Capture. By selecting this option, you can build a logging target table, for example.

- Click the **6. Review Publications** button on the left. The Review and complete publications pane (Figure 8-24) is presented before you apply the changes.

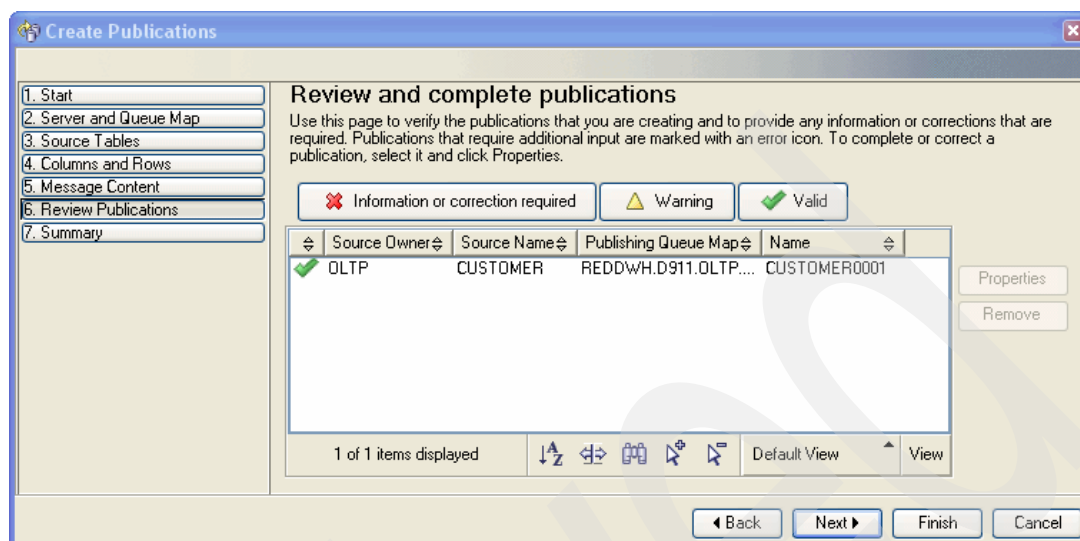


Figure 8-24 Event publishing - Review and complete publications panel

**Steps 4 and 5:** In this scenario, steps 4 and 5 of the Create Publications launchpad do not apply to our implementation, yet they are worthy of an explanation.

Step 4, “Columns and Rows” applies when you select a single table in step 3, “Source Tables.” In this case, you can identify the specific columns, key columns, and rows to be published in the Columns and Rows pane. However, if you select more than one table in step 3, “Source Tables,” this pane is not displayed. During this implementation, we publish all the columns of the source table and therefore do not see this pane.

By using step 5, “Message Content,” you can control the message content. You must specify whether Q Capture is to send messages when a selected source table contains changed columns or when any table has a column that changes. You can also indicate whether messages will contain only changed columns or unchanged columns. In addition, you can specify whether messages will include old and new values. Because this option is not relevant for this set up, we did not apply it.

## Event publishing message formats

The messages in event publishing can be published in XML or delimited format. The captured changes are made available in the target queue for an application. In our scenario, we used DataStage to get the messages from the queue, apply transformation to the data, and store the new information in our data warehouse. We used the delimited format, but XML is also supported by DataStage.

Because you have to configure DataStage, or create your own application, knowing the internal structure of a message can be of great help for debugging. Table 8-1 on page 190 shows the message header structure of a delimited message.

Table 8-1 DEL message format - Message header

Header element name	Description
type	An integer value that is used internally by IBM.
identifier	An identifier for the sending program. This value is used internally.
date	The date that the delimited message was put on the send queue.
time	The time that the delimited message was put on the send queue.
table_owner	The source table owner.
table_name	The source table name.
operation	<b>ISRT</b> An INSERT operation. For inserts, the before values of each column are null and are represented by column delimiters with no values between them. <b>REPL</b> An UPDATE or replace operation. Updates include the <i>before</i> values for each column followed by the <i>after</i> values. <b>DELE</b> A DELETE operation. For deletes, the after values of each column are null and are represented by column delimiters with no values between them.
transaction_identifier	A character string that uniquely identifies a DB2 unit of work
commit_lsn	A time-based log sequence number of the COMMIT statement for the transaction.
commit_time	The time stamp of the COMMIT statement for the transaction.
plan_name	The DB2 application plan name.
segment_number	Four numeric characters that are used to link segmented transaction messages.

Table 8-2 describes the format of the data part of the message.

Table 8-2 DEL message format - Data from the source table

Data element name	Description
column_value	The data value from the source table. A null column value is represented by two consecutive column delimiters.
column_delimiter	A single character that is used to separate column values. The default is a comma.
record_delimiter	A single character that is used to denote the end of a change-data record. The default is a new-line character

Example 8-12 shows the schematic representation of how the data is distributed inside the data part of the message. It is a succession of values and column delimiters until the end of the record, where there is a record delimiter.

Example 8-12 Message format

column_value	column_delimiter	column_value	column_delimiter	record_delimiter
--------------	------------------	--------------	------------------	------------------

You may need to change the delimiters that are provided by default. The update of publications and queue maps are done using the DB2 Replication Center. Figure 8-25 shows how to select the queue map that will have its message properties changed.

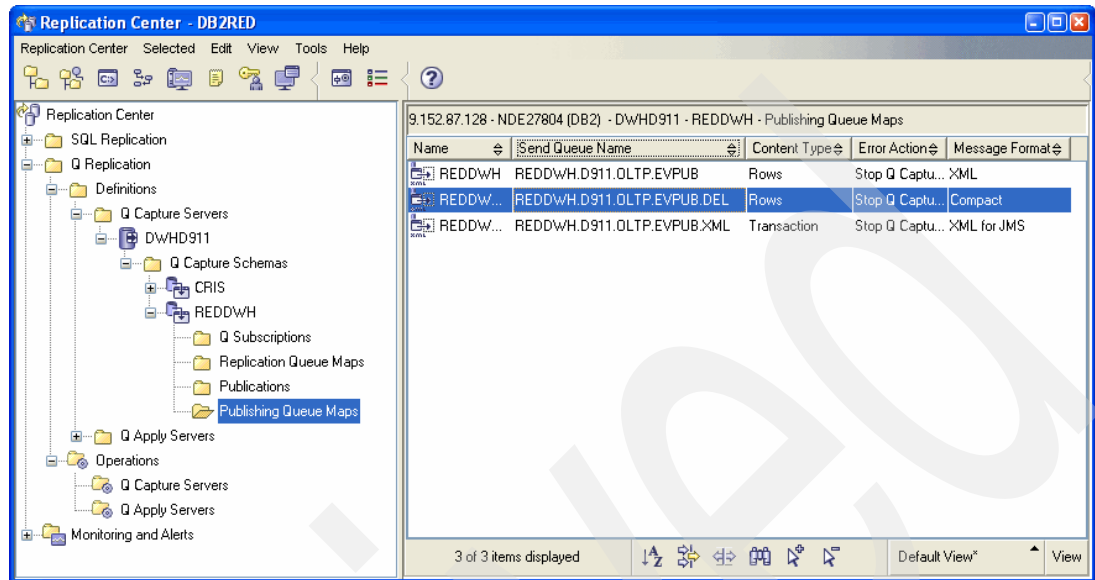


Figure 8-25 Updating Publishing Queue Maps

In the Publishing Queue Map Properties window (Figure 8-26), you can change the delimited format message properties.

**Publishing Queue Map Properties - REDDWH\_DEL**

9.152.87.128 - NDE27804 (DB2) - D\WHD911 - REDDWH - REDDWH\_DEL

Use this window to modify the attributes of the selected publishing queue map.

**General Properties**

Send queue: REDDWH.D911.OLTP.EVPUB.DEL

Type of message content: Row operation

Maximum message length: 64 KB

Options for handling queue errors: Stop Publications

☐ Allow the Q Capture program to send heartbeat messages

Interval between heartbeat messages: 0 seconds

**Message Format**

☒ XML format

☐ Add a JMS-compliant (MQRFH2) message header to all messages that are put on this send queue (for JMS applications and message brokers).

☐ Keep Q Capture running after code page conversion errors. Publish the data in error in binary format.

☒ Delimited format

Code page for publishing delimited messages: D

Column delimiter: ,

Character string delimiter: " (left quotation mark)

New line value: % (percent sign)

Decimal point value: .

Default

OK Cancel Help

Figure 8-26 Changing queue map properties

For the purpose of this book, we changed the publishing queue map parameters as follows:

- ▶ Type of message content: Row Operation
- ▶ Delimited format: <blank>
- ▶ Column delimiter: , (comma)
- ▶ Character string delimiter: " (left quotation mark)
- ▶ New line value: % (percent sign)

## Browsing event publishing messages

During our implementation, we found in multiple occasions the need for browsing the messages that event publishing was generating for debugging purposes. To browse these messages in the z/OS system, we need a tool that can access these messages from MQSeries in a non-destructive way and convert the information from Unicode to a readable ISPF format.



Example 8-13 shows a REXX™ program that is used to browse our event publishing queues.

*Example 8-13 Event publishing message browse REXX code sample*

---

```

/* REXX                                                                    */
/* NOTE: ERROR HANDLING CODE WAS REMOVED FOR CLARITY                      */
/*-----*/

ARG CSQ DBSSID Q1 /* RECEIVE QUEUE MANAGER AND QUEUE NAME AS PARAMETER */

RCC= RXMQV('INIT') /* INITIALISE THE MQ SERIES INTERFACE */
RCC = RXMQV('CONN', CSQ) /* CONNECT TO QUEUE MANAGER */

/* OPEN QUEUE Q1 FOR OUTPUT AND BROWSE ACCESS */
OO = MQOO_INQUIRE+MQOO_OUTPUT+MQOO_BROWSE+MQOO_SET
RCC = RXMQV('OPEN', Q1, OO , 'H2', 'OOD.' )

/* BROWSE ALL MSGS ON QUEUE Q1 */
DO I=1
  G.0      = 100000
  G.1      = ''
  IGMO.OPT = MQGMO_WAIT+MQGMO_BROWSE_NEXT
  RCC = RXMQV('GET', H2,'G.', 'IGMD.', 'OGMD.', 'IGMO.', 'OGMO.')

  IF ( WORD(RCC,1) <> 0 ) THEN LEAVE
  SAY COPIES('-',85)
  SAY 'MESSAGE NUMBER' I
  SAY 'MESSAGE LENGTH' G.0

  DO CONV = 1 TO (ABS(G.0/80)+1) /* UNICODE CONVERSION */
    START = (CONV * 80)-79
    CAST_LINE = SUBSTR(G.1,START,80)
    ADDRESS TSO 'SUBCOM DSNREXX'
    IF RC <> 0 THEN SUBCC = RXSUBCOM('ADD','DSNREXX','DSNREXX')

    ADDRESS DSNREXX 'CONNECT ' DBSSID
    ADDRESS DSNREXX 'EXECSQL DECLARE C1 CURSOR FOR S1'

    SQLSTMT = "SELECT CAST(XXXXX AS CHAR(80) CCSID 1208) AS CONVT ",
              "FROM ( ",
              "      SELECT CAST(''||CAST_LINE||'",
              "      AS VARCHAR(3800) CCSID UNICODE FOR BIT DATA" ,
              "      ) AS XXXXX" ,
              "      FROM SYSIBM.SYSDUMMY1) AS DD"

    ADDRESS DSNREXX 'EXECSQL PREPARE S1 FROM :SQLSTMT'
    ADDRESS DSNREXX 'EXECSQL OPEN C1'
    ADDRESS DSNREXX 'EXECSQL FETCH C1 INTO :CONVT'
    ADDRESS DSNREXX 'EXECSQL CLOSE C1'

    SAY LEFT(CONV,2,' ') ":" CONVT

  END /* DO CONV = 1 TO (ABS(MLEN.0/80)+1) */
END /* DO I=1 */
SAY COPIES('-',85)

RCC = RXMQV('BACK', )
RCC = RXMQV('TERM', )

```

---

This REXX program gets the messages from the queue passed as a parameter as shown in the JCL sample in Example 8-14. It uses a DB2 CAST function to convert the message text to a readable format and print the messages in 80 columns lines. You must agree to IBM terms before downloading and using the MQSeries SupportPac™ used for this program. For more details about the SupportPac, refer to the *MA18: A Rexx Interface to IBM MQSeries for MVS/ESA™ Version 2.0* document:

<ftp://ftp.software.ibm.com/software/integration/support/supportpacs/individual/ma18.pdf>

Example 8-14 shows the output of this program for a delimited format message.

*Example 8-14 JCL sample for event publishing message browse*

---

```
//CRISMQUN JOB (), 'BROWSE QCAPT MESS', REGION=OK, NOTIFY=CRIS,
//          MSGCLASS=X, CLASS=C
// *-----
//RUN      EXEC PGM=IKJEFT01
//SYSPROC DD DISP=SHR, DSN=CRIS.UTIL.CNTL
//STEPLIB DD DSN=CRIS.UTIL.MQREXX, DISP=SHR      <-- SUPPORTPACK MA18
//          DD DSN=SYS1.MQM.Q801.CSQLOAD, DISP=SHR <-- MQ SERIES
//          DD DSN=SYS1.MQM.V600.SCSQANLE, DISP=SHR <-- MQ SERIES
//          DD DSN=SYS1.MQM.V600.SCSQAUTH, DISP=SHR <-- MQ SERIES
//          DD DSN=SYS1.DSN.V910.SDSNLOAD, DISP=SHR <-- DB2
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
// * REXX NAME
// *      |      QUEUE MANAGER
// *      |      |      DB2 SUBSYSTEM
// *      |      |      |      QUEUE NAME
// *      V      V      V      V
// * REXXUNI1 Q801 D911 REDDWH.D911.OLTP.EVPUB
//SYSTSIN DD *
REXXUNI1 Q801 D911 REDDWH.D911.OLTP.EVPUB
// *
```

---

Example 8-15 shows the output of this program for a delimited format message.

*Example 8-15 Browsing a delimited event publishing message*

---

```
MESSAGE NUMBER 8
MESSAGE LENGTH 260
1  : 10, "IBM", "2008106", "021741852842", "OLTP", "CUSTOMER", "ISRT", "0000:0000:4815:566e:
2  : 0000", "0000:0000:4815:5923:0000", "2008-04-15-09.17.41", "ADB      ", "0000,,,,,,16
3  : 07, "KATRIN NOACK", "LUXEMBLAAN 2", "GERMANY      ", "555-55-5555      ", "-
4  : 1000.0, "DIRECTOR " @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
// *
```

---

Example 8-16 shows an XML example.

*Example 8-16 Browsing a XML event publishing message*

---

```
MESSAGE NUMBER 2
MESSAGE LENGTH 1534
1  :      @@@      <<psc><Command>Publish</Command><PubOpt>N
2  : one</PubOpt></psc>@@      <mcd><Msd>jms_text</Msd></mcd>@@      <jms></jms><?xml ve
3  : rsion="1.0" encoding="UTF-8" ?><msg xmlns:xsi="http://www.w3.org/2001/XMLSchema-
4  : instance" xsi:noNamespaceSchemaLocation="mqcap.xsd" version="1.0.0" dbName="DWH
5  : 911"><trans authID="CRIS      " correlationID="javaw.exe      " planName="DISTSERV" i
```

---

### 8.3 Operating Q replication and event publishing

### Example 8-17 Sample Q Capture output for event publishing

```
ASN0572I  "Q Capture" : "REDDWH" : "WorkerThread" : The "mqpub
9.1.0 (APAR pk60401, ASNRBASE APAR pk60401)" program initialized
successfully.
```

To solve this problem, the CUSTOMER table is altered as required, and the subscription must be activated again. Figure 8-27 shows how to activate the subscription by using the DB2 Replication Center.

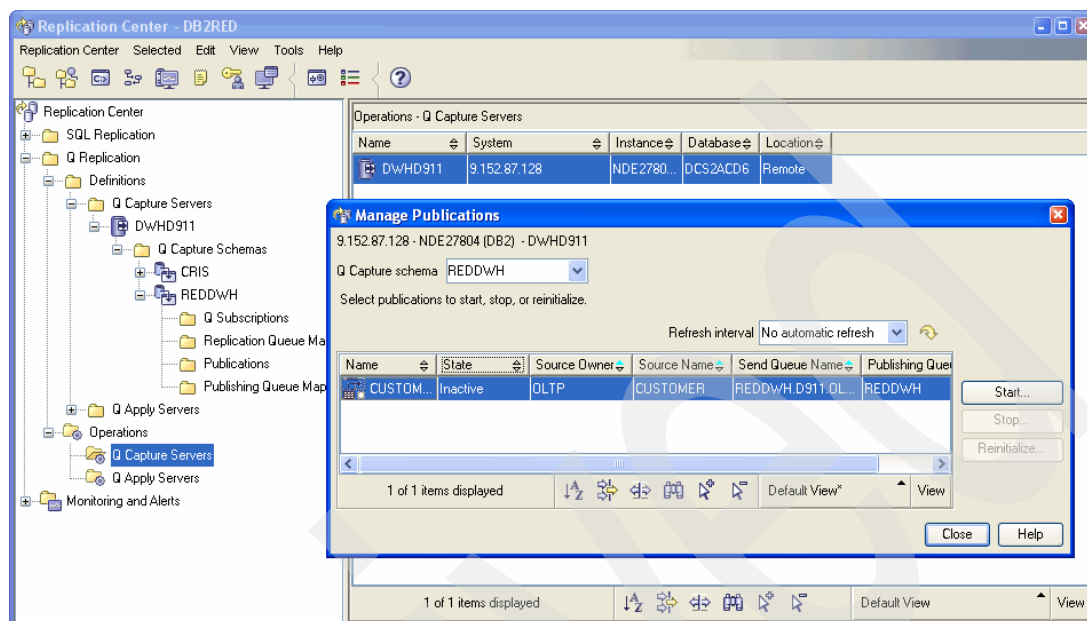


Figure 8-27 Starting an inactive event publishing subscription

After this change, the Q Capture started task activates the subscription and starts capturing changes on the source table. Example 8-18 shows the output in the Q Capture started task.

**Example 8-18 Q Capture output - A subscription being started**

```
ASN7010I "Q Capture" : "REDDWH" : "WorkerThread" : The program
successfully activated publication or Q subscription
"CUSTOMER0001" (send queue "REDDWH.D911.OLTP.EVPUB", publishing
or replication queue map "REDDWH") for source table
"OLTP.CUSTOMER".
```

Apply also uses the IBMQREP\_APPLYTRACE table for tracing information. This table contains informational, warning, and error messages.

The DESCRIPTION column, which is defined as VARCHAR(1024), contains the description of the information. The REXX code in Example 8-19 can be used to format this table's contents in an easier to read format.

**Example 8-19 REXX sample for formatting the contents of the IBMQREP\_APPLYTRACE table**

```
/* REXX                                                                    */
/* NOTE: ERROR HANDLING CODE WAS REMOVED FOR CLARITY                      */
/*-----*/

ARG DBSSID QSCHEMA

ADDRESS TSO 'SUBCOM DSNREXX'
IF RC <> 0 THEN SUBCC = RXSUBCOM('ADD','DSNREXX','DSNREXX')

ADDRESS DSNREXX 'CONNECT ' DBSSID
ADDRESS DSNREXX 'EXEC SQL DECLARE C1 CURSOR FOR S1'
```

```

SQLSTMT = "SELECT OPERATION,TRACE_TIME,DESCRIPTION, ",
          "COALESCE(REASON_CODE,0)",
          "COALESCE(MQ_CODE,0)",
          "FROM " || QSCHEMA || ".IBMQREP_APPLYTRACE"

ADDRESS DSNREXX 'EXECSQL PREPARE S1 FROM :SQLSTMT'
ADDRESS DSNREXX 'EXECSQL OPEN C1'
ADDRESS DSNREXX 'EXECSQL FETCH C1 INTO :OPER,:TIMS,:DESC,:REAS,:MQCD'
DO WHILE SQLCODE = 0
  SAY '+' || COPIES('-',82) || '+'
  SAY " | MESS TYPE: " LEFT(OPER,67," ") " |"
  SAY " | TIME:      " LEFT(TIMES,67," ") " |"
  SAY " | REASON CODE:" LEFT(REAS,67," ") " |"
  SAY " | MQ CODE:   " LEFT(MQCD,67," ") " |"
  SAY " | DESCRIPTION:" LEFT(' ',67," ") " |"
  DO I = 1 TO (ABS(LENGTH(DESC)/80)+1)
    START = (I * 80)-79
    DESC_LINE = SUBSTR(DESC,START,80)
    SAY " |" DESC_LINE " |"
  END /* DO I = 1 TO (ABS(DESC/80)+1) */
  ADDRESS DSNREXX 'EXECSQL FETCH C1 INTO :OPER,:TIMS,:DESC,:REAS,:MQCD'
END /* DO WHILE SQLCODE = 0 */
SAY '+' || COPIES('-',82) || '+'

ADDRESS DSNREXX 'EXECSQL CLOSE C1'
/* END */

```

#### JCL Sample for REXX code above:

```

//CRISMQUN JOB (),'BROWSE QCAPT MESS',REGION=OK,NOTIFY=CRIS,
//          MSGCLASS=X,CLASS=C
//*-----
//RUN      EXEC PGM=IKJEFT01
//SYSPROC DD DISP=SHR,DSN=CRIS.UTIL.CNTL
//STEPLIB DD DSN=SYS1.DSN.V910.SDSNLOAD,DISP=SHR <-- DB2
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
/* REXX NAME
/*
/*      |      DB2 SUBSYSTEM
/*      |      |      QUEUE APPLY SCHEMA
/*      V      V      V
/* REXXAPPL D912 REDDWH
//SYSTSIN DD *
REXXAPPL D912 REDDWH
/*

```

---

Example 8-20 shows an example of trace information being formatted by this program.

*Example 8-20 Output sample of formatted information in the IBMQREP\_APPLYTRACE table*

MESS TYPE:	INFO
TIME:	2008-04-11-02.47.31.931200
REASON CODE:	0
MQ CODE:	0
DESCRIPTION:	ASN7613I "Q Apply" : "REDDWH" : "BR00000" : Start processing queue (receive queue "REDDWH.D911.TO.D912.QREP", replication queue map "DWH911_TO_DWH912"), application single byte codepage "500", double byte codepage "500", source codepage "500", endian conversion required "0", float conversion "0".
MESS TYPE:	ERROR
TIME:	2008-04-11-02.47.32.093161
REASON CODE:	0
MQ CODE:	0
DESCRIPTION:	ASN0552E "Q Apply" : "REDDWH" : "BR00000SP005" : The program encountered an SQL error. The server name is "DWH911". The SQL request is "CONNECT". The table name is "N/A". The SQLCODE is "-950". The SQLSTATE is "42705". The SQLERRMC is "". The SQLERRP is "DSNLTSET".

Q Capture and Apply are USS applications that run under a user ID that has a UNIX profile defined in USS. In the home directory of this user ID, you can find useful information in a deep level of detail, which provides a great value for debugging and understanding how the programs are working.

We defined the user QREP for running Q Capture and Q Apply in our installation. To browse this information, assuming that you have the necessary privileges, you invoke OpenMvs by running the TSO OMVS command. Then you browse the home directory of the Q Capture started task user as shown in Example 8-21. In this example, the `cd /u/qrep` command fails because of a lack of authorization. After becoming root with the `su` command, we get into the qrep user directory. In this example, we list the unhidden files with the `ls -l` command.

The log file names are composed of the server name, the subscription schema, and QAPP for the apply of QCAP for capture, followed by the particle `.log`.

*Example 8-21 Getting information from USS*

```
DWH1:CRIS:/u/cris>cd /u/qrep
cd: /u/qrep: EDC5111I Permission denied.
DWH1:CRIS:/u/cris>su
:CRIS:/u/cris>cd /u/qrep
:CRIS:/u/qrep>
:CRIS:/u/qrep>ls -l
total 1000
-rw-r--r--  1 QREP    DE#03557   7803 Apr 10 15:42 D911.CRIS.QAPP.log
-rw-r--r--  1 QREP    DE#03557 506047 Apr 10 15:40 D911.CRIS.QCAP.log
-rw-r--r--  1 QREP    DE#03557  42431 Apr 11 12:13 D911.REDDWH.QCAP.log
-rw-r--r--  1 QREP    DE#03557  19195 Apr 11 12:13 D912.REDDWH.QAPP.log
:CRIS:/u/qrep>
===>
RUNNING
```

You can browse these files by using the **oedit** command as shown in Example 8-22.

*Example 8-22 Browsing Q Capture log using OEDIT*

---

```
oedit ./D911.REDDWH.QCAP.log
```

```
EDIT      /u/qrep/./D911.REDDWH.QCAP.log
```

```
Command ==>
```

```
000213 2008-04-11-03.13.14.905434
000214 <masterSub::selectKeyColumns>
000215 ASN7023W "Q Capture" : "REDDWH" : "WorkerThread" :
000216 Source table "OLTP_B.LINEITEM" does not have a primary key, unique constraint, or unique index.
000217 2008-04-11-03.13.14.908136
000218 <subMgr::handleCAPSTART>
000219 ASN7017I "Q Capture" : "REDDWH" : "WorkerThread" :
000220 The target table "DWHODS.LINEITEM_QREP" is ready to be loaded from source table
000221 "OLTP_B.LINEITEM" for publication or Q subscription "LINEITEM0001".
000222 2008-04-11-03.13.19.910982
000223 <subMgr::handleSignal>
000224 ASN7019I "Q Capture" : "REDDWH" : "WorkerThread" :
000225 "CAPSTART" signal was received and will be processed.
000226 2008-04-11-03.13.19.918373
000227 <subMgr::handleCAPSTART>
000228 ASN7017I "Q Capture" : "REDDWH" : "WorkerThread" :
000229 The target table "DWHODS.ORDER_QREP" is ready to be loaded from source table
000230 "OLTP_B.ORDER" for publication or Q subscription "ORDER0001".
```

---

Archived



## Setting up ETL components for a data warehouse

We use IBM Information Server on Linux on System z to both model and run the extract, transform, and load (ETL) jobs for accessing data from and loading it to our DB2 for z/OS subsystems. We also use WebSphere Classic Federation for DB2 for z/OS data and flat files.

In the following sections, we explain how to set up the required components both on z/OS and Linux on System z. This chapter includes the following sections:

- ▶ 9.1, “Overview of components for ETL on Linux on System z and z/OS” on page 202
- ▶ 9.2, “Configuring a HiperSocket connection to z/OS on Linux on System z” on page 203
- ▶ 9.3, “Setting up BatchPipes” on page 206
- ▶ 9.4, “Setting up WebSphere Classic Federation” on page 210
- ▶ 9.5, “Installing IBM Information Server” on page 228

## 9.1 Overview of components for ETL on Linux on System z and z/OS

Figure 9-1 shows an overview of the components that we install for both the initial and incremental updates of our data warehouse. We assume that DB2 for z/OS is set up and running and that the z/OS components are System Modification Program/Extended (SMP/E) installed.

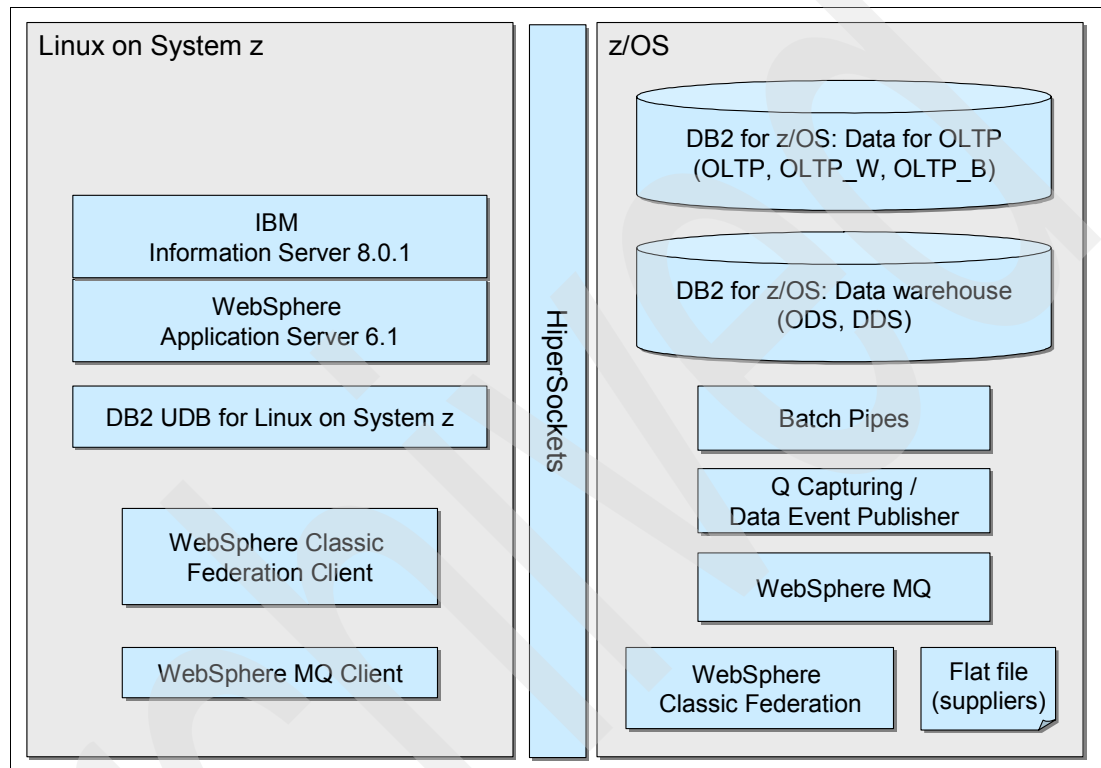


Figure 9-1 Components setup for ETL

We run a logical partition (LPAR) with Linux on System z and another LPAR with z/OS with the following configuration:

- ▶ For connecting components between the LPARs, we configure and use HiperSockets.
- ▶ On z/OS, we configure the BatchPipes subsystem for loading data in DB2 for z/OS.
- ▶ To access the flat file that is storing our supplier data, we install WebSphere Classic Federation (server components on z/OS and client components on Linux on System z).
- ▶ For incremental updates, we use Data Event Publisher on z/OS and WebSphere MQ (server components on z/OS and the client on Linux on System z).
- ▶ Information Server for Linux on System z with the DataStage component is installed and configured on Linux on System z. It comes with WebSphere Application Server and DB2 for Linux on System z for metadata.

## 9.2 Configuring a HiperSocket connection to z/OS on Linux on System z

If you start with a fresh installation of Linux on System z and run z/OS on the same physical machine, you can configure a HiperSocket connection from your LPAR running Linux on System z to z/OS. By doing this, you can access the remote DB2 for z/OS system using TCP/IP but avoid any additional physical hardware and software network stack overhead.

As a prerequisite, our z/OS system already has channels defined for HiperSockets. These definitions are in the Input Output Configuration Data Set (IOCDs) for z/OS. In our case, they contain four channels (E0, E1, E2 and E3) for the four available frame sizes of 16 KB, 24 KB, 40 KB, and 64 KB respectively (Example 9-1).

*Example 9-1 IOCDs definition for HiperSocket channels*

---

```
* HIPERSOCKET CHANNEL MAX. FRAMESIZE 16 KB
  CHPID PATH=(CSS(0,1),E0),SHARED,TYPE=IQD
  CNTLUNIT CUNUMBR=8000,PATH=((CSS(0),E0),(CSS(1),E0)),UNIT=IQD
  IODEVICE ADDRESS=(8000,032),CUNUMBR=(8000),UNIT=IQD
*
* HIPERSOCKET CHANNEL MAX. FRAMESIZE 24 KB
*
  CHPID PATH=(CSS(0,1),E1),SHARED,CHPARM=40,TYPE=IQD
  CNTLUNIT CUNUMBR=8100,PATH=((CSS(0),E1),(CSS(1),E1)),UNIT=IQD
  IODEVICE ADDRESS=(8100,032),CUNUMBR=(8100),UNIT=IQD
*
* HIPERSOCKET CHANNEL MAX. FRAMESIZE 40 KB
*
  CHPID PATH=(CSS(0,1),E2),SHARED,CHPARM=80,TYPE=IQD
  CNTLUNIT CUNUMBR=8200,PATH=((CSS(0),E2),(CSS(1),E2)),UNIT=IQD
  IODEVICE ADDRESS=(8200,032),CUNUMBR=(8200),UNIT=IQD
*
* HIPERSOCKET CHANNEL MAX. FRAMESIZE 64 KB
*
  CHPID PATH=(CSS(0,1),E3),SHARED,CHPARM=C0,TYPE=IQD
  CNTLUNIT CUNUMBR=8300,PATH=((CSS(0),E3),(CSS(1),E3)),UNIT=IQD
  IODEVICE ADDRESS=(8300,032),CUNUMBR=(8300),UNIT=IQD
```

---

To make a HiperSocket connection available on Linux on System z:

**Notes:**

- Configuration utilities, such as YaST, are not available on all Linux distributions.
- We use SUSE Linux Enterprise Server 10 (SLES 10, Linux version 2.6.16.53-0.18-default).

1. By using the YaST configuration utility on Linux on System z, in the YaST Control Center window (Figure 9-2), under Network Devices, select **Network Card**.

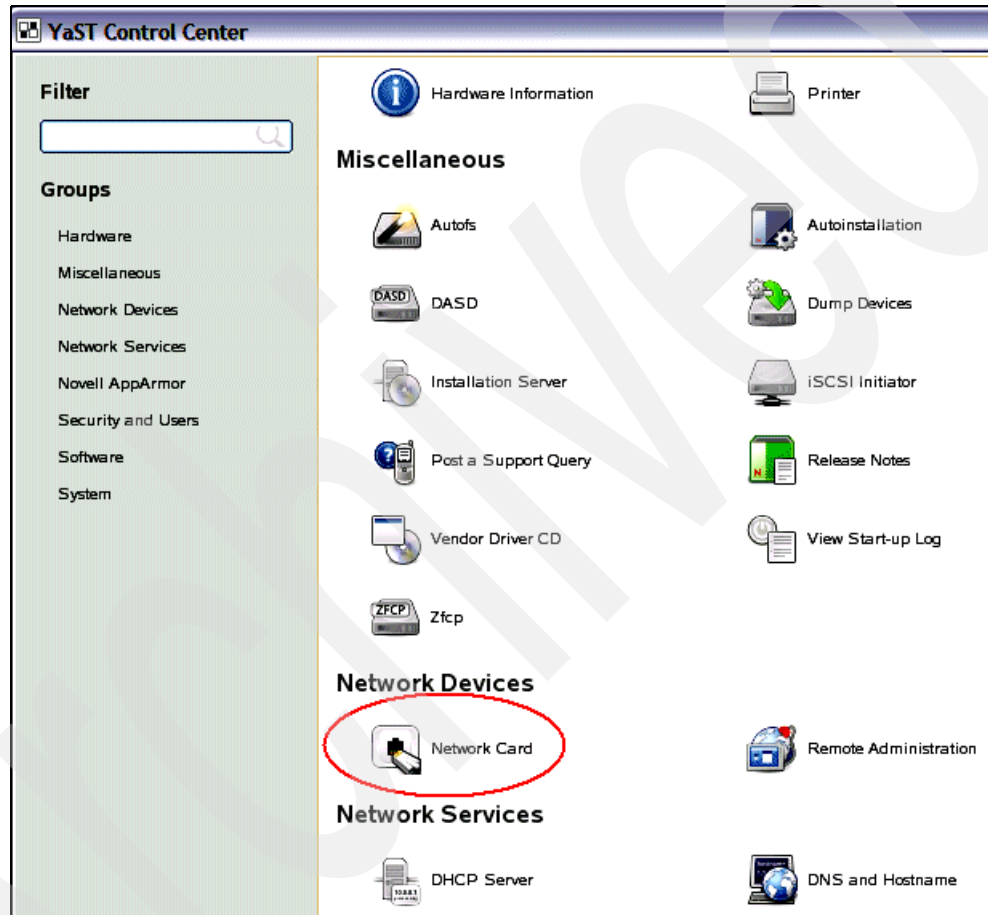
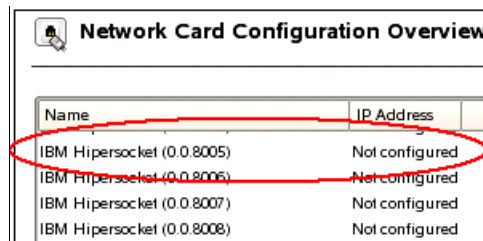


Figure 9-2 YaST Control Center

2. For the Network Setup Method option, select **Traditional Method with ifup** and click **Next**.

3. In the Network Card Configuration Overview (Figure 9-3), which shows the available and not yet configured HiperSockets, choose a HiperSocket and click **Edit**.

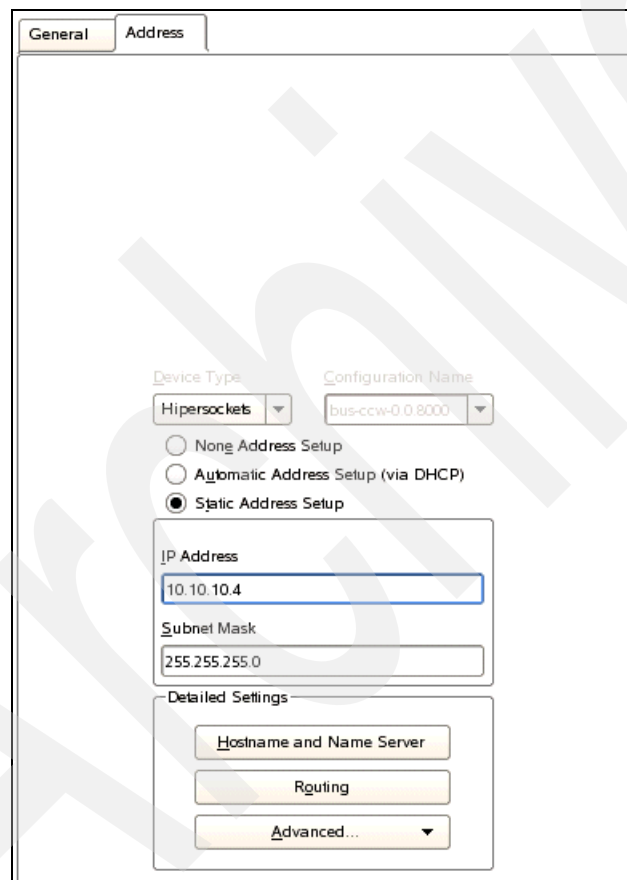


The screenshot shows a window titled "Network Card Configuration Overview". It contains a table with two columns: "Name" and "IP Address". There are four rows, each representing an IBM HiperSocket. The first row is circled in red. The "IP Address" column for all rows contains the text "Not configured".

Name	IP Address
IBM Hipersocket (0.0.8005)	Not configured
IBM Hipersocket (0.0.8006)	Not configured
IBM Hipersocket (0.0.8007)	Not configured
IBM Hipersocket (0.0.8008)	Not configured

Figure 9-3 Network Card Configuration Overview

4. Assign the new connection a static IP address. In this example, we type 10.10.10.4 as shown in Figure 9-4. Assign the HiperSocket device name. On this same page, you see the channels as defined on z/OS.



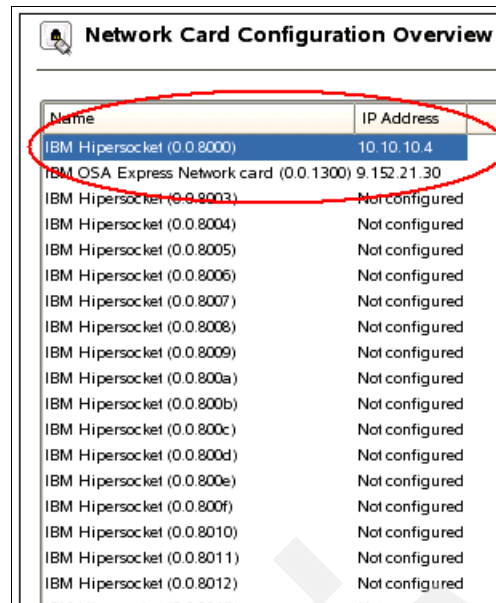
The screenshot shows a configuration window with two tabs: "General" and "Address". The "Address" tab is selected. It contains the following fields and options:

- Device Type:** A dropdown menu set to "Hipersockets".
- Configuration Name:** A dropdown menu set to "bus-ccw-0.0.8000".
- Address Setup Options:** Three radio buttons: "None Address Setup", "Automatic Address Setup (via DHCP)", and "Static Address Setup". The "Static Address Setup" option is selected.
- IP Address:** A text field containing "10.10.10.4".
- Subnet Mask:** A text field containing "255.255.255.0".
- Detailed Settings:** A section containing three buttons: "Hostname and Name Server", "Routing", and "Advanced..." (with a dropdown arrow).

Figure 9-4 Assigning the static IP address

5. Accept the default of the remaining settings and click **Finish**.

The configured HiperSocket connection is now displayed in the Network Card Configuration Overview (Figure 9-5).



Name	IP Address
IBM Hipersocket (0.0.8000)	10.10.10.4
IBM OSA Express Network card (0.0.1300)	9.152.21.30
IBM Hipersocket (0.0.8003)	Not configured
IBM Hipersocket (0.0.8004)	Not configured
IBM Hipersocket (0.0.8005)	Not configured
IBM Hipersocket (0.0.8006)	Not configured
IBM Hipersocket (0.0.8007)	Not configured
IBM Hipersocket (0.0.8008)	Not configured
IBM Hipersocket (0.0.8009)	Not configured
IBM Hipersocket (0.0.800a)	Not configured
IBM Hipersocket (0.0.800b)	Not configured
IBM Hipersocket (0.0.800c)	Not configured
IBM Hipersocket (0.0.800d)	Not configured
IBM Hipersocket (0.0.800e)	Not configured
IBM Hipersocket (0.0.800f)	Not configured
IBM Hipersocket (0.0.8010)	Not configured
IBM Hipersocket (0.0.8011)	Not configured
IBM Hipersocket (0.0.8012)	Not configured

Figure 9-5 HiperSocket connection displayed

6. Apply the new setting to the system.
7. Type the **ping** command to check whether the network is reachable and verify that the configuration is working as expected. See Example 9-2.

*Example 9-2 Using the ping command to verify the HiperSocket connection*

```
# ping 10.10.10.2
PING 10.10.10.2 (10.10.10.2) 56(84) bytes of data.
64 bytes from 10.10.10.2: icmp_seq=1 ttl=64 time=0.131 ms

#ping 10.10.10.4
PING 10.10.10.4 (10.10.10.4) 56(84) bytes of data.
64 bytes from 10.10.10.4: icmp_seq=1 ttl=64 time=0.029 ms
```

## 9.3 Setting up BatchPipes

In our scenario, DataStage integration jobs extract and transform data based on defined data sources. The jobs eventually write the result to a data set in the file system or insert records into a database in DB2 for z/OS by using the load utility.

Traditionally, the LOAD utility reads from a data set and cannot start processing before writing to the data set is completed. Particularly, there is no way that the ETL job can write while the LOAD job processes records.

BatchPipes offer a way for both “writer” and “reader” jobs to run concurrently and thereby provide a new approach to efficiently load data from remote servers into DB2 for z/OS. The recent enhancements facilitate and allow the use of BatchPipes in our context.

## Feeding z/OS BatchPipes through z/OS FTP

With the authorized program analysis report (APARs) mentioned in Table 9-1 on page 209, FTP is extended to support the feeding of data to BatchPipes. The log output in Example 9-3 shows an FTP command that is triggered by a DataStage job on Linux on System z. It feeds data into BatchPipes for subsystem BP0 by specifying the site command `site subsys=BP01` for FTP.

*Example 9-3 Log output for the FTP command issued by a DataStage DB2z stage*

---

```
DB2Z_1: [0] pftp
[1] -schema
[2] record
( X: nullable int32 {position=0, binary, big_endian, null_field=""};
  S: nullable string[max=200] {position=4, padchar="@", ebcdic, prefix=2, null_field=""};
  _nullFieldVector_[2]: uint8 {delim=none, position=206, binary};
)
[3] -uri
[4] FTP://10.10.10.2/'FNEUMAN.IN00000'
[5] -open_command
[6] quote site LRECL=209;quote site RECFM=FB;quote site subsys=BP01;debug 1
[7] -user
[8] FNEUMAN
[9] -password
[10] xxxxx
[11] -mode
[12] put
[13] -overwrite
[14] -transfer_type
[15] binary
[16] -norepartition
```

---

## Using BatchPipes in the DB2 LOAD utility

With the APARs mentioned in Table 9-1 on page 209, the LOAD utility can load data from a BatchPipes data set by using the following keywords:

- ▶ **SUBSYS** name, which specifies the BatchPipes subsystem name  
When SUBSYS is specified, LRECL and RECFM are required.
- ▶ **LRECL**, which specifies the record length of the BatchPipes subsystem  
There is no default value, and this option is required when SUBSYS is specified.
- ▶ **RECFM**, which specifies the record format of the BatchPipes subsystem file  
Valid values are F, FB, V, or VB. There is no default value, and this option is required when SUBSYS is specified.

The log output in Example 9-4 shows the LOAD utility call that is triggered by a DataStage integration job through the DSNUTILS stored procedure.

*Example 9-4 Log output for a load utility call through DSNUTILS*

---

```
TEMPLATE TMPL_IN DSN 'FNEUMAN.IN00000'
SUBSYS(BP01) RECFM(FB) LRECL(209)
TEMPLATE TMPL_WK1 DSN 'FNEUMAN.WORK1' UNIT SYSDA
DISP(NEW,DELETE,DELETE)
SPACE(16,1) CYL
TEMPLATE TMPL_WK2 DSN 'FNEUMAN.WORK2' UNIT SYSDA
DISP(NEW,DELETE,DELETE)
SPACE(16,1) CYL
TEMPLATE TMPL_ERR DSN 'FNEUMAN.SYSERR' UNIT SYSDA
```

```

DISP(MOD,CATLG,DELETE)
SPACE(16,1) CYL
LOAD DATA INDDN TMPL_IN RESUME NO REPLACE
LOG NO NOCOPYPEND EBCDIC
SORTKEYS 0 SORTDEVT SYSDA
WORKDDN(TMPL_WK1,TMPL_WK2) ERRDDN TMPL_ERR
INTO TABLE FNEUMAN.FNT11 (
  "X" POSITION(1:4) INTEGER NULLIF(207)=X'FF',
  "S" POSITION(5:206) VARCHAR NULLIF(208)=X'FF'
);

```

## Interaction between DataStage and DB2 for z/OS

Figure 9-6 depicts the interaction between the components. A DataStage job running on Linux on System z creates an output that is sent through FTP and the BatchPipes extension to the BatchPipes subsystem on z/OS. While transferring data to the batch pipe, DataStage invokes the DSNUTILS administration stored procedure to trigger the LOAD utility. LOAD reads from the batch pipe and loads the data into the desired DB2 for z/OS table.

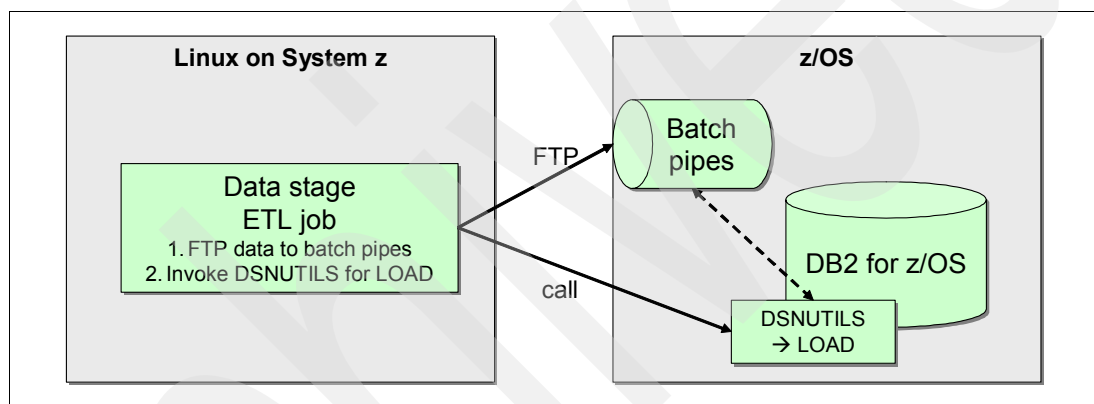


Figure 9-6 Interacting with batch pipes

## Troubleshooting BatchPipes

With /BP01 STATUS, you can inspect the status of the BatchPipes subsystem BP01 (Example 9-5) and check if writer or reader jobs are currently running.

### Example 9-5 BatchPipes subsystem status

```

RESPONSE=DWH1
ASFP210I 16:22:23 BP01 STATUS 639
#JOBS=4      #PIPES=3      #ALLOC=4      STATUS=ACTIVE  MODE=LOCAL
#WAITALLOC=0      MAX WAITALLOC TIME=00:00:00
#WRITERS=0
#WAITOPEN=3      MAX WAITOPEN TIME=00:59:01
#IDLE=0      MAX IDLE TIME=00:00:00
#WAIT=0      MAX WAIT TIME=00:00:00
#WAITCLOSE=0      MAX WAITCLOSE TIME=00:00:00
#WAITTERM=0      MAX WAITTERM TIME=00:00:00
#READERS=0
#WAITOPEN=1      MAX WAITOPEN TIME=00:07:34
#IDLE=0      MAX IDLE TIME=00:00:00
#WAIT=0      MAX WAIT TIME=00:00:00
#WAITEOF=0      MAX WAITEOF TIME=00:00:00
#WAITCLOSE=0      MAX WAITCLOSE TIME=00:00:00
#WAITTERM=0      MAX WAITTERM TIME=00:00:00

```



To get a list of currently opened batch pipes, you can issue the following command:

```
/BP01 ST,p=*
```

Example 9-6 shows output with three open batch pipes, named FNEUMAN.IN00000, SREENI.IN00000 and SREENI.PART.IN00000). The default name that Information Server chooses for the batch pipes are in the form *dsn*.IN00000.

*Example 9-6 Listing of currently opened batch pipes in the system*

---

```

RESPONSE=DWH1
ASFP210I 16:23:33 BP01 STATUS 644
PIPE=FNEUMAN.IN00000                                DSPNM=ASFPDQ01
  JOB=DSNUTILS SYS=DWH1      STEP=IEFPROC  NUM=STC08157
    READ      WAITOPEN=00:08:44  COUNT=0      WAITS=0
      REMAINING OPENS=(W=1 ,R=0 )
PIPE=SREENI.IN00000                                DSPNM=ASFPDQ01
  JOB=SREENI   SYS=DWH1      STEP=STEP1    NUM=
    WRITE     WAITOPEN=01:00:11  COUNT=0      WAITS=0
      REMAINING OPENS=(W=0 ,R=1 )
  JOB=SREENI   SYS=DWH1      STEP=STEP1    NUM=
    WRITE     WAITOPEN=00:58:32  COUNT=0      WAITS=0
      REMAINING OPENS=(W=0 ,R=1 )
PIPE=SREENI.PART.IN00000                          DSPNM=ASFPDQ01
  JOB=SREENI   SYS=DWH1      STEP=STEP1    NUM=
    WRITE     WAITOPEN=00:50:12  COUNT=0      WAITS=0
      REMAINING OPENS=(W=0 ,R=1 )

```

---

You can obtain additional information for specific batch pipes by using a commands such as the following example:

```
/BP01 ST,p=FNEUMAN.IN00000,FLOW
```

**Important:** We encountered situations where the DataStage job failed and we had to manually cancel either the job writing into BatchPipes (FTP) or the job reading from BatchPipes (LOAD). If you are using a BatchPipes file, you cannot restart the LOAD utility. If the application that populates the BatchPipes file terminates, you must terminate the job where the LOAD utility is executing. If the utility was invoked from a stored procedure, you must also terminate the Workload Manager (WLM) application environment of the LOAD utility that reads the BatchPipes file. After you terminate the job, terminate the LOAD utility by using the DB2 TERM UTILITY command. Then you can resubmit the LOAD job.

## List of required APARs for BatchPipes support

Table 9-1 shows the required maintenance for this function.

*Table 9-1 Required APARs for BatchPipes support*

APAR	Description
PK37032 for z/OS	Introduces BatchPipes support for FTP in z/OS as of 1.8
PK54242 for z/OS	FTP server using BatchPipes is not setting the proper LRECL
PK34251 (PTF UK26290) for DB2 z/OS V8	BatchPipes support for DB2 LOAD
PK34251 (PTF UK25291) for DB2 9 for z/OS	BatchPipes support for DB2 LOAD

If PK54242 is not installed on your system, you receive an error message in the log such as the one listed in Example 9-7.

*Example 9-7 Error message in the log if PK54242 is missing*

---

```
$HASP373 DSNUTILS STARTED
ICH70001I FNEUMAN LAST ACCESS AT 16:19:43 ON THURSDAY, APRIL 24, 2008
ASFP328I BATCHPIPES FAILURE. LRECL IS NOT CONSISTENT WITH PIPE. 630
          JOB=FNEUMAN STEP=STEP1 DD=SYS00001 SUBSYS=BP01
IEC150I 913-74,IGG0199G,FNEUMAN,STEP1,SYS00001
```

---

## 9.4 Setting up WebSphere Classic Federation

The WebSphere Classic Federation Server for z/OS architecture consists of following components:

- ▶ Data server
- ▶ Data connectors
- ▶ Classic Data Architect
- ▶ Metadata catalog
- ▶ Clients (Open Database Connectivity (ODBC), Java Database Connectivity (JDBC), and command line interface (CLI))

### Data server

Data servers perform all data access. The data server consists of several components or services. A major service embedded in the data server is the query processor that acts as the relational engine for Classic Federation. These services are defined by service information entries in a configuration file.

Data servers perform the following functions:

- ▶ Accept SQL queries from clients
- ▶ Determine the type of data to access
- ▶ Transform SQL queries into the native file or database access language
- ▶ Optimize queries
- ▶ Create relational result sets from native database records
- ▶ Process post-query result sets as needed

A data server accepts connection requests from client applications. Client applications can access a data server by using the ODBC, JDBC, or CLI client that IBM WebSphere Classic Federation Server for z/OS provides. The following services run under the data server:

- ▶ Region controller

The region controller is the main service of a data server. It monitors and controls the other services that run within the data server. The region controller directly or indirectly activates each service based on the configuration parameters that you define in a master configuration member by using service information entry parameters.

- ▶ Connection handlers

A connection handler listens for connection requests from client applications and routes them to the appropriate query processor task. The connection handler task can load the following communication protocols:

- TCP/IP
- z/OS Cross Memory Services
- WebSphere MQ

- ▶ Query processors

The query processor is the subcomponent of the data server that processes SQL requests. The SQL requests can access a single database or file system or reference multiple types of databases or file systems. There are two types of query processors:

- *Single-phase commit query processor (CACQP)* accesses and joins information from multiple data sources and performs updates to a single data source.
- *Two-phase commit query processor (CACQPRRS)* accesses and joins information from multiple data sources and performs updates to multiple data sources. The two-phase commit query processor uses z/OS Resource Recovery Services to coordinate the data source updates. The two-phase commit query processor supports the CA-Datcom, DB2 for z/OS, IMS, and VSAM data sources.

- ▶ Initialization services

Initialization services are special tasks that initialize different types of interfaces to underlying database management systems and z/OS system components.

- ▶ System exits

System exits are subcomponents of the data server and the query processor. The system exits are designed to run in a multiuser environment and perform security, accounting, and workload management functions to support large numbers of concurrent users.

- ▶ Logger service

The logger service is a task for system monitoring and troubleshooting. A single logger task can run within a data server. The logger reports data server activities and is used for error diagnosis.

## Data connectors

WebSphere Classic Federation for z/OS supports data connectors, which provide access to the following data sources:

- ▶ Adabas

Provides access to Adabas files.

- ▶ CA-Datcom

Provides access to CA-Datcom files.

- ▶ CA-IDMS

Provides access to CA-IDMS files.

- ▶ DB2 for z/OS

Provides access to DB2 for z/OS tables.

- ▶ IMS

Provides access to IMS data using the IMS DRA interface or the IMS BMP/DBB interface.

- ▶ Sequential

Provides access to sequential files or members.

- ▶ VSAM

Provides access to native VSAM files, VSAM files under the control of CICS, and VSAM files under the control of DFSMSHfs.

The query processor dynamically loads one or more data connectors to access the target database or file system that is referenced in the SQL request.

## Classic Data Architect

The Classic Data Architect is the administrative tool that is used to map the data definitions to the logical tables. It helps to perform following tasks:

- ▶ Define tables, columns, primary keys, indexes, stored procedures, and views
- ▶ Specify user authorization for all objects
- ▶ Import existing physical definitions from copybooks, CA-IDMS schemas, and IMS database descriptors (DBDs)
- ▶ Generate Data Definition Language (DDL) for the objects that you create that can be run directly on a server or saved to a script file
- ▶ Generate DDL scripts from objects that are already defined in the catalog and export DDL scripts to a data set on the server for use with the metadata utility
- ▶ Connect directly to a classic data source and view the objects in the system catalog

## Metadata catalog

A metadata catalog is a set of relational tables that contain information about how to convert data from non-relational to relational formats. The data server accesses the information stored in these catalogs. The information generated from Classic Data Architect is stored in a metadata catalog.

## Clients (ODBC, JDBC, and CLI)

WebSphere Classic Federation Server for z/OS provides the ODBC, JDBC, and CLI clients. The clients enable client applications or tools to submit SQL queries to the data server. The clients use TCP/IP or WebSphere MQ communication protocols to establish a connection with a target data server.

**Note:** In this book, we do not discuss how to leverage WebSphere Classic Event Publisher to handle incremental updates to legacy data sources. We assume that the data in the referenced legacy data sources remains constant and that changes are just considered for processing during a full refresh of the data warehouse.

### 9.4.1 Setting up the WebSphere Classic Federation server on z/OS

After completing the SMP/E installation, you must complete the following tasks to configure WebSphere Classic Federation on z/OS:

- ▶ APF authorize the SCACLOAD and SCACSASC data sets
- ▶ Configure the data servers and services

#### Configuring the data server on z/OS

The data server configuration parameters that define services and other operational and tuning parameters are defined in configuration files that are stored as members in a configuration partitioned data set (PDS). To implement WebSphere Classic Federation, we use the following members stored in SCACCONF PDS to configure a data server:

- ▶ Data server configuration (CACDSCF)
- ▶ Query processor configuration (CACQPCF)

To configure the data server:

1. Copy the sample data server job control language (JCL) in a separate proclib or JCL library for editing per the required configuration. Customize the JCL for the environment and point to our configuration member 'WCF.CACDS01.SCACCONF(CACDSCF1)' on the VHSCONF DD statement. Refer to Example 9-8 for the data server JCL procedure details.

*Example 9-8 Data server procedure*

---

```
//CACDS   PROC CAC='SYS1.WSC.V910',      INSTALLED HIGH LEVEL QUALIFIER
//          DB2='SYS1.DSN.V910',        DB2 HIGH LEVEL QUALIFIER
//          CONFIG=CACDSCF1,           DATA SERVER CONFIG FILE
//          SOUT='*',                   SYSOUT CLASS
//          RGN=32M                     REGION SIZE
//CACPROC  EXEC PGM=CACNTL,TIME=1440,REGION=&RGN
//* PARM='&PARM1 '
//STEPLIB DD DISP=SHR,DSN=&CAC..SCACLOAD
//*       DD DISP=SHR,DSN=&ADA..LOAD
//       DD DISP=SHR,DSN=&DB2..SDSNLOAD
//VHSCONF DD DISP=SHR,DSN=WCF.CACDS01.SCACCONF(&CONFIG)
//CTRANS  DD DISP=SHR,DSN=&CAC..SCACSASC
//*
//* META DATA CATALOGS
//CACCAT  DD DISP=SHR,DSN=WCF.CACDS01.CATALOG
//CACINDX DD DISP=SHR,DSN=WCF.CACDS01.CATINDX
```

---

2. Copy CACDSCF and CACQPCF from the SCACCONF member to a separate configuration library, which is WCF.CACDS01.SCACCONF in our case.
3. Modify the CACDSCF member for the required configuration. We made the following modification in the CACDSCF member:
  - a. Define a query processor (REDDWHZ) with the following query processor service definition in the CACDSF member in our configuration library (copied as CACDSF1). Refer to Example 9-9 for details about the definition.

*Example 9-9 Query processor definition*

---

```
*****
*
* QUERY PROCESSOR SERVICE INFO ENTRY
*   THE LAST SUBPARAMETER POINTS TO A QP SERVICE CONFIGURATION FILE
SERVICE INFO ENTRY = CACQP REDDWHZ 2 5 10 20 4 5M 5M CACQPCF
*
*****
```

---

The Query processor definitions field shown in Example 9-9 has the following fields:

- Field 1** Type of query processor. CACQP describes it as single phase commit query processor.
- Field 2** Provides the name in the data server for the service. It is REDDWHZ in our case.
- Field 3** Service start class. Provides the priority to the service definition. Service class 2 is less than the data server's main controller task (service class = 0) and the logger task (service class=1).
- Field 4** Minimum of number of instances of the service.
- Field 5** Maximum number of instances of the service.
- Field 6** Maximum number of connections per task.

**Field 7** Tracing level as the service passes the logger service. Trace level 4 means informational and warning messages only.

**Field 8** Response time out.

**Field 9** Idle time out.

**Field 10** Service specific information. The name of the member of SCACCONF library.

- b. Configure the TCP/IP connection handler. We modified the TCP/IP connection handler service information entry in the CACDSF1. We defined port 5999 in the definition where the data server will listen to incoming requests. Refer to Example 9-10 for the exact definition.

---

*Example 9-10 TCP/IP connection handler configuration*

---

```
*
* TCP/IP CONNECTION HANDLER
* REFER TO DOCUMENTATION FOR DETAILED INFORMATION ON LAST SUBPARAMETER
SERVICE INFO ENTRY = CACINIT TCPIP 2 1 1 100 4 5M 5M \
TCP/0.0.0.0/5999
*
```

---

- c. Enable SAF Exit by using the statements in Example 9-11. SAF Exit is used to verify that a user has authority to access a physical file or PSB referenced in an SQL query.

---

*Example 9-11 SAF Exit*

---

```
* SAF (SECURITY) SYSTEM EXIT
SAF EXIT = CACSX04
```

---

4. Start the data server. The data server can be started by issuing the start command from the MVS log. The data server job log should look similar to the log in Example 9-12.

---

*Example 9-12 Data server job log*

---

```
CAC00105I LOG V9.1 08172007: STARTED
CAC00100I CONTROLLER: LOGGING STARTED
CAC00105I QUERY PROCESSOR V9.1 11142007: STARTED
CAC00102I CONTROLLER: STARTED CACQP
CAC00105I QUERY PROCESSOR V9.1 11142007: STARTED
CAC00102I CONTROLLER: STARTED CACQP
CAC00105I QUERY PROCESSOR V9.1 11142007: STARTED
CAC00102I CONTROLLER: STARTED CACQP
CAC00105I QUERY PROCESSOR V9.1 11142007: STARTED
CAC00102I CONTROLLER: STARTED CACQP
CAC00105I QUERY PROCESSOR V9.1 11142007: STARTED
CAC00102I CONTROLLER: STARTED CACQP
CAC00102I CONTROLLER: STARTED CACINIT
CAC00103I DATA SERVER: V9.1 08172007 READY
CAC00105I CONNECTION HANDLER V9.1 08172007: STARTED
```

---

## 9.4.2 Defining and registering the flat file in Classic Data Architect

To map the sequential files in the Classic Federation data server, we used the Classic Data Architect tool. We describe the configuration of Classic Data Architect for the data server in “Configuring the Classic Data Architect for the data server” on page 215.

### Configuring the Classic Data Architect for the data server

To map sequential files in Classic Federation, Classic Data Architect connects to the data server that is running on z/OS. Figure 9-7 shows the configuration parameters for our scenario.

**Edit Connection**

**Connection Parameters**  
Select the database manager, JDBC driver, and required connection parameters.

**Connection identification**  
☐ Use default naming convention  
Connection Name: REDDWHZ

**Select a database manager:**

- Classic Integration
  - V9
- Cloudscape
- DB2 UDB
- DB2 UDB iSeries
- DB2 UDB zSeries
- Derby
- Generic JDBC
- Informix
- MySQL
- Oracle
- SQL Server
- Sybase

JDBC driver: Classic Integration Server JDBC Driver

**Connection URL details**  
Data source: REDDWHZ  
Host: 9.152.87.128  
Port number: 5999  
Code page: Cp037  
JDBC driver class: com.ibm.cac.jdbc.Driver  
Class location: C:\Program Files\IBM\WSClassic91\CDA\ecd [Browse...](#)  
Connection URL: jdbc:cac:REDDWHZ:tcp/9.152.87.128/5999:codepage=Cp037

**User information**  
User ID: gauran  
Password: \*\*\*\*\*  
[Test Connection](#)

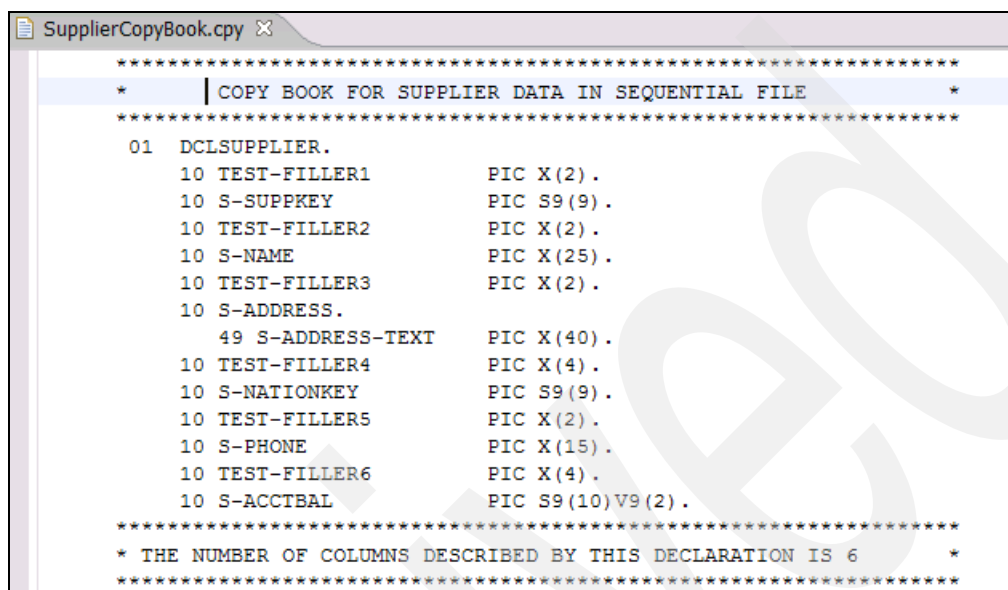
[?< Back](#) [Next >](#) [Finish](#) [Cancel](#)

Figure 9-7 Data server configuration in Classic Architect

The data source name in the connection details point to the query processor name configured in the CACDSF1 configuration file. The port number used here is also configured in the Service Info Entry for the CACINIT in CACDSF1 member.

## Defining copybook for the sequential file

A copybook must be defined in Data Architect. The copybook is used later for defining a sequential table in the data server. We defined the copybook shown in Figure 9-8 in consideration of the sequential file format described in “SUPPLIER data” on page 82.



The screenshot shows a window titled "SupplierCopyBook.cpy" with a close button. The content is a copybook definition for a sequential file. It starts with a header line: "COPY BOOK FOR SUPPLIER DATA IN SEQUENTIAL FILE". Below this, there is a declaration for a table named "DCLSUPPLIER". The declaration lists several fields with their data types and lengths: "TEST-FILLER1" (PIC X(2)), "S-SUPPKEY" (PIC S9(9)), "TEST-FILLER2" (PIC X(2)), "S-NAME" (PIC X(25)), "TEST-FILLER3" (PIC X(2)), "S-ADDRESS" (PIC X(40)), "TEST-FILLER4" (PIC X(4)), "S-NATIONKEY" (PIC S9(9)), "TEST-FILLER5" (PIC X(2)), "S-PHONE" (PIC X(15)), "TEST-FILLER6" (PIC X(4)), and "S-ACCTBAL" (PIC S9(10)V9(2)). The declaration is enclosed in a block of asterisks. At the bottom, there is a line indicating the number of columns: "THE NUMBER OF COLUMNS DESCRIBED BY THIS DECLARATION IS 6".

```
*****
* COPY BOOK FOR SUPPLIER DATA IN SEQUENTIAL FILE *
*****
01 DCLSUPPLIER.
   10 TEST-FILLER1          PIC X(2) .
   10 S-SUPPKEY             PIC S9(9) .
   10 TEST-FILLER2          PIC X(2) .
   10 S-NAME                PIC X(25) .
   10 TEST-FILLER3          PIC X(2) .
   10 S-ADDRESS.
      49 S-ADDRESS-TEXT     PIC X(40) .
   10 TEST-FILLER4          PIC X(4) .
   10 S-NATIONKEY           PIC S9(9) .
   10 TEST-FILLER5          PIC X(2) .
   10 S-PHONE               PIC X(15) .
   10 TEST-FILLER6          PIC X(4) .
   10 S-ACCTBAL             PIC S9(10)V9(2) .
*****
* THE NUMBER OF COLUMNS DESCRIBED BY THIS DECLARATION IS 6 *
*****
```

Figure 9-8 Supplier copybook



## Mapping the table

First we define the sequential table in our local data model, then we define the corresponding the copybook, and finally we go into the details of defining the sequential table by using the classic data architect (DWHRES\_MODEL). Figure 9-9 shows how to start the sequential table configuration.

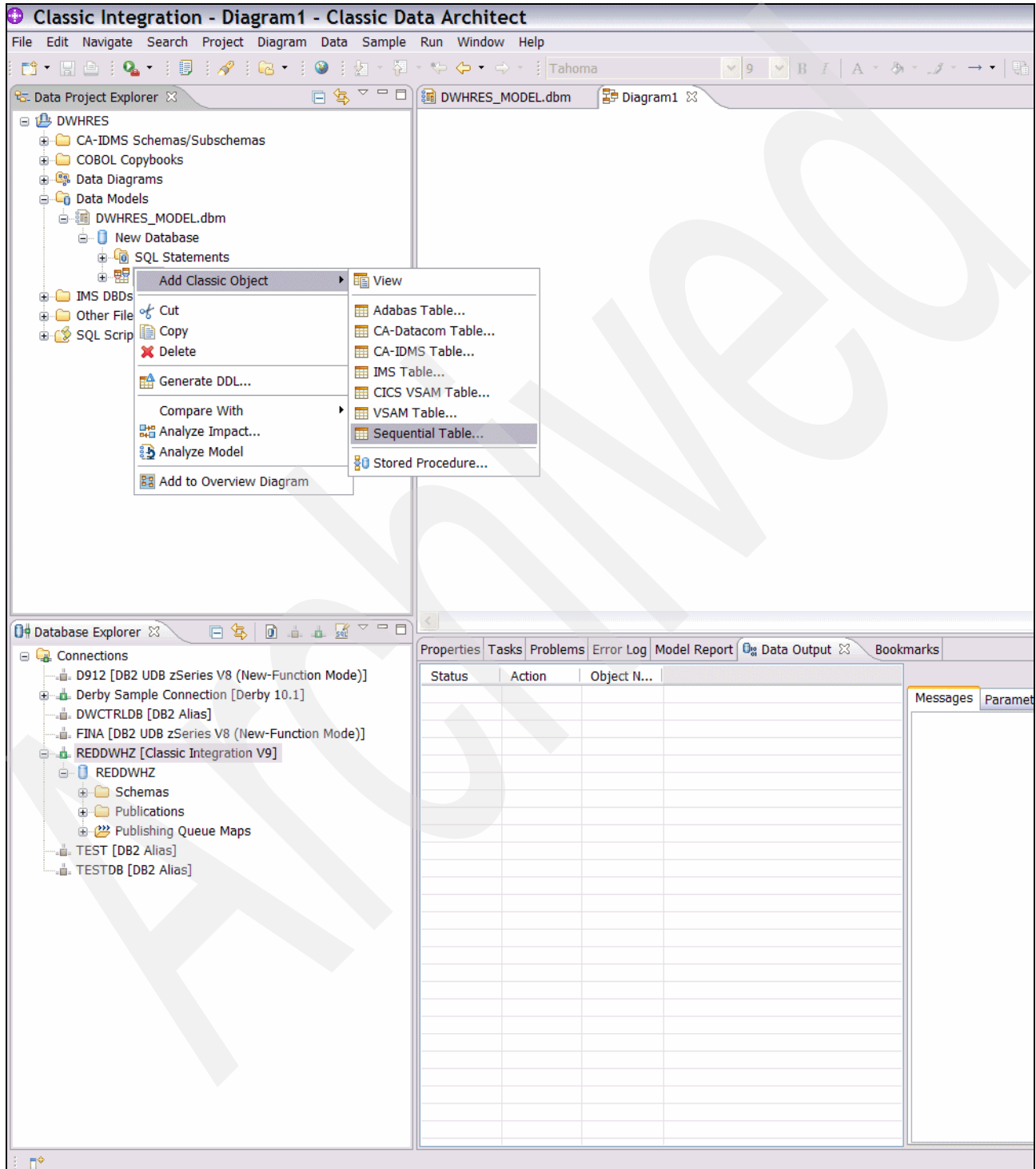


Figure 9-9 Mapping a sequential table

In the New Sequential Table – Specify Location and COBOL Copybook window (Figure 9-10), we specify the copybook and schema name for the sequential table.


The screenshot shows a Windows-style dialog box titled "New Sequential Table" with a close button (X) in the top right corner. Below the title bar is a tab labeled "Specify Location and COBOL Copybook". A small icon of a database cylinder with an arrow is to the right of the tab. The main area contains the following fields and controls:

- Database model: A text box containing "\DWHRES\DWHRES\_MODEL.dbm" and a "Browse..." button to its right.
- Copybook: A text box containing "\DWHRES\SupplierCopyBook.cpy" and a "Browse..." button to its right.
- Schema name: A dropdown menu showing "CAC".
- Below these fields is a section titled "Create View" with two radio buttons: "No" (which is selected) and "Yes".

At the bottom of the dialog box is a footer bar containing a help icon (question mark), and four buttons: "< Back", "Next >", "Finish", and "Cancel".

Figure 9-10 Parameters for the sequential table

In the New Sequential Table – Specify Sequential Information window (Figure 9-11), we specify the table name and the dataset name that it maps to in the system. We can also specify a DD name here but it name should be coded in the data server job that is running on the host. The data in the data set in the figure is described in “SUPPLIER data” on page 82.



The image shows a software window titled "New Sequential Table" with a subtitle "Specify Sequential Information". Below the subtitle is a descriptive sentence: "Use this page to specify the sequential file which is being mapped as a table." The window contains several input fields and sections:

- Select 01 level:** A dropdown menu with "DCLSUPPLIER" selected.
- Table name:** A text field containing "SUPPLIER\_SEQ".
- Dataset:** A section with two radio buttons, "DS" (selected) and "DD". Below them is a text field for "Name:" containing "GAURAN.WCF.SUPPLIER".
- Record exit:** A section with two text fields, "Name:" and "Maximum length:", both currently empty.
- Comment:** A large text area for comments, currently empty.
- Navigation:** At the bottom, there are four buttons: "< Back", "Next >", "Finish", and "Cancel". A help icon (?) is also present.

Figure 9-11 Sequential table mapping - Table name and dataset name

[illegible]

*Figure 9-12 Summary of fields*

After the sequential table is defined in the local data model, we define it in the data server by using the Generate DDL function as shown in Figure 9-13.

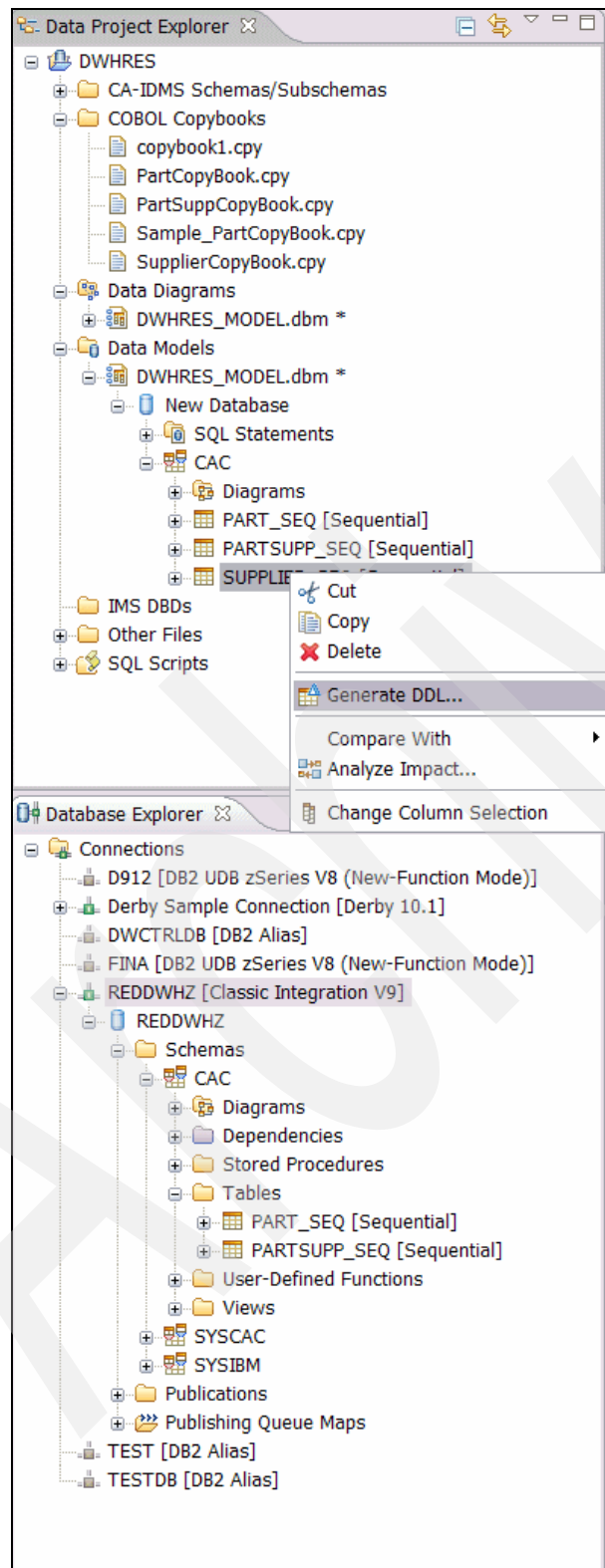


Figure 9-13 Generate DDL

We edit the generated DDL to change the fields defined as USE AS INTEGER or USE AS DECIMAL to USE AS CHAR because we are reading from a sequential file in which data is stored in a character format. Figure 9-14 shows the updated DDL.

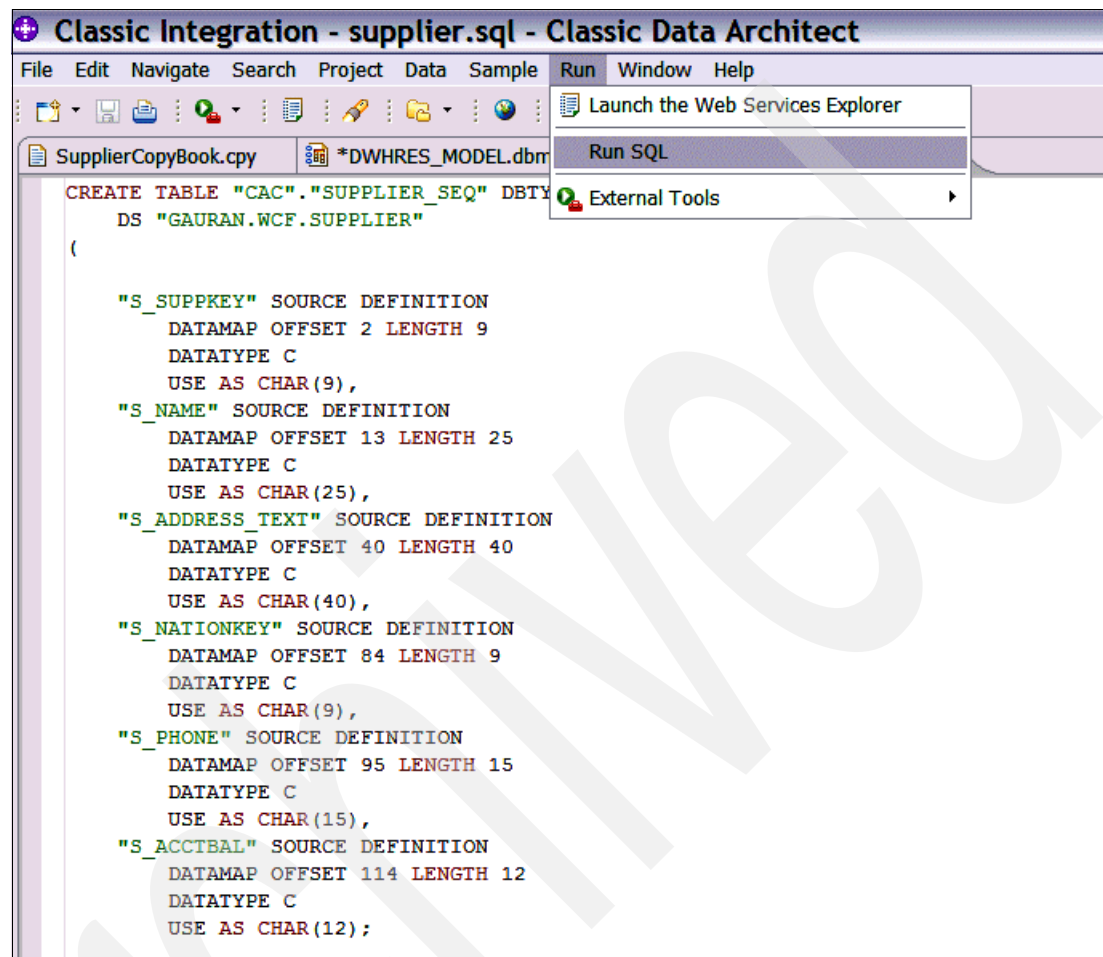


Figure 9-14 Updated SUPPLIER sequential table DDL

In the Connection Selection window, we run the edited DDL on the data server that is running on the host as shown in Figure 9-15.

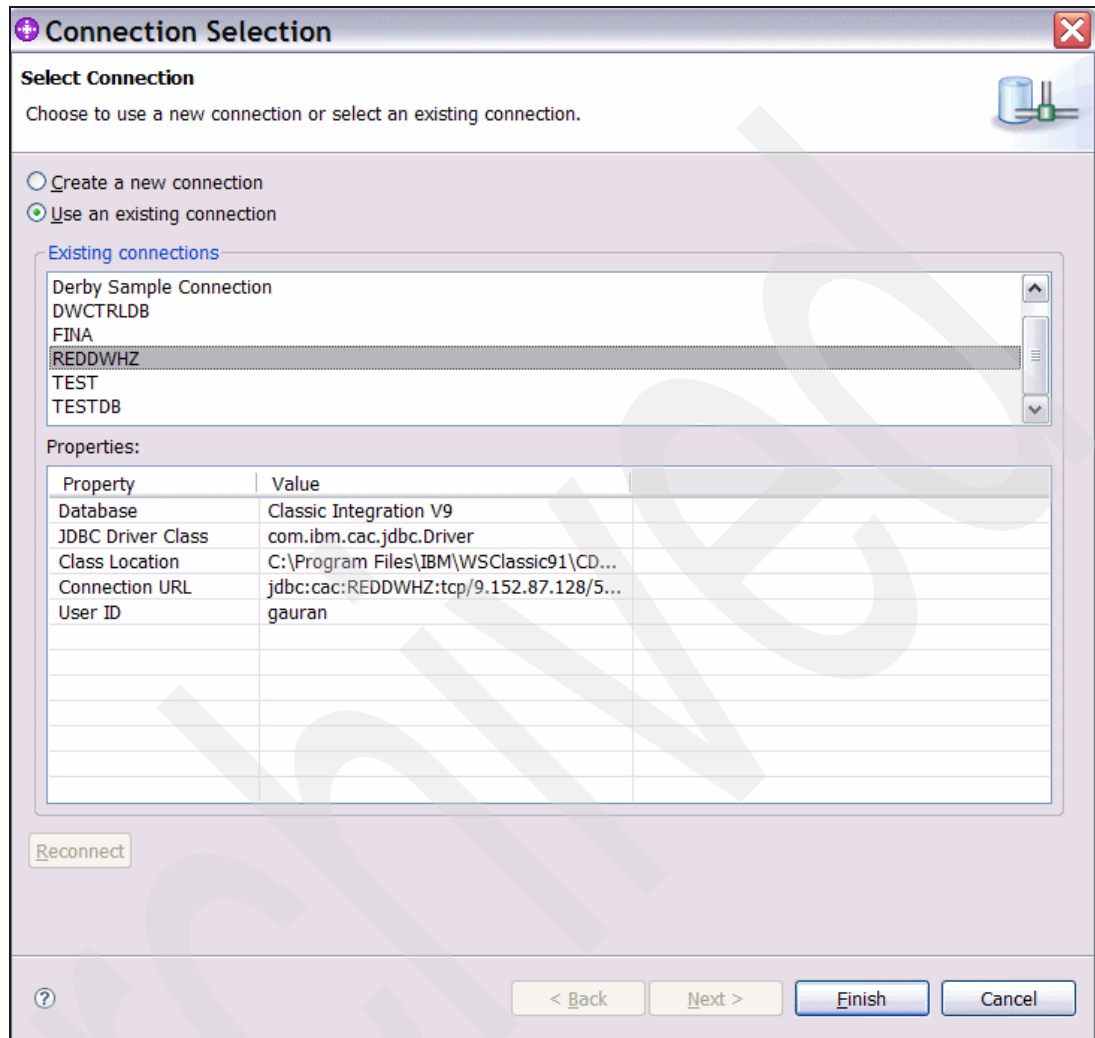


Figure 9-15 Running the DDL on the data server

We define the sequential table on the data server as shown in Figure 9-16.

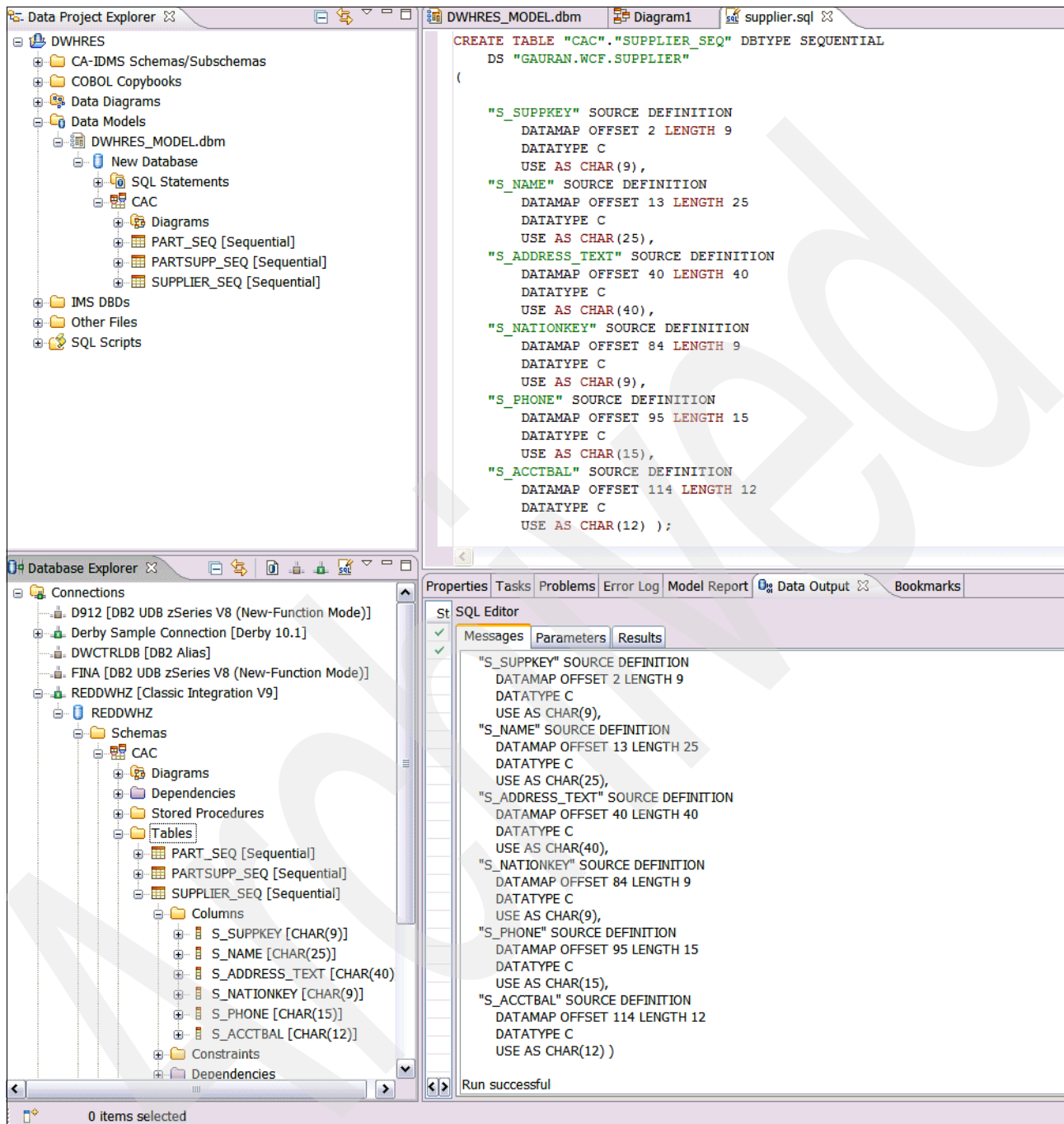


Figure 9-16 Running the DDL on the data server



Figure 9-17 shows the table and its contents displayed from Classic Data Architect.

The screenshot displays the Classic Data Architect interface. The top pane shows the SQL definition for the 'SUPPLIER\_SEQ' table. The bottom right pane shows the 'Sample Contents' of the table, which includes columns for S\_SUPPKEY, S\_NAME, S\_ADDR..., S\_NATIO..., S\_PHONE, and S\_ACCT....

**SQL Definition:**

```
CREATE TABLE "CAC"."SUPPLIER_SEQ" DBTYPE SEQUENTIAL
DS "GAURAN.WCF.SUPPLIER"
(
  "S_SUPPKEY" SOURCE DEFINITION
    DATAMAP OFFSET 2 LENGTH 9
    DATATYPE C
    USE AS CHAR(9),
  "S_NAME" SOURCE DEFINITION
    DATAMAP OFFSET 13 LENGTH 25
    DATATYPE C
    USE AS CHAR(25),
  "S_ADDRESS_TEXT" SOURCE DEFINITION
    DATAMAP OFFSET 40 LENGTH 40
    DATATYPE C
    USE AS CHAR(40),
  "S_NATIONKEY" SOURCE DEFINITION
    DATAMAP OFFSET 84 LENGTH 9
    DATATYPE C
    USE AS CHAR(9),
  "S_PHONE" SOURCE DEFINITION
    DATAMAP OFFSET 95 LENGTH 15
    DATATYPE C
    USE AS CHAR(15),
  "S_ACCTBAL" SOURCE DEFINITION
    DATAMAP OFFSET 114 LENGTH 12
    DATATYPE C
    USE AS CHAR(12) );
```

**Table Contents:**

S_SUPPKEY	S_NAME	S_ADDR...	S_NATIO...	S_PHONE	S_ACCT...
2	Supplier#...	89eJ5ksX...	5	15-679-8...	4032.68
3	Supplier#...	q1,G3Pj6...	1	11-383-5...	4192.40
4	Supplier#...	Bk7ah4CK...	15	25-843-7...	4641.08
5	Supplier#...	Gcdm2rJ...	11	21-151-6...	-283.84
6	Supplier#...	tQxuVm7...	14	24-696-9...	1365.79
7	Supplier#...	s4TicNG...	23	33-990-9...	6820.35
8	Supplier#...	9Sq4bBH...	17	27-498-7...	7627.85
9	Supplier#...	1KhUgZe...	10	20-403-3...	5302.37
10	Supplier#...	Saygah3g...	24	34-852-4...	3891.91
11	Supplier#...	JfwTs,LZr...	18	28-613-9...	3393.08
12	Supplier#...	aLIW q0...	8	18-179-9...	1432.69
13	Supplier#...	HK71HQy...	3	13-727-6...	9107.22
14	Supplier#...	EXsnO5p...	15	25-656-2...	9189.82
15	Supplier#...	oIXVbNBf...	8	18-453-3...	308.56
16	Supplier#...	YjP5C5z...	22	32-822-5...	2972.26
17	Supplier#...	c2d,ESHR...	19	29-601-8...	1687.81
18	Supplier#...	PGGVE5P...	16	26-729-5...	7040.82
19	Supplier#...	edZT3es...	24	34-278-3...	6150.38
20	Supplier#...	iybAE,Rm...	3	13-715-9...	530.82
21	Supplier#...	81Cavellc...	2	12-253-5...	9365.80
22	Supplier#...	okiiQfK 8l...	4	14-144-8...	-966.20
23	Supplier#...	ssetugTc...	9	19-559-4...	5926.41
24	Supplier#...	C4nPVLrV...	0	10-620-9...	9170.71

Figure 9-17 Supplier sequential table contents

**Note:** The method described in “Mapping the table” on page 217 is for the SUPPLIER table. We follow a similar process to map the PART and PART SUPPLIER table.

### 9.4.3 Installing and configuring the WebSphere Classic Federation client on Linux on System z

To access a data source through WebSphere Classic Federation on Linux on System z and to connect to it from Information Server, you have to install the WebSphere Classic Federation client on Linux on System z and define an ODBC data source for metadata retrieval.

**Note:** There is no dedicated client for WebSphere Classic Federation. You must install the WebSphere Classic tools, which include the client code for connection to a WebSphere Classic Federation server. In the following description, we discuss the WebSphere Federation client.

#### Installing WebSphere Classic Federation client on Linux on System z

The following support document points to the download information for the Linux on System z client:

<http://www.ibm.com/support/docview.wss?uid=swg21291254>

You can download this information directly from the following FTP address:

<ftp://ftp.software.ibm.com/software/data/integration/iicf/>

The file for Linux on System z at the time when we wrote the book was *cac91zlx\_120807*. This file contains a script and the binaries and can be run directly from the Linux on System z shell. To install the client:

1. Log in as root and enter the following command:

```
./cac91zlx_120807 -console
```

**Note:** The client installation uses a GUI by default. Without an X server, you can add the `-console` parameter to run it on the console.

2. After the Java SE Runtime Environment (JRE™) is extracted and you see the text in Figure 9-18, press Enter to accept the default selection. Then type 1 to continue.

```
Welcome to the InstallShield Wizard for IBM WebSphere Classic Tools V9.1

The installation wizard will install the following tools and clients for
Classic Federation Server, Event Publisher and Replication Server for z/OS :
- ODBC driver and CLI client
- JDBC client

Click Next to continue.

Press 1 for Next, 3 to Cancel or 5 to Redisplay [1]
```

Figure 9-18 Installation wizard for WebSphere Classic Federation client installation

3. In the next panel (Figure 9-19), install the ODBC and CLI client, the JDBC client, or both clients. Select both available options and press Enter to confirm the current selection.

```
Select the features for IBM WebSphere Classic Tools that you would like to
install:

IBM WebSphere Classic Tools, V9.1

To select/deselect a feature or to view its children, type its number:

1. [x] ODBC driver and CLI client (installed)
2. [x] JDBC client (installed)

Other options:

0. Continue installing

Enter command [0]
```

Figure 9-19 Selecting installation options for the WebSphere Classic Federation client

4. In the next panel (Figure 9-20), confirm the proposed default path (/opt/ibm/wsclassic91) and press Enter. This is the path where the WebSphere Classic Federation client will be installed.

```
IBM WebSphere Classic Tools, V9.1 will be installed in the following location:

/opt/IBM/wsclassic91

with the following features:

ODBC driver and CLI client
JDBC client

for a total size:

75.9 MB

Press 1 for Next, 2 for Previous, 3 to Cancel or 5 to Redisplay [1]
```

Figure 9-20 Installation directory for the WebSphere Classic Federation client

The installation procedure lays down the file in the specified directory (showing the progress of the installation based on percent complete).

5. After some seconds, when you see the screen that indicates a successful installation (Figure 9-21), type 3 to finish the installation procedure and press Enter.

```
The InstallShield Wizard has successfully installed IBM WebSphere Classic
Tools, V9.1. Choose Finish to exit the wizard.

Press 1 for Next, 3 to Finish or 5 to Redisplay [1] 3
```

Figure 9-21 Confirmation message for a successful client installation

## Configuring the WebSphere Classic Federation client

During configuration of the WebSphere Classic Federation server on z/OS, we chose REDDWHZ as the data source name, which we must use in the client configuration as well.

The settings for the WebSphere Classic Federation client are stored in the cac.ini file, which is located by default in /opt/IBM/wsclassic91/cli/lib. Edit this file as follows and as shown in Example 9-13.

For the DATASOURCE name, type REDDWHZ. This is the name of the remote data source that matches the service name of the service information entry parameter in the query processor task on the data server.

The IP address 10.10.10.2 is the (HiperSocket) IP address to the z/OS system and port 5999 is the value that has been defined during setup of WebSphere Classic Federation Server on z/OS.

For the DEFLOC parameter, which must point to a valid WebSphere Classic Federation data source, type REDDWHZ.

*Example 9-13 The cac.ini file*

---

```
NL CAT = /opt/IBM/wsclassic91/cli/lib
NL = US English
* user id/pwd needed for catalog security
USERID = cacuser
USERPASSWORD = cacpwd
* default datasource location
DEFLOC = REDDWHZ
DATASOURCE = REDDWHZ tcp/10.10.10.2/5999
* performance and memory parameters
FETCH BUFFER SIZE = 32000
MESSAGE POOL SIZE = 4000000
* codepage parameters
SERVER CODEPAGE = IBM-037
CLIENT CODEPAGE = IBM-850
```

---

Additional configuration steps, such as setting environment variables and configuring an ODBC data source, are required to access a WebSphere Classic Federation Server from within an Information Server environment. These settings are described in 9.5.6, “Cataloging the DB2 for z/OS subsystems” on page 248, and 9.5.7, “Setting up ODBC connections to DB2 for z/OS databases and WebSphere Classic Federation” on page 248.

## 9.5 Installing IBM Information Server

We use Information Server version 8.0.1 with fix pack 1 for this project. We install it on a Linux on System z that runs on an LPAR of our System z.

We use the server install image to install server components on Linux on System z. In addition, we use a client install image to install components, such as the Designer and Administrator client, on a Windows machine.

**Naming and terminology:** The announced next version for IBM Information Server comes with slightly different naming and terminology. For this book, we tried to use the new names, such as InfoSphere DataStage, where it is appropriate and where we feel it does not cause confusion. The figures reflect the names of Information Server 8.0.1.

## 9.5.1 Topology considerations: Product tiers

**Note:** With Information Server 8.0.1, the topology is defined with tiers. The next version uses the term “tier” instead and comes with a slightly different grouping. Because we refer to this topology in the installation steps, we use the 8.0.1 terminology here.

An typical Information Server installation involves the following five tiers, which are illustrated in Figure 9-22 on page 230:

► Client tier

The clients for IBM Information Server include InfoSphere DataStage and InfoSphere QualityStage clients and the Information Server console. InfoSphere Information Analyzer and InfoSphere Information Services Director are part of the IBM Information Server console. The clients can be installed by the suite installer or stand-alone client installer.

For this project, we need and use the following components:

- InfoSphere DataStage and InfoSphere QualityStage Administrator to create and define projects
- InfoSphere DataStage and InfoSphere QualityStage Designer to model ETL jobs that cleanse, transform, and integrate data
- InfoSphere DataStage and InfoSphere QualityStage Director to validate, schedule, run, and monitor our ETL jobs

We do not use and show InfoSphere Information Analyzer and InfoSphere Service Director™ in this book.

For this book, the client components are installed on a Windows system.

► Repository tier

The repository contains the metadata for the Information Server. The metadata server, hosted by the application server, connects to the repository to store and retrieve data.

**Note:** With version 8.0.1 of the Information Server, the repository cannot be hosted by a database on DB2 for z/OS. For this project, we decided to use a database on DB2 for Linux on System z instead. Future versions might support DB2 for z/OS for the repository as well.

► Services tier

The services tier includes InfoSphere Metadata Server, InfoSphere Business Glossary, Information Server Web console, the services components of the products that you install, and the information center.

For this book, we need and use the Information Server Web console primarily to create users and authorize them to use components and features of Information Server.

► Engine tier

The engine tier consists of the engine components of the products that you install, such as InfoSphere DataStage and InfoSphere QualityStage. The engine tier also includes WebSphere Federation Server. The services tier must be installed on the same computer or on a different computer in the same network before you install the engine tier.

For this book, we install the engine tier and the services tier on the same computer, which is our Linux on System z system.

We need the following components:

- Engine components

This is basically the component that runs the ETL jobs that we define in our projects. DataStage offers a server engine and a parallel engine to run respective job types.

- InfoSphere Metadata Server agent

This is a Java process that runs in the background and offers access to the metadata server.

- ODBC drivers

We need the ODBC drivers to access our remote DB2 for z/OS system and retrieve metadata information, such as tables definitions from there.

- Documentation tier

The documentation for the product modules and components that you install is available in several formats, including PDF, online, and information center.

You can distribute product components among multiple physical servers and install multiple engine components on multiple server for scaleout.

For the purpose of this book, we use a Windows XP machine for the client and documentation tiers (Computer A in Figure 9-22) and a single Linux on System z system for repository, services and engine tiers (Computer B in Figure 9-22).

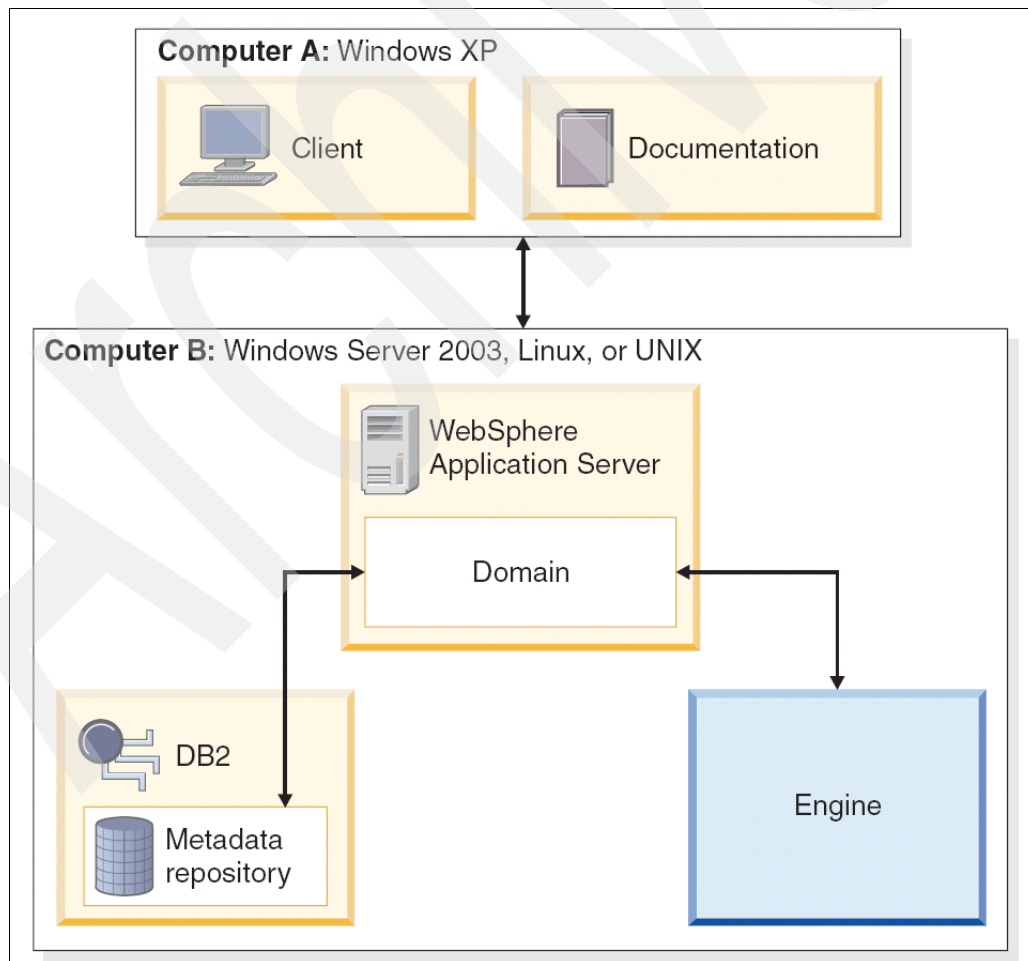


Figure 9-22 Information Server 8.0.1 tiers

## 9.5.2 User ID and registry considerations before installing Information Server

Several user IDs must be created to work with Information Server. They can be created by the installation procedure or you can create them beforehand.

We have the required user IDs created by the installation procedure. Table 9-2 lists the user IDs that we used for the Information Server installation. They are the defaults that are proposed by the installation procedure. The user IDs that are related to the repository tier are used to manage the DB2 system and access the repository database there.

Table 9-2 The Information Server user IDs

User ID	Group	tier	Required user
xmeta	N/A	Repository	Owner of the repository database, also used to access the repository.
db2inst1	db2iadm1	Repository	Owner of the DB2 instance on DB2 for Linux on System z.
db2fenc1	db2fenc1	Repository	Fenced user for DB2 instance on DB2 for Linux on System z.
dasusr1	dasadm1	Repository	DB2 administration (DAS) user.
wasadmin	N/A	Services	Application Server ID, required to start and stop WebSphere Application Server and manage installed applications (if required).
dsadm	dstage	Engine	InfoSphere DataStage administrator.
admin	N/A	Services	Information Server administrator; use this user ID to log into the Information Server Web console.

For the users who can log on to the software, you must decide if you want to define them in the operating system user registry or in Information Server's own internal user registry. There are pros and cons for both options and the decision ultimately should be taken based on requirements of the environment.

For this project, we use the operating system user repository because we want to use the user IDs and credentials for other software components on the same system. Therefore, having the same set of users created in a dedicated Information Server user repository results in two sets of credentials that users have to manage. If most of the users gain access to DataStage and do not access other software on the same system, using the internal registry might be a better choice.

## 9.5.3 Installing server components for Linux on System z

In the following steps, we describe the installation of Information Server for Linux on System z. We use the GUI-driven interactive installation method, which requires an X Window System running on the client machine, for example, Windows.

1. Log in as root or with the user ID that should own the installation.
2. Run `./install` from your server installation image on Linux on System z.

3. In the installation wizard window (Figure 9-23) that opens on your X Window System, click **Next**.

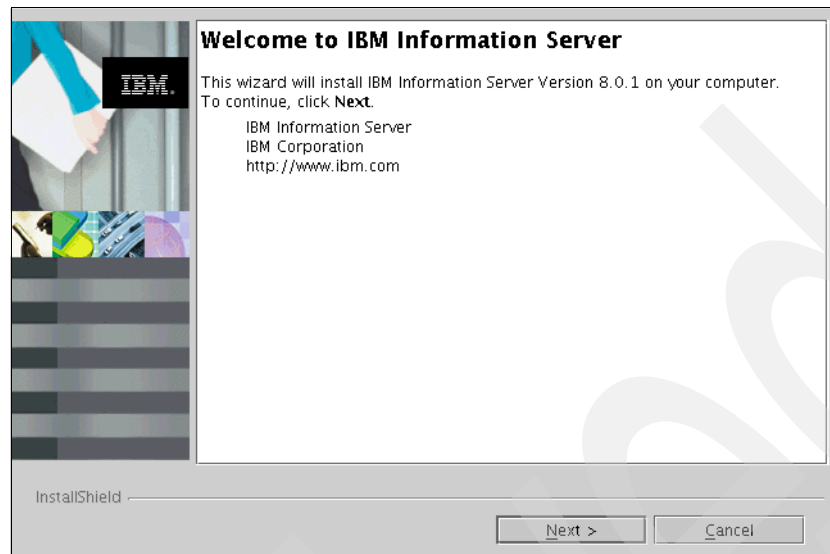


Figure 9-23 Welcome panel for the Information Server installation

4. In the next panel that shows the licensing agreement, click the **accept** option and then click **Next**.
5. Read the comment about the firewall restrictions during installation and click **Next**.
6. In the Installation and Response File Selection panel (Figure 9-24), install the suite, create a response file, or do both. By choosing to have a response file created, you can rerun the installation in silent mode again. Click **Next**.

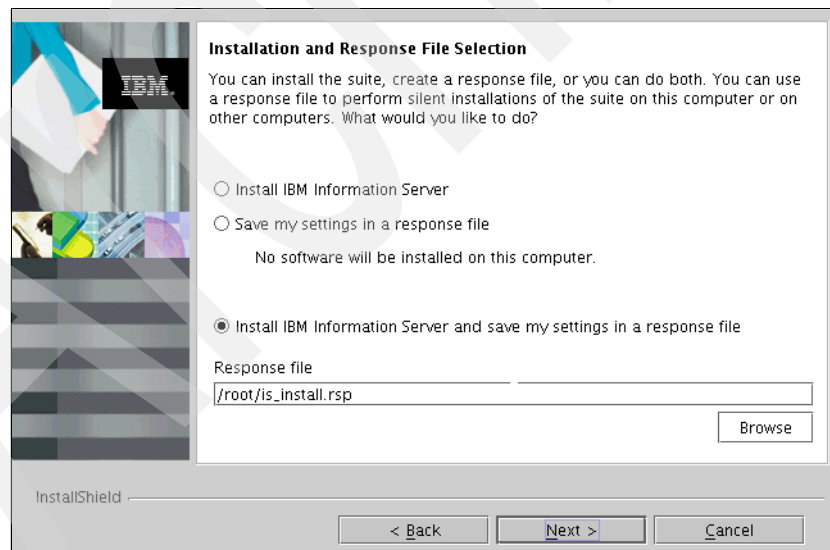


Figure 9-24 Installation option to create a response file



7. In the Installation Directory panel (Figure 9-25), specify the installation directory of Information Server. We accept the default proposal to install the components under /opt/IBM/InformationServer. Then click **Next**.

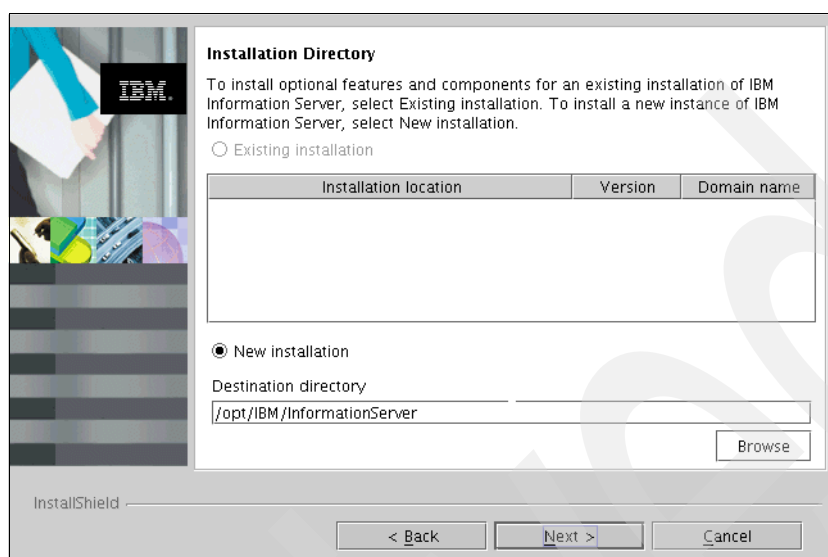


Figure 9-25 Choosing the installation directory

8. In the IBM Information Server Layers panel (Figure 9-26), select the service tiers to install. As explained in 9.5.1, "Topology considerations: Product tiers" on page 229, we install the engine, services and repository tiers on our server. For shared access to the documentation files, we include documentation. Therefore, we click **Select All**. Click **Next**.

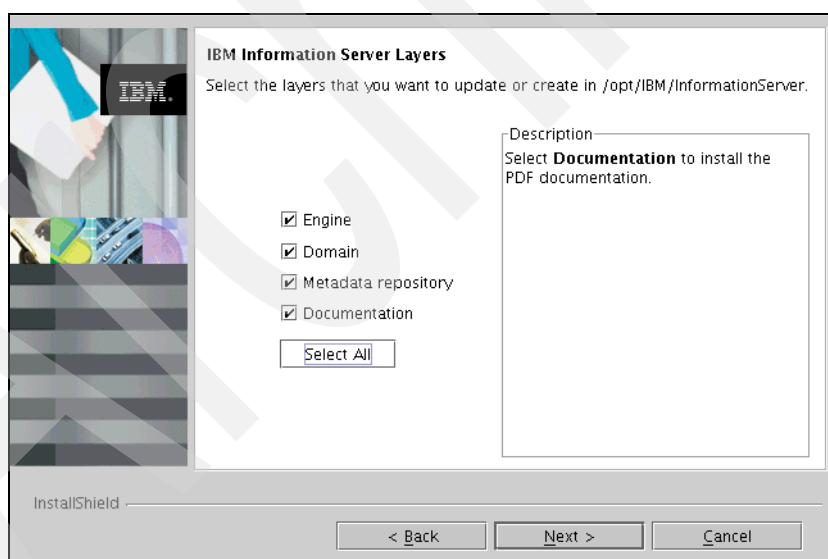


Figure 9-26 Selecting the server tiers for installation

9. In the next panel, point to the location of the XML license file, which is the license for your Information Server installation. Click **Next**.

10. In the Product Module and Component Select panel (Figure 9-27), select the available product component to install for the selected tiers. Make sure that you select **InfoSphere DataStage and QualityStage** and **Documentation**. The Metadata repository option is selected by default and cannot be cleared. Click **Next**.

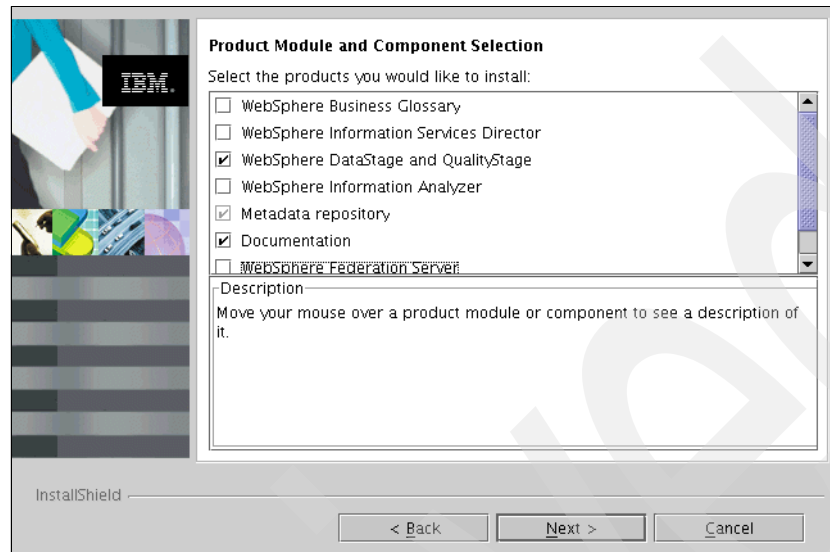


Figure 9-27 Selecting the product components to install

11. In the Installation Type panel (Figure 9-28), select **Typical** for the installation type and click **Next**.

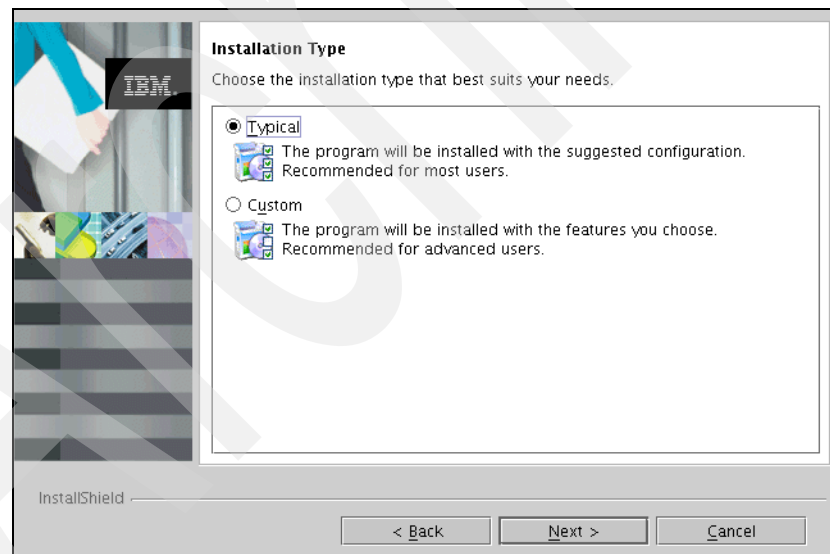


Figure 9-28 Selecting the installation type

12. In the next panel, upgrade the existing DataStage installation in a given directory. Because we run a new installation, select **New Installation** and click **Next**.

13. In the DB2 Version 9.1 Is Not Installed on Your Computer panel (Figure 9-29), read the message, which explains that you need a version of DB2 installed on the server and that the installation routine ensures that it is set up automatically for you. Click **Next**.

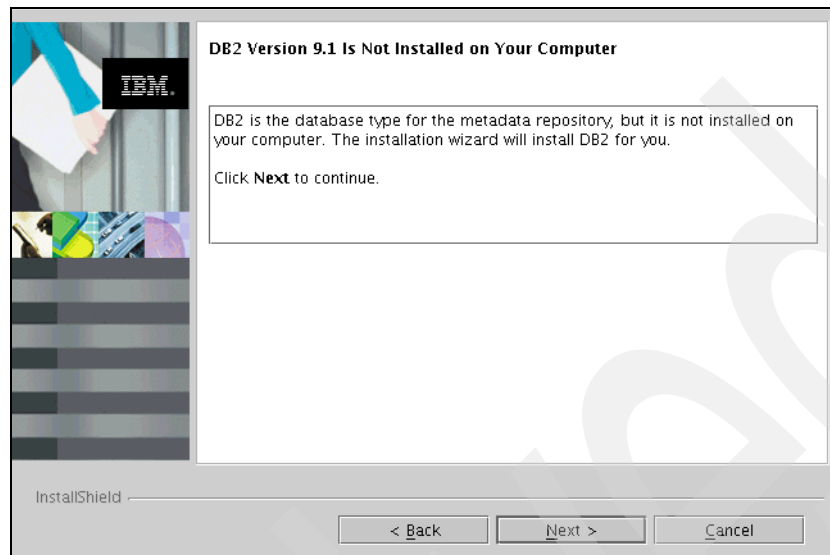
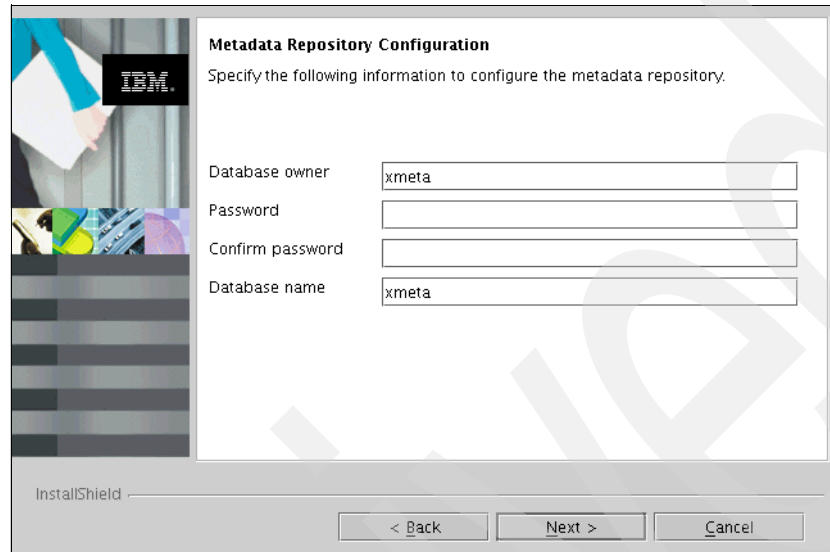


Figure 9-29 Message about DB2 installation

**Important:** If you have already run a DB2 instance on your Linux on System z, you might want to use an existing installation. However, make sure that the version you use is compatible with Information Server. For example, using DB2 9.5 for Linux on System z might not work together with Information Server. If in doubt, have Information Server install the DB2 version that comes with Information Server.

14. In the Metadata Repository Configuration panel (Figure 9-30), enter the user ID to use (or create) for creating and accessing the repository. As outlined in 9.5.2, “User ID and registry considerations before installing Information Server” on page 231, we use the suggested default user ID `xmeta` and have Information Server create the user ID for us. If the user ID already exists, specify the correct password. Click **Next** to validate the credentials.



**Metadata Repository Configuration**  
Specify the following information to configure the metadata repository.

Database owner	<input type="text" value="xmeta"/>
Password	<input type="password"/>
Confirm password	<input type="password"/>
Database name	<input type="text" value="xmeta"/>

InstallShield

< Back    Next >    Cancel

Figure 9-30 Specifying the user ID to create and access the repository

15. In the WebSphere Application Server Selection panel (Figure 9-31), choose whether you want to install a new version of WebSphere Application Server for the services tier or configure an existing one. Select **Install WebSphere Application Server** and click **Next**.

**Note:** Because Information Server requires special settings on the server instance, we recommend that you do not run other applications in the same instance. If you already have an existing and compatible installation of WebSphere Application Server on your Linux on System z, create a new profile and dedicate this one for Information Server. Otherwise, we recommend that you create a new instance of WebSphere Application Server.

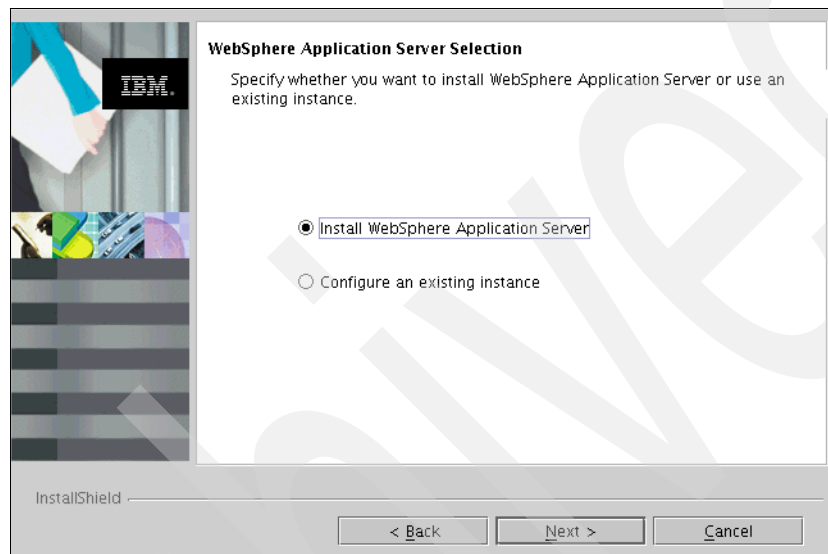


Figure 9-31 Installation option for WebSphere Application Server

16. Enter the directory in which to install your copy of WebSphere Application Server. We confirm the default of `/opt/IBM/WebSphere/AppServer` for this location and then click **Next**.

17. In the Choose the style of user registry panel (Figure 9-32), choose whether to use an internal user registry or a local OS user registry. We choose **Local OS User Registry** and type the wasadmin user ID for WebSphere Application Server management. See 9.5.2, “User ID and registry considerations before installing Information Server” on page 231, for a discussion about user registry selection and the list of required user IDs. Click **Next**.

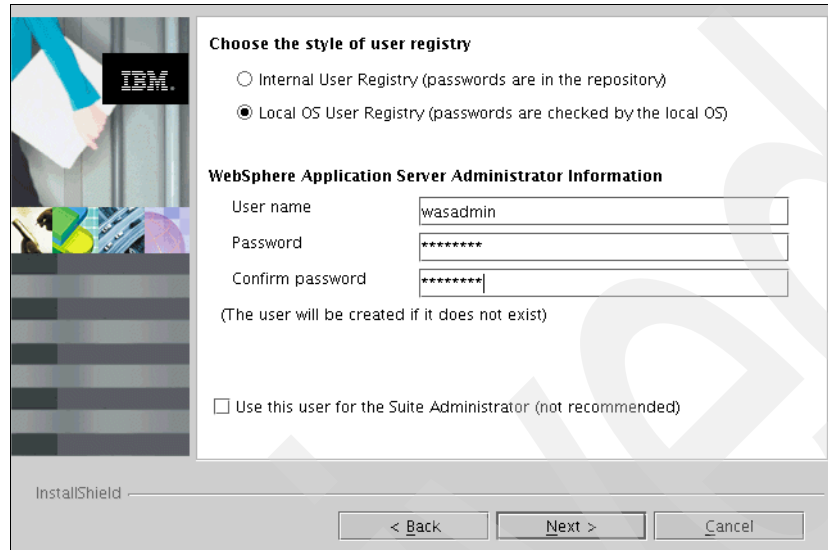


Figure 9-32 User registry selection and WebSphere Application Server user ID

18. In the IBM Information Server Administrator Information panel (Figure 9-33), log in to the Information Server Web console. As described in 9.5.2, “User ID and registry considerations before installing Information Server” on page 231, we use admin for the user ID. Enter the password of the existing user or the one to be created and click **Next**.

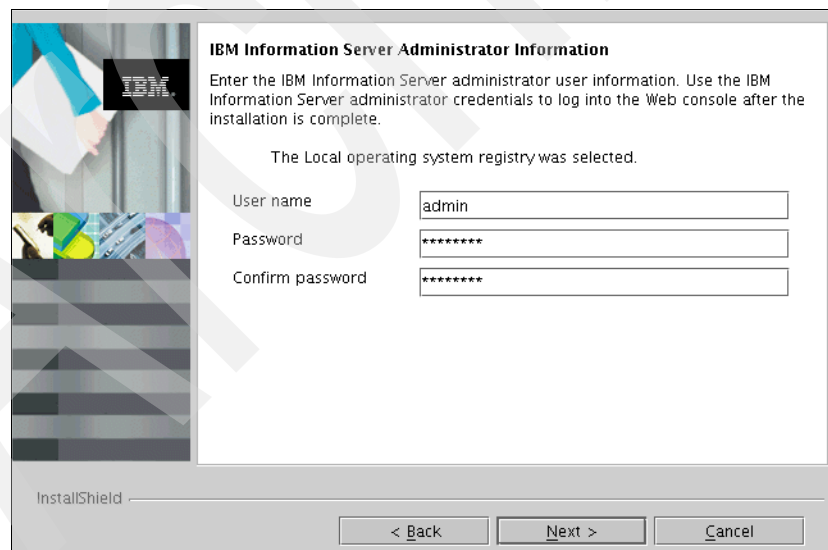


Figure 9-33 Selecting the admin user ID for the Information Server Web console

19. In the WebSphere DataStage Projects panel (Figure 9-34), create an initial DataStage project.

- a. In the empty project list, click **New Project** to name a new project to create.

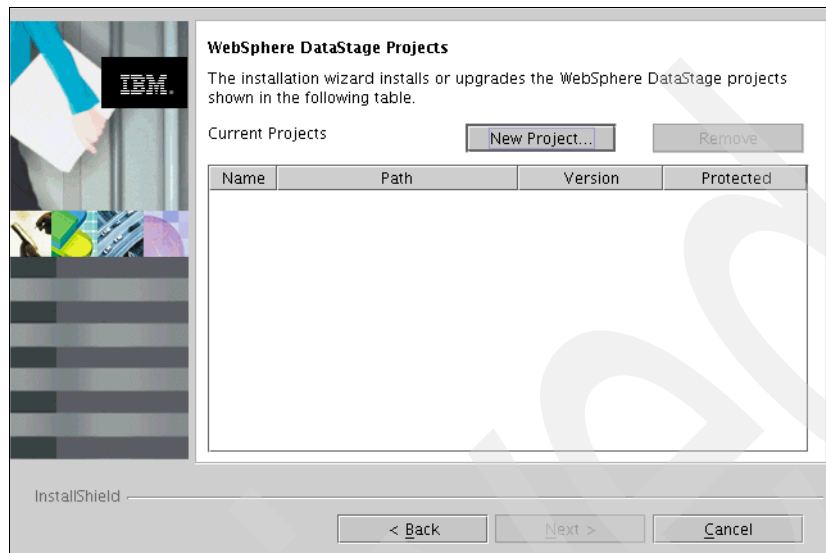


Figure 9-34 DataStage project list during installation

- b. In the next window that opens (Figure 9-35), type the name of the new project. We type a project name of p1, leave the project path at its default location, and click **OK**. The project list in Figure 9-34 now shows the newly defined project p1.

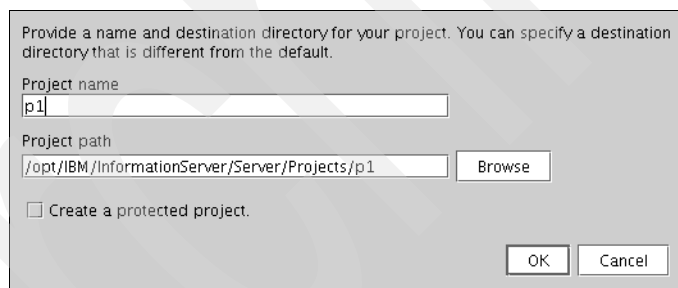


Figure 9-35 Defining a new DataStage project during installation

- c. Back in the WebSphere DataStage Projects panel, click **Next**.

20. The next steps cover the installation of DB2 for Linux on System z. The first step (Figure 9-36) asks for the target location for the DB2 installation. In our case, we choose the default location `/opt/IBM/db2/v9`. Click **Next**.

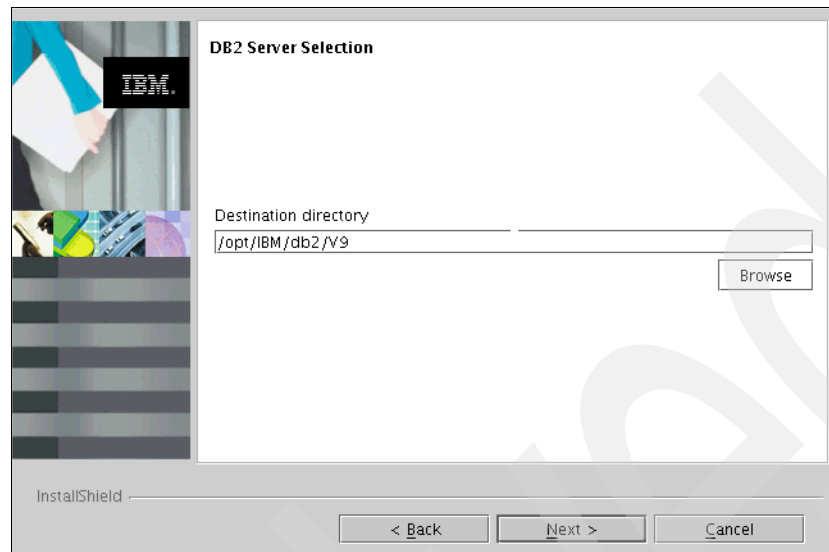


Figure 9-36 Installation location for DB2 for Linux on System z

21. In the next panels, provide the user IDs and passwords to use for the DB2 instance. According to 9.5.2, “User ID and registry considerations before installing Information Server” on page 231, we defined the user IDs `dasuser1`, `db2inst1`, and `db2fenc1`. Add a desired password and accept the suggested defaults. Click **Next** on each panel.
22. In the WebSphere DataStage Administrator panel (Figure 9-37), define one user ID on the operating system level. This user also must be part of the `dstage` group. You must do this regardless of whether you choose the operating system user registry or the DataStage internal registry to manage your users. This ID is effectively used to administer DataStage. Provide the password and for Group name, type `dstage`. Confirm and click **Next**.

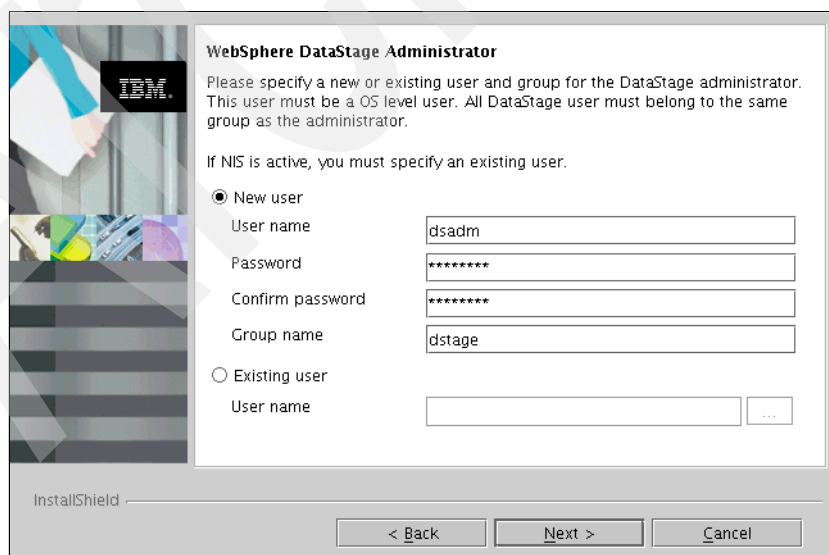


Figure 9-37 Defining the InfoSphere DataStage administrator user



23. In the DataStage Instance Tag panel (Figure 9-38), type a DataStage instance tag. If multiple instances of DataStage are installed, make sure that they have different names and port numbers. For our installation, we verify that port number 31538 is not used by any other application on our server. We then keep the defaults. Click **Next**.

**Note:** Check /etc/services on Linux on System z to see which ports are registered. After your installation, you will find an entry like the following example in the file:

```
dsrpc          31538/tcp # RPCdaemon
DSEngine@/opt/IBM/InformationServer/Server/DSEngine
```

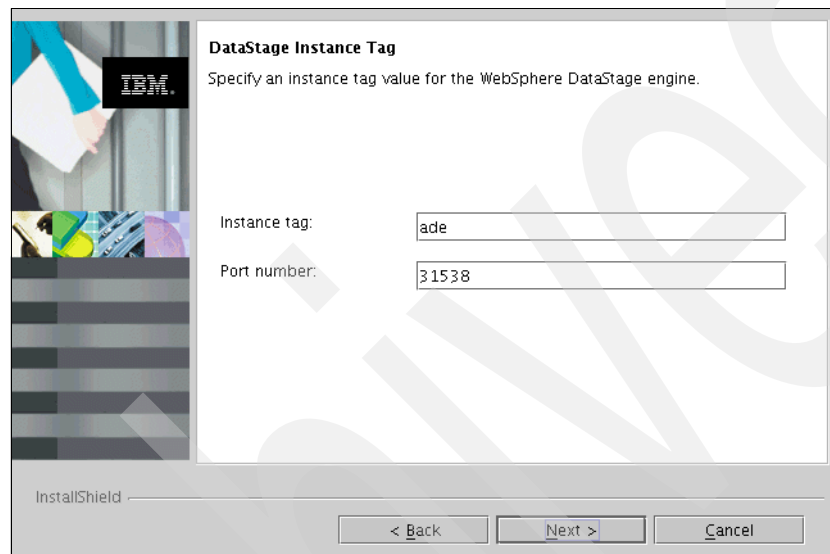


Figure 9-38 Name and port number for the DataStage instance

24. In the next panels, depending on your installation, install the following options:

- National Language Support
- IBM WebSphere MQ Plug-ins
- Oracle® Operator Configuration
- SAS Configuration

For our book, we do not use any of these options. To confirm not to install these options, click **Next** in the respective installation panels.

25. In the Pre-installation Summary panel (Figure 9-39), review the configuration decisions you made and confirm them before the procedure starts to install DataStage and QualityStage on the server. Click **Install** to proceed.

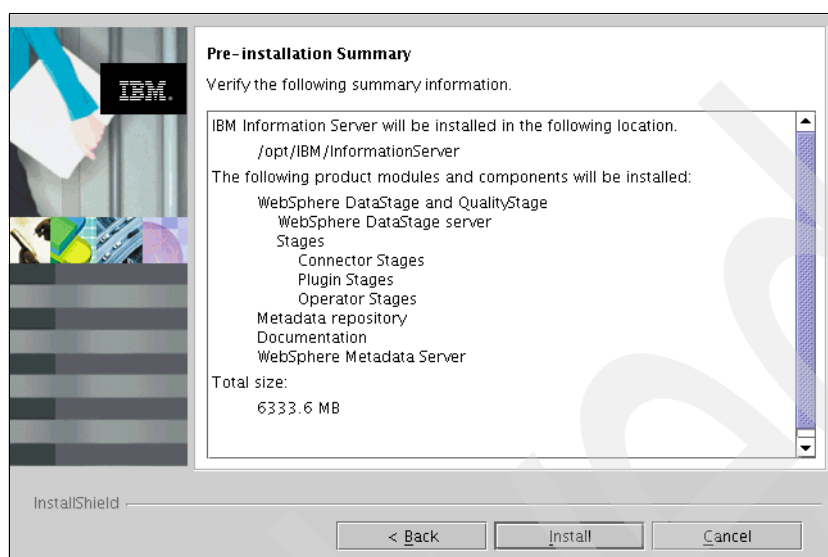


Figure 9-39 Summary before installation of DataStage and QualityStage

**Note:** During installation, you might encounter an warning message like the following example:

Warning: / partition has insufficient space to install the items selected.  
1217.4 MB additional space would be needed to install the selected items.

In this case, work with the Linux volume manager or disk partitioning utilities to provide more disk space for the file system where you plan to install Information Server.

This completes the installation of the Information Server on Linux on System z.

#### 9.5.4 Installing DataStage and QualityStage designer clients on Windows

The client for DataStage and QualityStage can be installed with the server installation image for Windows, or you can use a dedicated client installation image. We describe and choose the client installation image option because we do not need additional server components on Windows and, therefore, prefer the smaller installation image.

1. Start the setup installation program on your Windows client (Figure 9-40) and click **Next**.



Figure 9-40 Starting the installation of DataStage and QualityStage client on Windows

2. Read and accept the license terms and click **Next**.
3. Read the suggestions to disable your firewall and click **Next**.
4. In the Installation and Response File Selection panel (Figure 9-41), select to install the suite, create a response file, or do both. By creating a response file, your installation options are recorded for later review and for running a silent installation without GUI interaction. This is particularly useful if you plan to install the client on many Windows systems and you do not want to repeat the GUI.

In our case, we click **Install IBM Information Server and save my settings in a response file** for later review. Click **Next**.

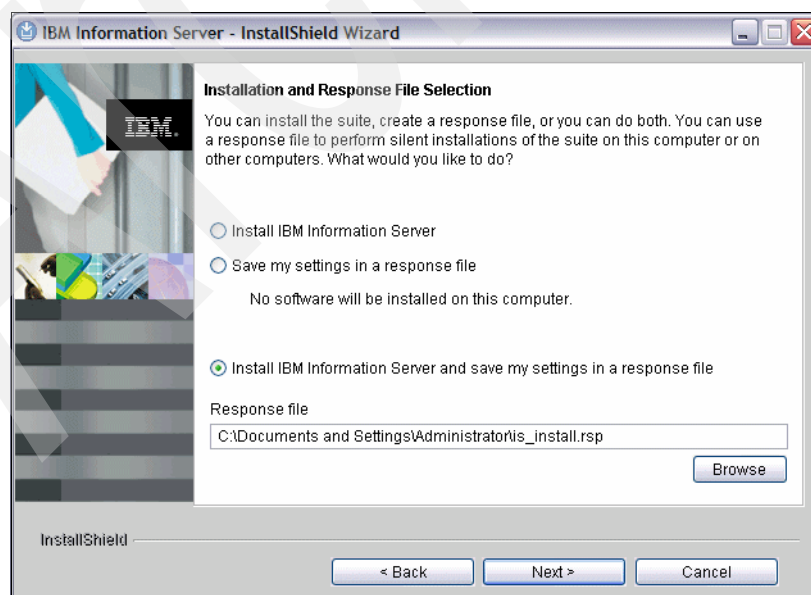


Figure 9-41 Installation option to create a response file

5. In the Installation Directory panel (Figure 9-42), specify an installation directory if one is on your Windows system. If an Information Server is already installed on your system, you are prompted to modify the existing one or you can create a new installation. Since we have not installed Information Server yet, we choose **New Installation**, and select the installation directory. Click **Next**.

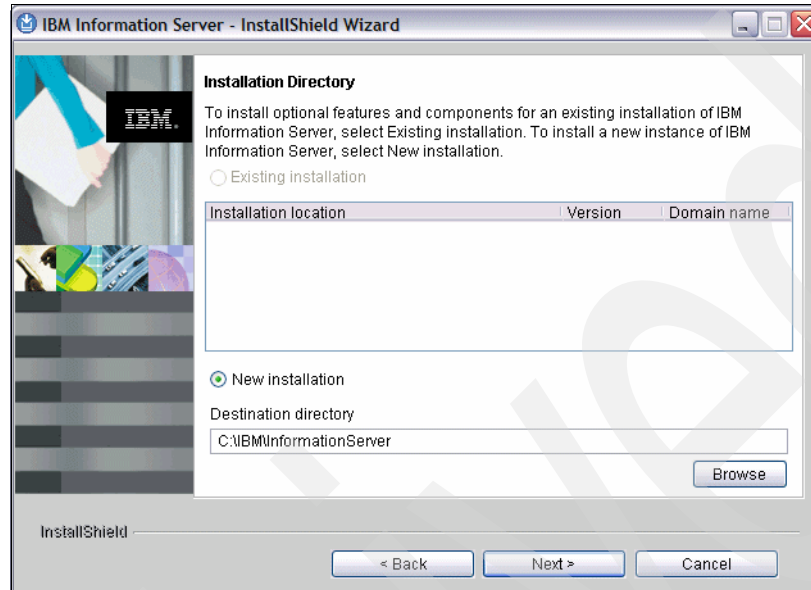


Figure 9-42 Choosing the installation directory

6. In the Product Module and Component Selection panel (Figure 9-43) select the products to install for the client. Select at least the first option **InfoSphere DataStage and QualityStage (client only)**. We recommend that you also install the documentation. The Information Analyzer and Information Services Director are additional components that you might want to use, but they are not crucial to build ETL jobs for our scenario. Then, click **Next**.

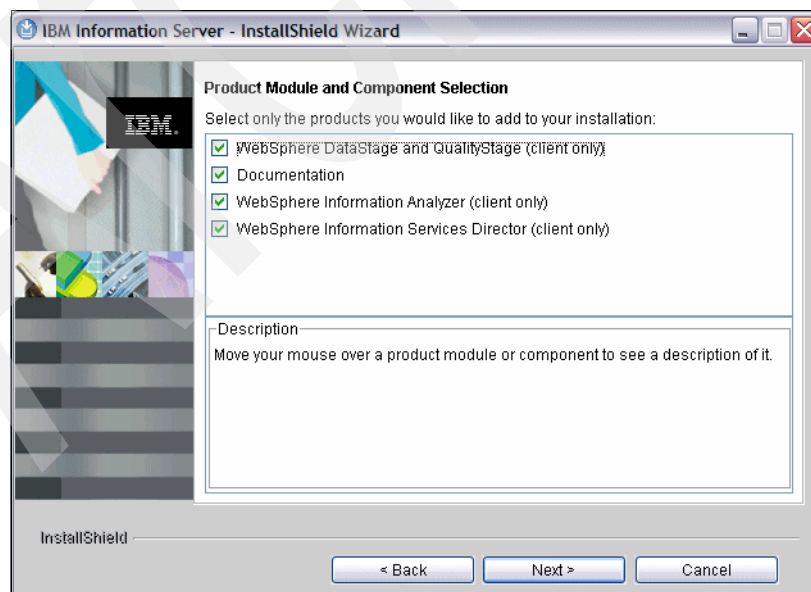


Figure 9-43 Installation options

7. In the Installation Type panel, select **Typical Installation** and click **Next**.
8. Depending on the options that you choose to install, when additional panels prompt for ODBC driver and template locations, confirm the defaults and click **Next** until you reach the summary page
9. In the Pre-installation Summary panel (Figure 9-44), review and confirm the components that you want to install. Then click **Next**.

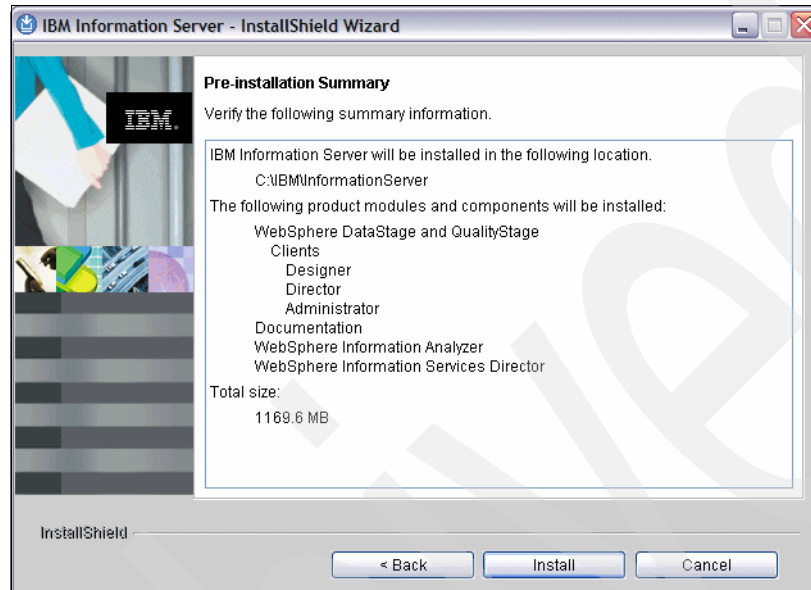


Figure 9-44 Pre-installation summary

The components are now installed on your Windows system. A post installation panel informs you that the installation was successful and lists the path to the response that was created. You are now ready to use the DataStage and QualityStage Designer and Administrator on your Windows system.

See 9.5.9, “Granting user access and creating a new DataStage project” on page 251, for an example where we use the DataStage and QualityStage Administrator to set up a new DataStage project.

### 9.5.5 Configuring DataStage to access DB2 for z/OS, WebSphere MQ, and WebSphere Classic Federation

During run time, the Information Server engine needs access to the client libraries of DB2, WebSphere MQ, and WebSphere Classic Federation on Linux on System z. In our environment, the 64-bit client libraries are in the following directories:

- ▶ DB2 runtime client: /opt/IBM/db2/V9/lib64
- ▶ WebSphere MQ libraries: /opt/mqm/lib64
- ▶ WebSphere Classic Federation: /opt/IBM/wsclassic91/lib/lib64

You must add these directories to the LD\_LIBRARY\_PATH environment variable for the effective user running an Information Server engine job that accesses DB2, WebSphere MQ, and WebSphere Classic Federation respectively.

You must also set the following environment variables for DB2 and WebSphere Classic Federation:

- ▶ DB2INSTANCE, which points to the DB2 instance to use
- ▶ CAC\_CONFIG, which points to the configuration file for WebSphere Classic Federation

There are two options to set LD\_LIBRARY\_PATH in DataStage:

- ▶ Modify the DataStage setting in the dsenv file.

This *preferred* option requires that you are logged in as root or a member of the dstage group. Edit dsenv in \$DSHOME, which points to /opt/IBM/InformationServer/Server/DSEngine in our installation, and add the highlighted lines in Example 9-14. The first highlighted line adds the 64-bit DB2 and WebSphere MQ libraries to the library path. The second group of highlighted lines defines the (local) DB2 instance to use.

*Example 9-14 Modifications in the dsenv file*

---

```
...
if [ -n "$DSHOME" ] && [ -d "$DSHOME" ]
then
    ODBCINI=$DSHOME/.odbc.ini; export ODBCINI
    HOME=${HOME:-/}; export HOME

    #LANG="<langdef>";export LANG
    #LC_ALL="<langdef>";export LC_ALL
    #LC_CTYPE="<langdef>";export LC_CTYPE
    #LC_COLLATE="<langdef>";export LC_COLLATE
    #LC_MONETARY="<langdef>";export LC_MONETARY
    #LC_NUMERIC="<langdef>";export LC_NUMERIC
    #LC_TIME="<langdef>";export LC_TIME
    #LC_MESSAGES="<langdef>"; export LC_MESSAGES

    LD_LIBRARY_PATH=~dirname $DSHOME~/branded_odbc/lib:~dirname
$DSHOME~/DSComponents/lib:~dirname
$DSHOME~/DSComponents/bin:$DSHOME/lib:$DSHOME/uvdl1s:$ASBHOME/apps/jre/bin:$ASBHOME/apps
/jre/bin/classic:$ASBHOME/lib/cpp:$ASBHOME/apps/proxy/cpp/linux-all-s390x_64:/opt/IBM/db
2/V9/lib64:/opt/mqm/lib64:/opt/IBM/wsclassic91/cli/lib/lib64:$LD_LIBRARY_PATH
    export LD_LIBRARY_PATH
fi

export DB2INSTANCE==db2inst1
export CAC_CONFIG=/opt/IBM/wsclassic91/cli/lib/cac.ini
...
```

---

- Modify the project settings.

In this option, you select the project. Then on the General tab, you click **Environment** and edit the LD\_LIBRARY\_PATH setting accordingly. Figure 9-45 shows the DataStage Administrator Client window that you reach. Note that this change only affects the settings for the selected projects. You must redo this for all projects where you want the setting to be applied.

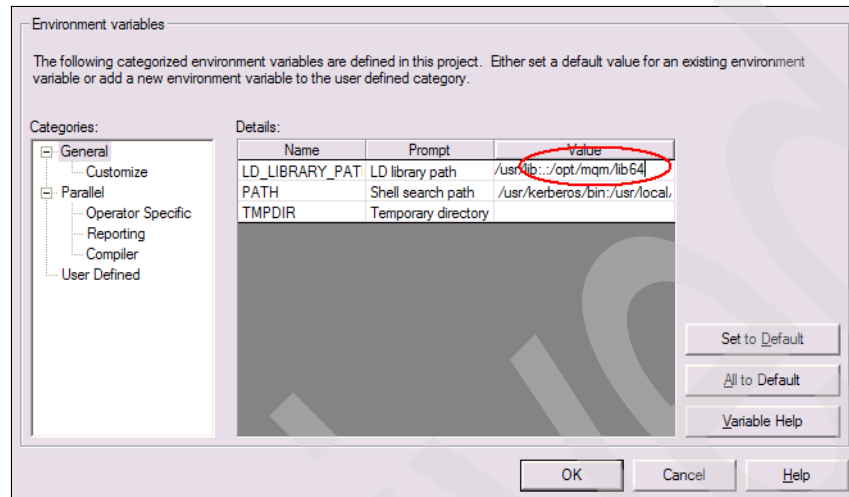


Figure 9-45 Editing the LD\_LIBRARY\_PATH for a DataStage project

If you do not add the WebSphere MQ libraries to the library path and run a job that has an MQ connector stage, you receive an error message in the job log such as the one in Example 9-15.

*Example 9-15 Exception in DataStage if MQ libraries are not accessible*

```
WebSphere_MQ_Connector_0: Error occurred during initializeFromArgs().
WebSphere_MQ_Connector_0: [IIS-CONN-WSMQ-000005] System call dlopen() failed with
OS error 2 (No such file or directory) (CC_WSMQUtil::resolveMqiMethodPointers(),
file CC_WSMQUtil.cpp, line 317)
```

**Note:** To access the WebSphere MQ client on Linux on System z, the effective user ID must be in the mqm group. We add all user IDs that need access to WebSphere MQ to the mqm group by editing the /etc/group file on our Linux on System z server.

DataStage accesses our two DB2 for z/OS subsystems, DWHD911 for OLTP data and DWHD912 for warehouse data, in two places:

- Access in a DataStage job is through cataloged databases, using DB2 Connect™. For DataStage jobs to access our DB2 for z/OS databases, we must catalog the two subsystems in our scenario on DB2 Connect on our Linux on System z.
- In DataStage, you define table metadata (column name and types) by using table definitions. You can import table definitions for existing tables through an ODBC connection to a database. For this option to work, you must set up an ODBC connection to DB2 for z/OS.

If you want to access WebSphere Classic Federation in your jobs, you need an ODBC connection definition for metadata retrieval and a configured WebSphere Classic Federation client setup as described in 9.4.3, “Installing and configuring the WebSphere Classic Federation client on Linux on System z” on page 226.

## 9.5.6 Cataloging the DB2 for z/OS subsystems

Our ETL processing needs access to two DB2 for z/OS subsystems: DWHD911 for OLTP (transactional) source data and DWHD912 for the target warehouse data.

You must have DB2 Connect installed and configured on Linux on System z to access the remote DB2 for z/OS subsystem. The Information Server installation also comes with an installation for DB2 on Linux on System z.

**Access license:** You need a valid license for DB2 Connect on Linux on System z in order to access the remote DB2 for z/OS system.

Use DB2 Connect and catalog the two subsystems. The commands in Example 9-16 assume that your HiperSocket connection to z/OS is assigned an IP address of 10.10.10.2 and that DWHD911 is accessible from remote on port 5911 and DWHD912 on port 5912 respectively.

*Example 9-16 Cataloging the DB2 subsystems*

---

```
-- catalog DWHD911
catalog tcpip node dwhd911 remote 10.10.10.2 server 5911 ostype mvs;
catalog database dwhd911 as dwhd911 at node dwhd911 authentication dcs
catalog dcs database dwhd912 as dwhd911 parms '.,,INTERRUPT_ENABLED'

-- catalog DWHD912
catalog tcpip node dwhd912 remote 10.10.10.2 server 5912 ostype mvs;
catalog database dwhd912 as dwhd912 at node dwhd912 authentication dcs
catalog dcs database dwhd912 as dwhd912 parms '.,,INTERRUPT_ENABLED'
```

---

## 9.5.7 Setting up ODBC connections to DB2 for z/OS databases and WebSphere Classic Federation

To register an ODBC connection for our two DB2 for z/OS subsystems (DWHD911 and DWHD912) and for WebSphere Classic Federation, perform the following steps when logged in as a user of the dstage group (or as root):

1. Change directory to \$DSHOME, which points to /opt/IBM/InformationServer/Server/DSEngine in our installation.
2. Run **dsenv** to set up environment variables in your current shell:  

```
. ./dsenv
```
3. Edit \$DSHOME/.odbc.ini to change the ODBC data source configuration and add definitions for our two subsystems and the WebSphere Classic Federation data source. Example 9-17 shows the definition for DWHD911, DWHD912, and REDDWHZ. Note that you must add references to the three new data source names in the [ODBC Data Sources] section at the top of the file.

*Example 9-17 ODBC data source definitions in file .odbc.ini*

---

```
[ODBC Data Sources]
DWHD911=Redbook OLTP Database
DWHD912=Redbook Warehouse Database
REDDWHZ=Classic Federation Server

[DWHD911]
Driver=/opt/IBM/InformationServer/Server/branded_odbc/lib/VMdb223.so
Description=DataDirect DB2 Wire Protocol Driver
```



```
AddStringToCreateTable=  
AlternateID=  
Collection=  
DynamicSections=100  
GrantAuthid=PUBLIC  
GrantExecute=1  
IpAddress=10.10.10.2  
IsolationLevel=CURSOR_STABILITY  
Location=DWHD911  
LogonID=  
Password=  
Package=  
PackageOwner=  
TcpPort=5911  
WithHold=1
```

**[DWHD912]**

```
Driver=/opt/IBM/InformationServer/Server/branded_odbc/lib/VMdb223.so  
Description=DataDirect DB2 Wire Protocol Driver  
AddStringToCreateTable=  
AlternateID=  
Collection=  
DynamicSections=100  
GrantAuthid=PUBLIC  
GrantExecute=1  
IpAddress=10.10.10.2  
IsolationLevel=CURSOR_STABILITY  
Location=DWHD912  
LogonID=  
Password=  
Package=  
PackageOwner=  
TcpPort=5912  
WithHold=1
```

**[REDDWHZ]**

```
Driver=/opt/IBM/wsclassic91/cli/lib/1ib64/libcacsqlcli.so  
Database=REDDWHZ
```

4. Edit the `uvodbc.config` file and add the data source names that should be available in DataStage projects. A master copy of this file is in the `$DSHOME` directory, and individual copies are in each project directory. If you edit `uvodbc.config` before you create your project, the definition in the master copy is copied to the project's version. If you created the project already, you must edit the configuration file in the project directory for the changes to take effect. Add the lines in Example 9-18 to the file.

---

*Example 9-18 The `uvodbc.config` file with settings for available ODBC data sources*

---

```
<DWHD911>  
DBMSTYPE = ODBC  
<DWHD912>  
DBMSTYPE = ODBC  
<REDDWHZ>  
DBMSTYPE = ODBC
```

---

**Important:** Add spaces before and after the equal sign. Otherwise you might encounter an error message such as the following example:

```
SQLConnect error:  Status = -1   SQLState = IM997   Natcode = 0
[SQL Client] An illegal configuration option was found
Invalid parameter(s) found in configuration file
```

5. Run the commands in Example 9-19 to bind the bind file to subsystem DWHD911. Do the same step for DWHD912. The bind file is located in the /opt/IBM/InformationServer/Server/DSCComponents/bin directory. Each user who accesses data through the DataStage DB2 z stage must have access to this bind file for the database the user wants to access. DataStage is using static SQL to access DB2 for z/OS.

*Example 9-19 DB2 commands to bind DataStage packages and grant access to public*

```
connect to DWHD911 user xxxx using yyyy
bind db2zeZLN.bnd datetime iso blocking all
grant execute on package DSKXS1.DB2ZEZLN to public
connect reset
```

## 9.5.8 Troubleshooting configuration problems

You can use the **ddtestlib** tool in Information Server to check if client libraries can be loaded properly. This is particularly useful in a 64-bit environment to make sure that you loaded the right libraries. As an example, you can issue the following command to check if the current user (with the proper LD\_LIBRARY\_PATH set) can load the CLI library for WebSphere Classic Federation:

```
/opt/IBM/InformationServer/Server/DSEngine/branded_odbc/bin/ddtestlib
libcacsqcli.so
```

Example 9-20 shows the type of output that you receive.

*Example 9-20 Output of ddtestlib*

```
Load of libcacsqcli.so successful, qehandle is 0x80002050
Unable to obtain version information. DataDirect ODBC drivers prior to version 4.2 SP3
have no version information available via this method.
```

You can use the **dssh** tool in the Information Server to check if a connection through the ODBC drivers can be established. Go to a project directory, which is /opt/IBM/InformationServer/Server/Projects/DWHzRedbook in our case, and run the **dssh** tool interactively as shown in Example 9-21 on page 251. The query that is used in the example does not return any results but indicates an error message if the connection cannot be established.

*Example 9-21 Using dssh to verify the ODBC connection to the Classic Federation Server*

```
lnxdwh2:/opt/IBM/InformationServer/Server/Projects/DWHzRedbook #  
/opt/IBM/InformationServer/Server/DSEngine/bin/dssh  
  
DataStage Command Language 8.0 Licensed Materials - Property of IBM  
(c) Copyright IBM Corp. 1997, 2006 All Rights Reserved.  
DWHzRedbook logged on: Fri Apr 25 11:55:04 2008  
  
>DS_CONNECT REDDWHZ  
Enter username for connecting to 'REDDWHZ' DBMS [root]: FNEUMAN  
Enter password for FNEUMAN:  
CAC00105I LOG V9.1 00002007: STARTED  
  
REDDWHZ> select count(*) from cac.part_seq;  
:-1  
REDDWHZ>
```

## 9.5.9 Granting user access and creating a new DataStage project

Information Server uses projects to group definitions, jobs, and data that belongs together. To create a new project named DWHzRedbook:

1. Launch the Information Server Web console through the Web interface, and log in using your suite administrator account. In the Information Server window (Figure 9-46), click the **Administrator** tab. In the left navigation pane, select **Users and Groups** → **Users** to open the user to whom you want to grant the right to access and use DataStage.

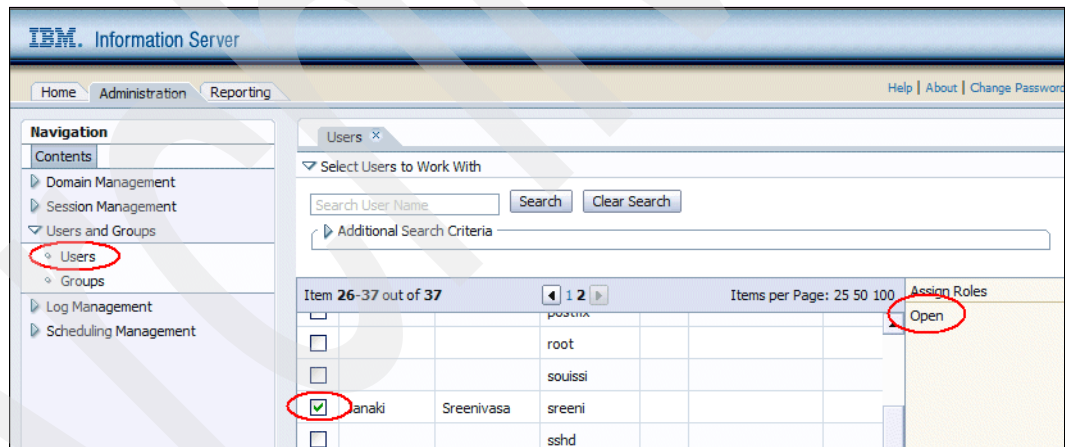


Figure 9-46 Granting use of DataStage

- Under Roles, select **Suite User** and **DataStage and QualityStage User** (Figure 9-47). This setting provides access to InfoSphere DataStage and QualityStage. Additionally, this role is used to filter the lists of users and groups that are shown in the InfoSphere DataStage Administrator client. If an IBM Information Server user does not have this role, the user cannot access any of the InfoSphere DataStage or QualityStage product modules, even if the user has been assigned InfoSphere DataStage or InfoSphere QualityStage project roles.

Click **Save and Close**.

The screenshot shows the IBM Information Server Administration console. The 'Users' tab is active, and the 'Select Users to Work With' section is open. The 'Roles' section on the right lists the following roles:

Role	Inherited
<input type="checkbox"/> Suite	
<input type="checkbox"/> Suite Administrator	
<input checked="" type="checkbox"/> Suite User	
Suite Component	
<input type="checkbox"/> Business Glossary Administrator	
<input type="checkbox"/> Business Glossary Author	
<input type="checkbox"/> Business Glossary User	
<input type="checkbox"/> DataStage and QualityStage Administrator	
<input checked="" type="checkbox"/> DataStage and QualityStage User	

Figure 9-47 Assigning roles for DataStage and QualityStage

**Note:** Depending on your installation, when you access DataStage through a client, you must ensure that you can access the UV\_USERS file. This file typically belongs to the dstage group. We add all user IDs that need access to DataStage to the dstage group by editing /etc/group file on our Linux on System z server.

- Using the WebSphere Server DataStage and QualityStage Administrator on your client installation, enter host name and port of your Information Server server installation (Figure 9-48) and log in as an administrator to attach to DataStage.

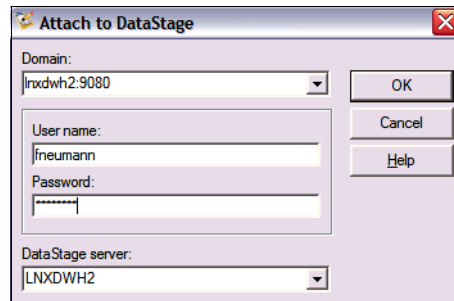


Figure 9-48 Logging in to DataStage

- In the WebSphere DataStage Administration window (Figure 9-49), in the list of existing projects, click **Add** to add a new project. In the Add Project window (inset in Figure 9-49), name it DWHzRedbook and click **OK**. When you return to the list of projects, click **Close**.

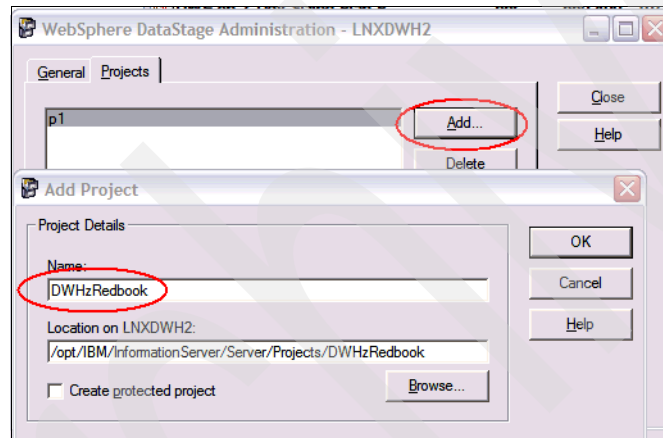


Figure 9-49 Adding a new project

5. For the new project, select **Properties**.
6. In the Project Properties window (Figure 9-50):
  - a. Click the **Permissions** tab.
  - b. Click **Add User of Group** to select the users that should be allowed to access the project. Only the users who you defined as DataStage and QualityStage users can be added to the project access list.
  - c. To allow client access, under User Role, select **DataStage and QualityStage Super Operator**.
  - d. Click **OK**.

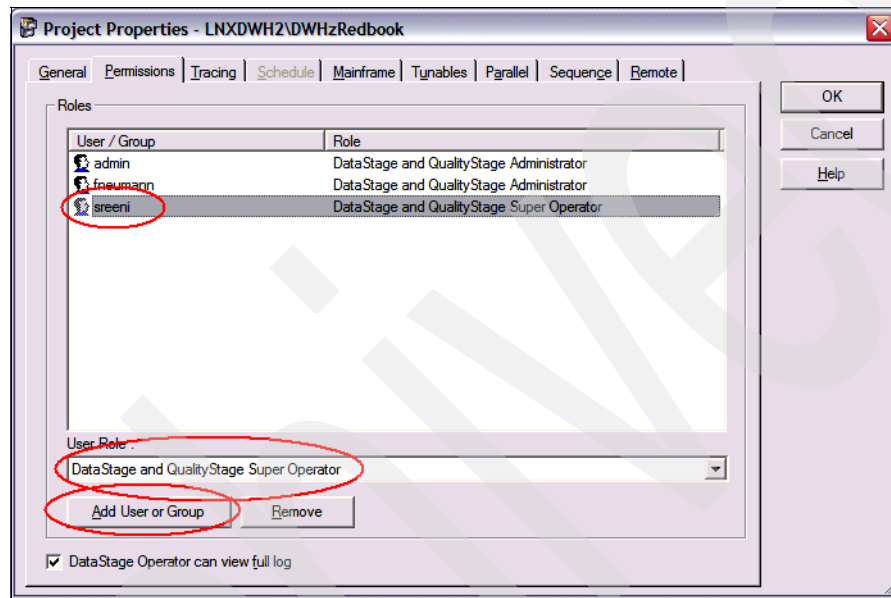


Figure 9-50 Adding a user of a group

Now both the new project and the users are setup to use the DataStage and QualityStage designer to model and save ETL jobs.

### 9.5.10 Defining multiple nodes for parallel execution

If you run a multiprocessor system for your Information Server engine, you can leverage the parallel execution of ETL jobs in DataStage.

The Linux on System z that we are using has two associated Integrated Facilities for Linux (IFLs). Therefore we define an additional DataStage configuration with two nodes:

1. Start the DataStage and QualityStage Designer on Windows and log into any of your projects.

2. From the menu, select **Tools** → **Configurations** and then select the **default** configuration. With a typical installation, the content looks as shown in Example 9-22 and defines a single node for execution of the stages in the Information Server engine.

*Example 9-22 Defining an additional node*

---

```
{
  node "node1"
  {
    fastname "lnxdwh2"
    pools ""
    resource disk "/opt/IBM/InformationServer/Server/Datasets" {pools ""}
    resource scratchdisk "/opt/IBM/InformationServer/Server/Scratch" {pools ""}
  }
}
```

---

3. To create an additional node, edit the text and add an additional node definition as outlined in Example 9-23.

*Example 9-23 Configuration with an additional node*

---

```
{
  node "node1"
  {
    fastname "lnxdwh2"
    pools ""
    resource disk "/opt/IBM/InformationServer/Server/Datasets" {pools ""}
    resource scratchdisk "/opt/IBM/InformationServer/Server/Scratch" {pools ""}
  }
  node "node2"
  {
    fastname "lnxdwh2"
    pools ""
    resource disk "/opt/IBM/InformationServer/Server/Datasets" {pools ""}
    resource scratchdisk "/opt/IBM/InformationServer/Server/Scratch" {pools ""}
  }
}
```

---

4. Click **Save** → **Save Configuration As**.
5. Type the name of your configuration for two nodes, for example **TwoNodes**. Click **Close**.

You now have two configurations defined, one named *default* and the other defined *TwoNodes*. Both are available in all your projects. You have to take additional steps to make them selectable for individual jobs and runs.

**Note:** If you change the default configuration, this change applies to all jobs and projects.

To give you a choice of which configuration to use for a job run, define an additional job parameter for the configuration environment variable:

1. With your DataStage job open in DataStage and QualityStage Designer, click **Edit** → **Job Properties** → **Parameters** and then click **Add Environment Variable**.
2. In the list of available environment variables (Figure 9-51 on page 256), select **Configuration file (\$APT\_CONFIG\_FILE)**.

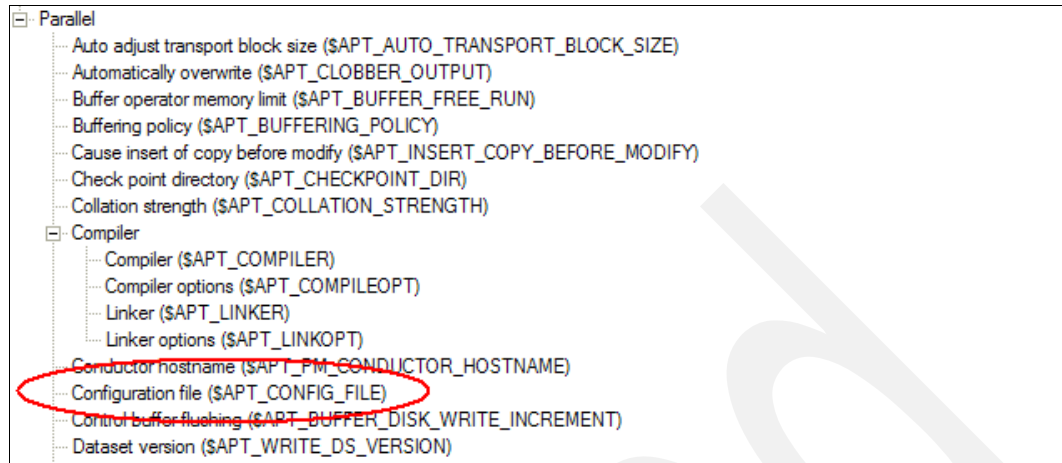


Figure 9-51 Selecting the configuration file environment variable for the job properties

3. In the job properties panel (Figure 9-52), click **OK**.

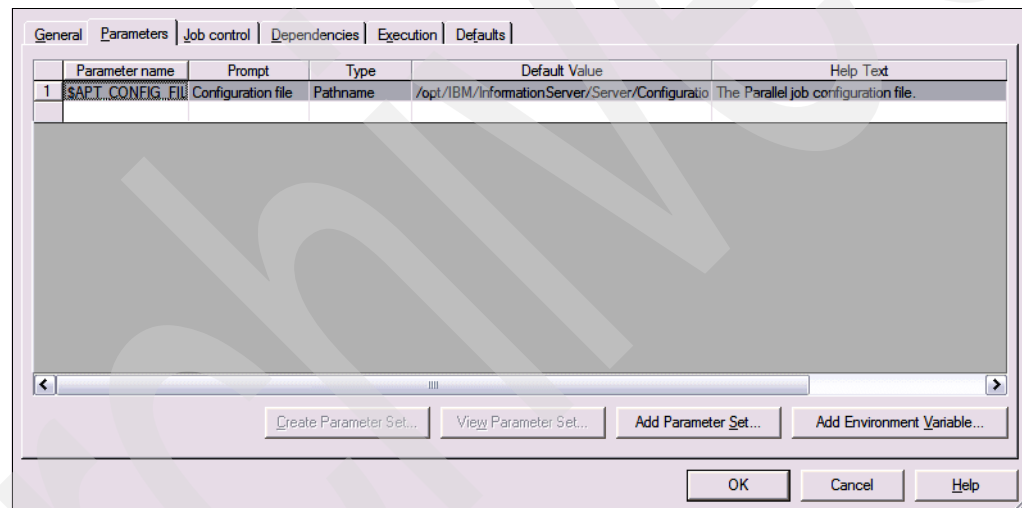


Figure 9-52 Job properties with the configuration file as a parameter

4. If you run your job again, specify the configuration to use in the window in Figure 9-53. Click the dotted square button on the right under Value to select **TwoNodes** from the available configurations. Then click **Run**.

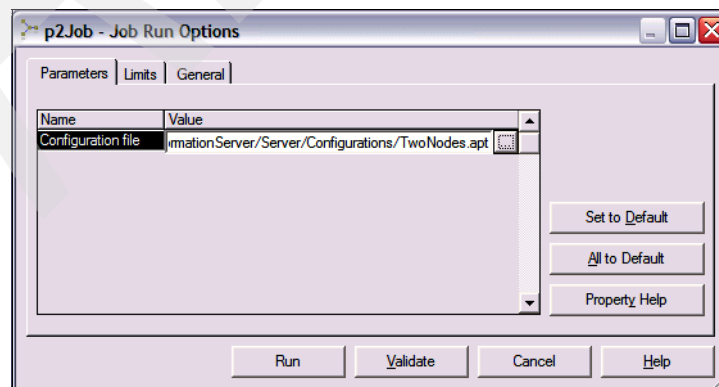


Figure 9-53 Selecting a different configuration for running a DataStage job



## Full load using DataStage

In this chapter, we describe the implementation for extract, transform, and load (ETL) processing from the operational (online transaction processing (OLTP)) data of our sample scenario into the data warehouse. We explain how, by the full load, all the initial data that existed in the OLTP environment was moved to the operational data store (ODS) and the dimensional data store (DDS) by using InfoSphere DataStage.

In Chapter 9, “Setting up ETL components for a data warehouse” on page 201, we explain the installation of the Information Server. This installation helps in using InfoSphere DataStage, which is central in the loading of the operational data in the ODS and DDS. In Chapter 11, “Incremental update with DataStage” on page 281, we expand on the jobs that read data from DB2 on z/OS tables to show a sample job that reads message queues that have been populated from the OLTP database log. This enables real-time business intelligence (BI).

In this chapter, we present an overview of the DataStage jobs that were developed to load all the initial data to satisfy the business scenario in Chapter 5, “The business scenario and data models” on page 71. We also explain some basic steps in modeling DataStage jobs for running, debugging, and improving performance as needed while maintaining data quality. More focus is employed on some of the stages needed for particular BI functions, such as the *surrogate key stage* (in this chapter) and the *slowly changing dimension stage* (in Chapter 11, “Incremental update with DataStage” on page 281). An additional objective of this chapter is to show how an ETL tool, such as DataStage, interacts well with DB2 on z/OS and enables strategic and operational BI.

This chapter includes the following topics:

- ▶ 10.1, “ETL data in our scenario” on page 258
- ▶ 10.2, “Loading overview” on page 260
- ▶ 10.3, “Load jobs for populating ODS from OLTP” on page 262
- ▶ 10.4, “Load jobs for populating a DDS from an ODS” on page 266
- ▶ 10.5, “Accessing WebSphere Classic Federation in DataStage jobs” on page 271
- ▶ 10.6, “Running and monitoring an ETL job in DataStage Director” on page 273
- ▶ 10.7, “Debugging load jobs: A brief look” on page 274
- ▶ 10.8, “Performance considerations” on page 276
- ▶ 10.9, “Naming standards” on page 279
- ▶ 10.10, “Data quality implementation during full load” on page 279

## 10.1 ETL data in our scenario

Our OLTP data sources contain information about orders from customers. Orders can contain line items of parts. These parts come from suppliers. For each line item of an order, we keep track of the part and the supplier along with other information such as quantity ordered. Each order also contains shipment information. These orders can be placed at the branches or on the Web. The transaction data of orders, shipment, and line items coming from the branches goes into the OLTP\_B source schema tables. The data that originates from the Web goes into the OLTP\_W source schema.

You can read more about the fictitious distribution company created for our business scenario in 5.1, “Background information” on page 72. The transactional data model that pertains to this scenario is discussed in 5.4.1, “The OLTP database model” on page 78. The data model for the ODS and the DDS is explained in 5.5, “The operational and dimensional data model” on page 85.

We go through the required table structures across the different environments for background, before we go in more detail about the load jobs that were developed to move data into these environments. The OLTP systems have the following main tables:

- ▶ Parts, suppliers, nation, region
- ▶ Order, lineitem, and shipment
- ▶ Customer

Parts, supplier, nation, and region information for the source OLTP system and for the target operational data store are basically the same. They are considered to be either stored in identical data structures or maintained in a data source that can be accessed from both OLTP and data warehouse systems. For the purpose of this book, we implement both options. That is, parts and supplier data is either stored in database tables or maintained in flat files and accessed through WebSphere Classic Federation.

For demonstration purposes, the part, supplier, nation, and region information is assumed to be static. It does not impose any sophisticated transformation or load requirements. To manage loading of the data with the same tool and the same technique, we use a DataStage ETL job.

Customer data stored in the source OLTP system slightly differs from the required target format in the operational data store (Figure 10-1). The column names are different. In addition, the OLTP system only provides a country name, while the target operational data store expects a key that points to a nation lookup table. Therefore, the ETL job needs to look up data from other tables and transform the column names accordingly.

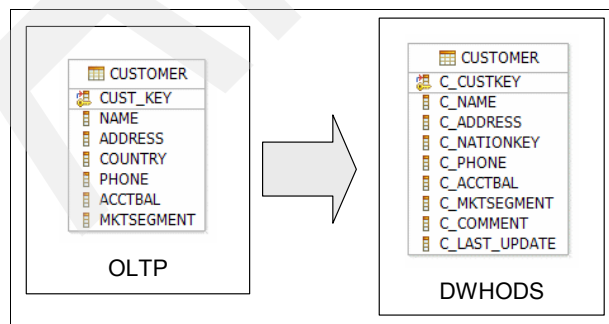


Figure 10-1 Transforming customer data from OLTP to ODS

Loading the *order* and *line-item* data in our operational data store requires more steps (Figure 10-2). When comparing DWHODS with our two OLTP database schemas (OLTP\_W and OLTP\_B), our ETL job must complete the following tasks:

- Load and merge data from both OLTP data sources.
- Adjust column names.
- Transform data from the *shipment table* and assign it to the *Orders* and *Lineitem* tables in the target ODS accordingly.

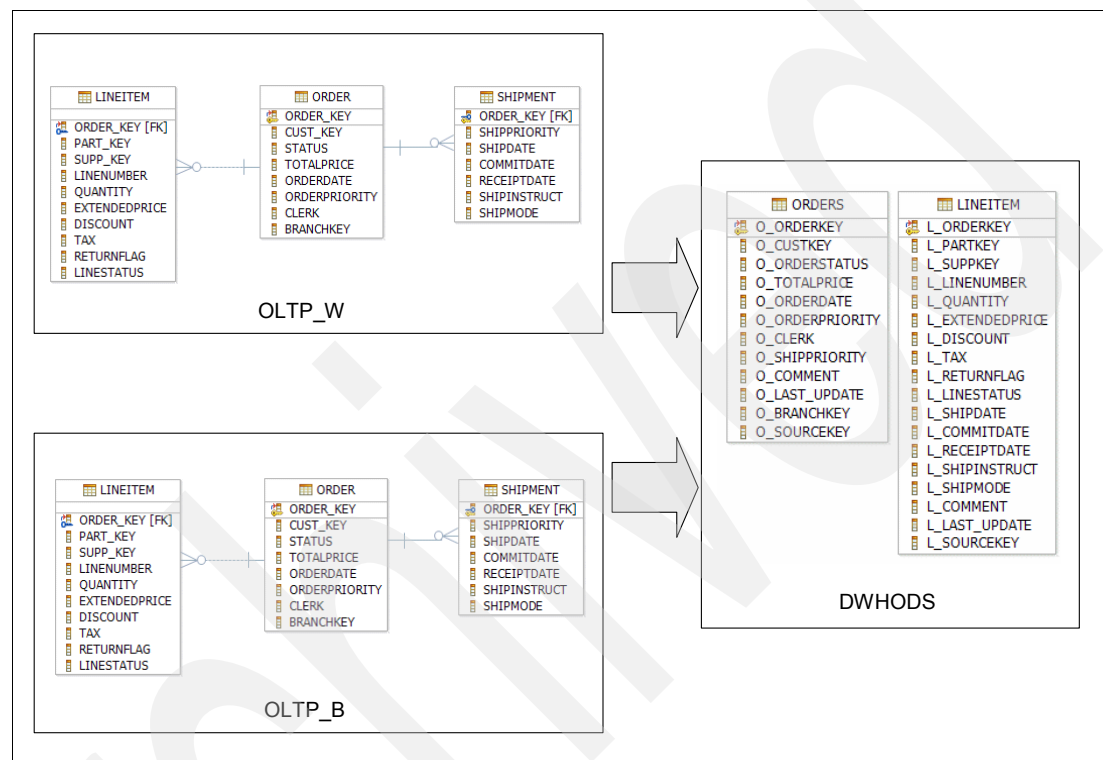


Figure 10-2 Transforming order and line-item data from OLTP to ODS

Conceptually, we can load our target DDS directly from the OLTP system and load the operational and DDS in one run. For our implementation, however, we derive the DDS data from the previously populated ODS (Figure 10-3 on page 260). The ETL jobs have the following predominant requirements:

- Building up the order transaction star schema of the DDS
- Maintaining history information for slowly changing dimension data
- Adding time stamps for changes
- Computing and resolving values that are required for reporting, for example, the quantity of part returns
- Assigning surrogate keys

To store the historical data, we must define and create new primary keys for some tables and load the star schema in two phases. First we populate the dimension tables, and then we populate the fact table with foreign keys that point to the dimension table records.

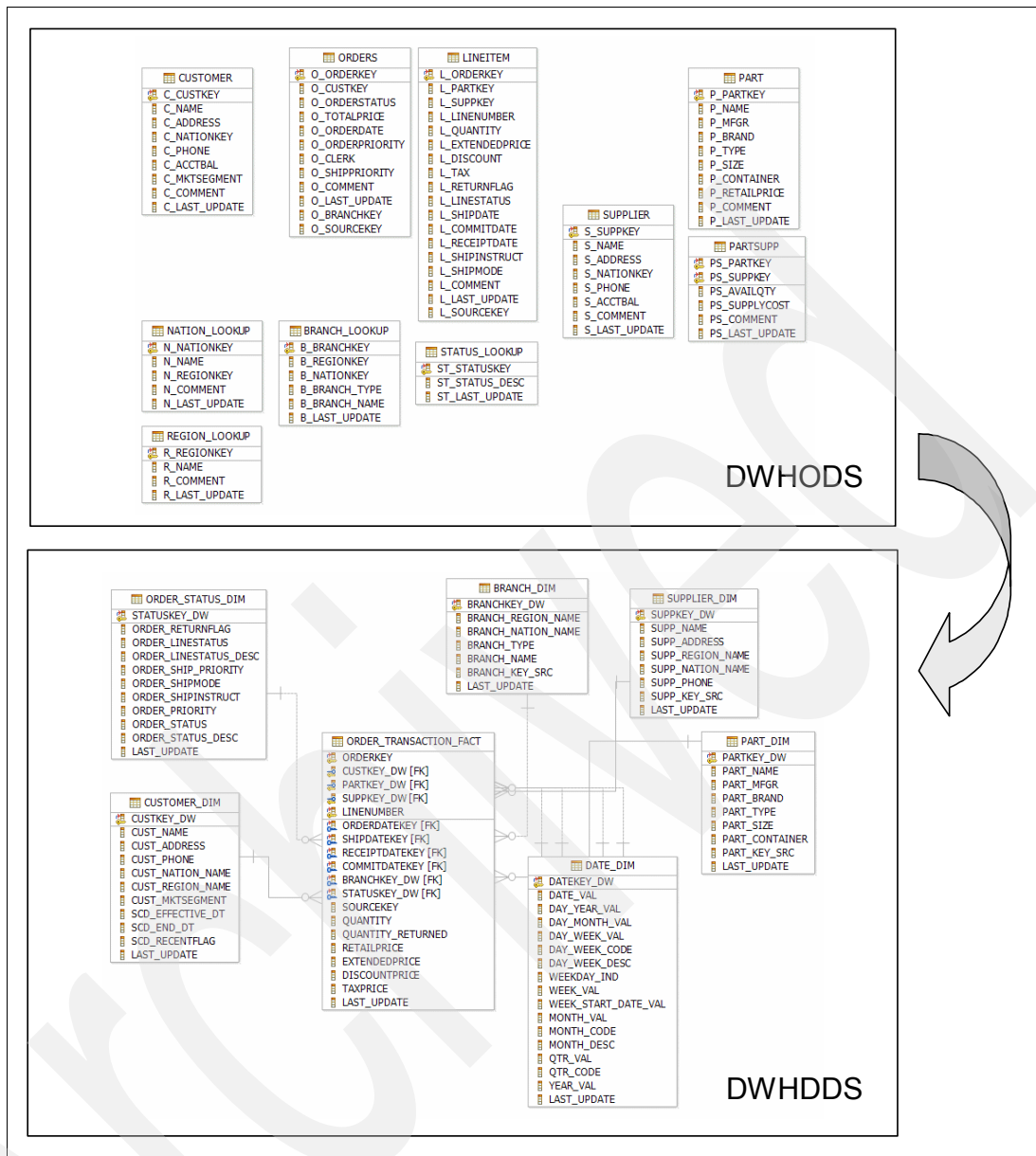


Figure 10-3 Transforming data from the ODS to the DDS

## 10.2 Loading overview

After WebSphere Information Server on Linux on System z is installed, which we explain in Chapter 9, “Setting up ETL components for a data warehouse” on page 201, you use the WebSphere DataStage designer tool most of the time to develop your jobs. As soon as you open the designer, you have a choice of jobs to design. For the purpose of loading all the data warehouse tables, we select parallel jobs. For the difference between parallel and server jobs, see 10.8.2, “Parallel jobs versus server jobs” on page 277.

In developing a DataStage parallel job, you have to select stages and link the m by using *DataStage links*. In DataStage, data conceptually moves along the links. Stages represent a particular process step. They can be a temporary place to hold this set of data and act upon it as needed, or they can be a source and target for the data that is moving via the links.

In the following sections, we describe briefly the context of a full load. Then in 10.3, “Load jobs for populating ODS from OLTP” on page 262, we describe how *DB2z stage* and *BatchPipes* are useful for moving the data. Following this, we explain some of the jobs that were developed for moving data in our scenario.

### 10.2.1 Full load description

The BI solution for the scenario in this book focuses on the movement of data from the operation transaction databases to the ODS and the DDS. The requirement of developing the jobs to do this is to provide you with a quick and reasonable example of the relative quickness in developing DataStage jobs to move the data from a transaction system to a data warehouse and BI solution. Since the focus of the book is to provide operational and strategic BI, the full load shows the entire ODS loaded with a current snapshot of data, and from there, moved to a data mart or DDS environment for querying and reporting.

### 10.2.2 Incremental load description

The incremental load shows a low latency movement of data from the operational system to the ODS to demonstrate that a consolidated view can be kept current by the movement of operational system updates into the data warehouse environment. Business activities in the operational database are written to the database log in general situations where logging is employed. This log is read by tools, such as Q replication and Event Publisher, and is written to a queue. The messages can be written in different formats such as delimited or XML formats. DataStage jobs can be set up to read the queues and update the tables in the data warehouse environment. This way users who are heavy consumers of this current data can cement personal relationships with customers and engage in cross-sell opportunities.

In 3.2.2, “Workload management” on page 35, we provide a glimpse of one of the DB2 for z/OS capabilities in handling a read of the tables while they are updated with minimum inconvenience by managing these mixed query loads relatively seamlessly. In Chapter 8, “Q replication and event publishing” on page 159, we explain the Q replication and Event Publisher tools and walk you through the population of a queue. Then in Chapter 11, “Incremental update with DataStage” on page 281, we show how messages in the queue can be read by a job in DataStage and used in the update of information.

### 10.2.3 Lookup tables

*Lookup stages* are used to look up the descriptions of coded values. These are used in the population of dimension tables in the DDS where descriptions needed to be populated. This means that a number of lookups are performed to the reference or lookup tables such as Nation and Region.

The Lookup stage is also handy for the population of any key values, such as surrogate keys, used in the dimensions that need to be looked up and placed in the Fact table.

The focus on explaining these jobs is not to teach DataStage comprehensively but to show some of the jobs that are created in order to load the business scenario data models.

## 10.3 Load jobs for populating ODS from OLTP

*DB2z stages* are DataStage process steps that facilitate in reading from a DB2 for z/OS database or writing data to a DB2 for z/OS database. We provide the following short descriptions, which include examples of properties for the stage.

### 10.3.1 DB2z stage: Reading DB2 data

In the DB2z stage to read DB2 data, you can specify the table name and database information from which to read data. You can specify the table (or a subset of the columns for a table) that you need to be read or extracted. The properties of the DB2z stage (Figure 10-4) require the name of the database (more specifically, the subsystem that is cataloged on Linux on System z) and the user credentials for accessing the database.

The column definition for the query result can be specified based on metadata of the underlying DB2 for z/OS tables through an import wizard, using Open Database Connectivity (ODBC).

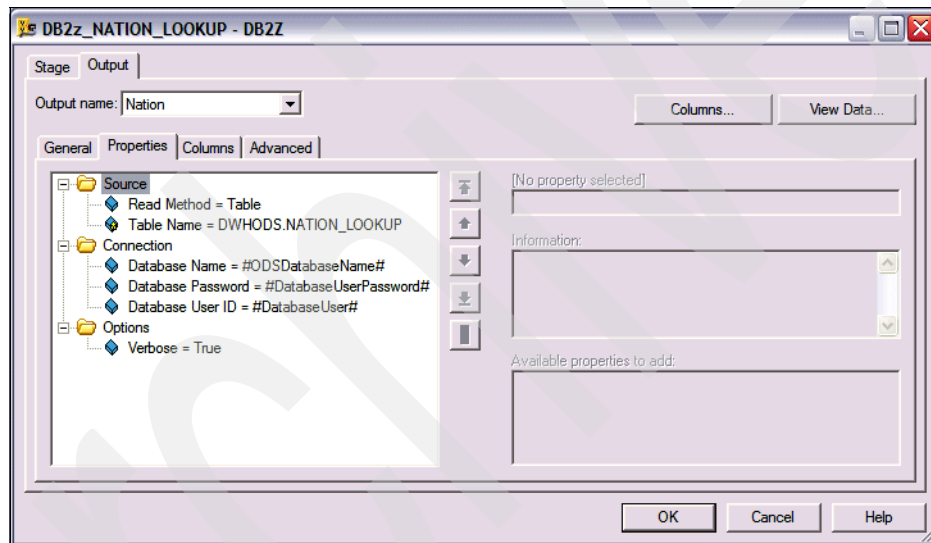


Figure 10-4 DB2z stage properties for reading data from DB2 for z/OS

### 10.3.2 DB2z stage: Writing DB2 data

If your target is a DB2 z/OS table, then you have a DB2z stage as the last target stage in your job. In this stage, you can choose options such as whether you want to append to data that already exists in the table or truncate existing table data and load replace into it.

The DB2z stage is optimized for loading data into DB2 for z/OS and uses the load utility for best performance. There are other options to insert data into a DB2 for z/OS database, such as the ODBC stage. However, from a performance point of view, the DB2z stage is the preferred choice. Figure 10-5 on page 263 shows the properties of the DB2 stage for writing DB2 data.

In our scenario the OLTP tables and the ODS and DDS tables are all setup in DB2 z/OS database. The OLTP tables are in one subsystem, and the ODS and DDS tables are setup on another subsystem. In 4.2, "The business intelligence architecture with System z" on

page 61, we provides information about Workload Manager, which you can set up to provide the proper priorities for the queries to access the two different subsystems.

You must use care when choosing some options in the target DB2z stage. The DB2z stage technically uses FTP to send data from Linux on System z to z/OS. The data submitted by FTP can either go into a data set on z/OS or immediately serve as input for a batch pipe.

If you are not using BatchPipes, place the amount of anticipated row count in the Row Count Estimate property of the DB2z stage. If this number is significantly less, the stage may not allocate a data set with enough space for your transform stages that you have in your job.

If your job stops, make sure that some of the created data sets are not lingering on the z/OS server.

A load card is created when you write data to a target with this stage. Make sure that you terminate this utility if it stops after an unsuccessful load. The default name for this is DB2ZLOAD, but you can change it and use a different name by the Utility ID option available.

For the Overwrite option of the Transfer property, choose **True** if you are running the job again. Otherwise the job execution might try to create the same data sets again. When it finds that they already exist on z/OS, it gives you a message.

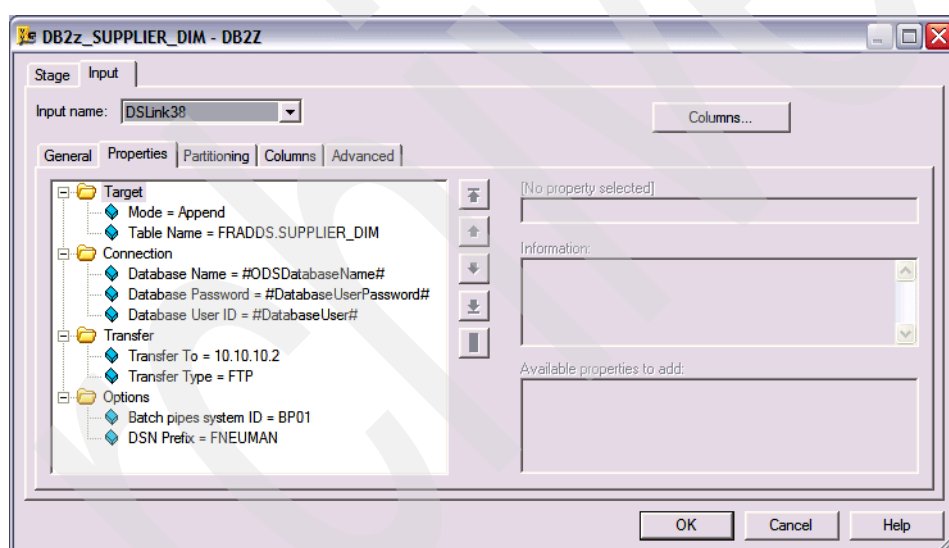


Figure 10-5 DB2z stage properties for writing data to DB2 for z/OS

Ensure that you apply the list of required APARs for BatchPipes support. You can find these APARs in Table 9-1 on page 209. If these APARs are not installed when you input your BatchPipes system ID, the job may not complete successfully. You might also receive error messages when you write to decimal data type fields in the target. Sometimes only a few rows can be pushed into the target table and the job may not stop. Be sure to verify the number of records in the target table after the load run. All of these records should be rectified if you install all the required APARs.

### 10.3.3 Parallel jobs and BatchPipes

Using BatchPipes with parallel jobs in DataStage is particularly beneficial because it allows continuous loading of data through the DB2 LOAD utility while data is being processed in the DataStage job. Since you do not create intermediate data sets, proper data set allocation is not required because traditionally a Job1, for example, writes data to an I/O device. When all

the records are written and the data set is closed, then a job, for example Job2, starts. Job2 reads the data from the device, which does not occur in BatchPipes. In BatchPipes, for example, Job1, which is a writer that writes data to the processor storage buffer, and Job2, which is a reader from that buffer, run concurrently. Job2 can obtain data from the processor storage buffer as soon as Job1 writes the first block. Output from Job1 becomes immediately available as input to Job2. You can think of the data as “flowing” from Job1 to Job2.

The processor storage buffer is called, in BatchPipe terms, a *pipe*, through which data flows, always in the same direction, from a writer job to a reader job. The writer →pipe →reader flow is called a *pipeline*.

In the traditional way, Job1 writes data to an I/O device, such as DASD or a tape. Then, when all the records are written and the data set is closed, Job2 starts. After Job2 finishes processing all the records, the data set is no longer required and can be deleted. In BatchPipes, as illustrated in Figure 10-6, which is from *IBM BatchPipes OS/390 V2R1 BatchPipeWorks User Guide*, SA22-7457, you can see that external storage devices are absent. Instead they are replaced by a processor storage buffer. This parallelism provides great elapsed time savings and helps you avoid worrying about setting a good row count estimate to ensure that enough space on DASD is allocated.

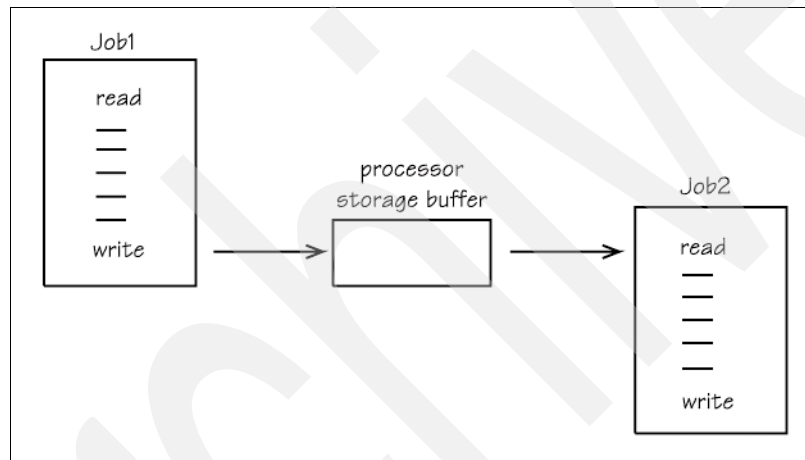


Figure 10-6 BatchPipe utility

### 10.3.4 Sample load jobs from OLTP to ODS

Let us start with a simple job to populate an ODS PART table from the OLTP PART table. In DataStage, the job when developed looks as shown in Figure 10-7.

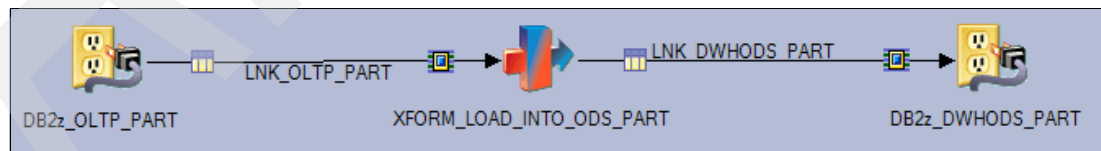


Figure 10-7 Sample simple job from OLTP to ODS



A successful job run looks as shown in Figure 10-8.

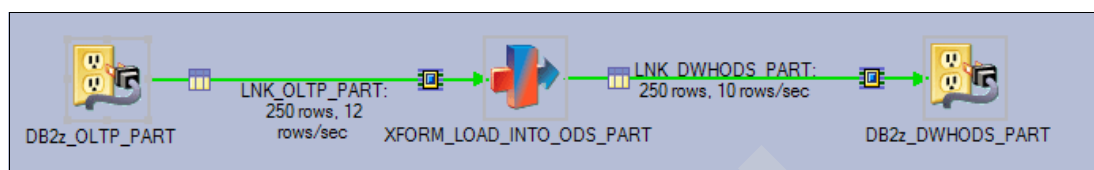


Figure 10-8 Sample simple job from OLTP to ODS successfully completed

You can use a DB2z stage if the source or target table is on DB2 for z/OS. You use a transformer stage to perform routine transformations that are needed and to map them from the source to the target. However, other stages can be of more interest. Although this book does not explain all possible stages while using DataStage, we discuss some of the stages that we used to load the ODS and DDS tables for our business scenario. We provide a quick primer for those who need to quickly develop DataStage jobs to load data. Refer to the *WebSphere DataStage Parallel Job Advanced Developer Guide*, LC18-9892, which explains all the stages in great detail.

Let us look at a job that uses some lookups. For the DWHODS.CUSTOMER table, we take the Country value that comes from the OLTP.CUSTOMER table and look up the nation key from the DWHODS.NATION\_LOOKUP table. See Figure 10-9.

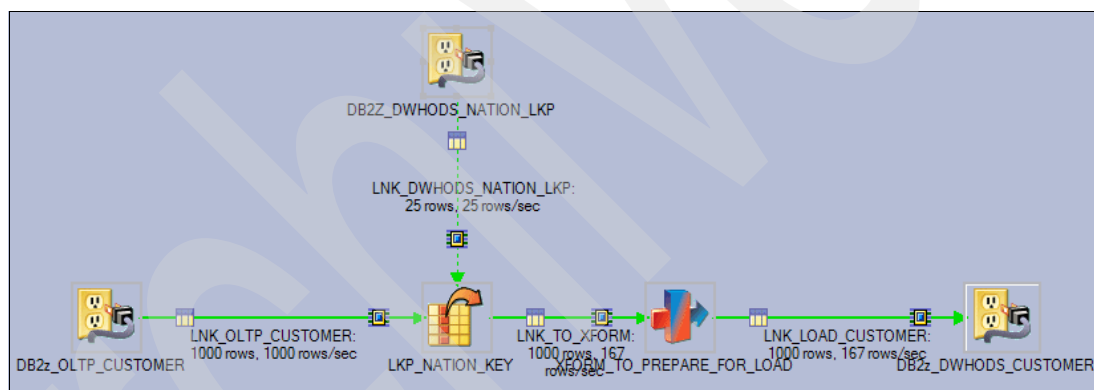


Figure 10-9 Population of ODS Customer table from the OLTP Customer table

Let us graduate to a slightly more interesting DataStage job. In the OLTP environment, there is an OLTP\_B.ORDER table and an OLTP\_W.ORDER table, because orders are coming from either the branch stores or the Web. In populating the target DWHODS.ORDERS table, we also need information from the OLTP\_B.SHIPMENT and OLTP\_W.SHIPMENT tables.

Here is the job that provides the load to the DWHODS.ORDERS table. You have to order by the key in the join stage before the data reaches the join stage. You can use a sort stage or the ORDER BY clause option when reading from the source Order and Shipment tables. You can also choose a Sort Funnel to keep the data in the required order.

Figure 10-10 shows that we received 6,000 order rows each from the Web and the branch, and we loaded the 12,000 rows into the target ODS Orders table. We also obtained additional information from an equal number of shipment rows.

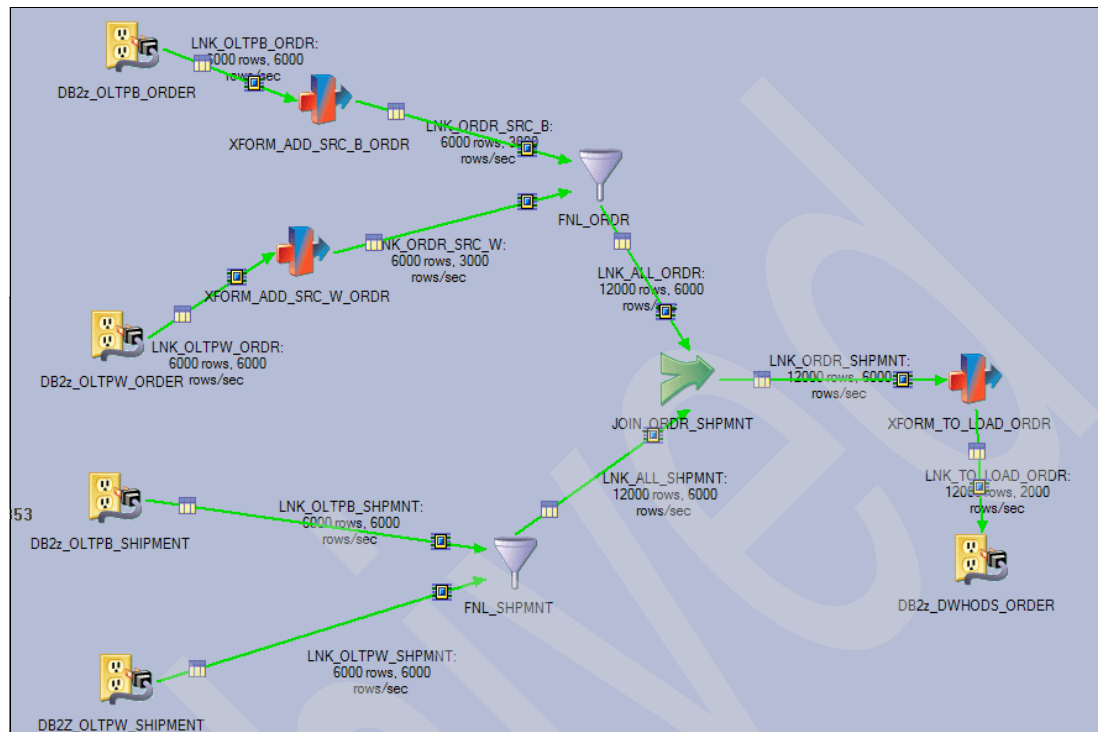


Figure 10-10 Load of DWHODS.ORDER table from the necessary OLTP tables

The job to load the DWHODS.LINEITEM is similar to the load of the Orders table accomplished previously. We also need to move the data from the ODS to the DDS. We go through the jobs and show the DDS load, which indicates a variation from the ODS loads. For example, we show a load from a flat file for the load of the Date dimension table, usage of the surrogate key stage, and so on.

## 10.4 Load jobs for populating a DDS from an ODS

Several jobs are created for the load of the DDS tables from the ODS data as part of the initial full load. We focus on jobs that deal with slightly different stages than those that we encountered in the load from OLTP to ODS. Let us initially look at the load of the Date dimension table.

### 10.4.1 Load of the Date dimension table

The contents that are loaded into the date dimension are at a day granularity for the calendar years 2007 and 2008. Because there are 731 days between these years, the job shown in Figure 10-11 on page 267 reflects the load of the 731 rows.

**Note:** One extra row was inserted to account for a Null value in the Ship, Receipt, or Commit dates.

The job shows the read from a sequential file. It also shows that, if needed, you can use more than one transformer stage in immediate succession. Especially if the number of rows is relatively small, such as for the date dimension, you can choose to keep your stages clean and use more than one transformer stage as needed.

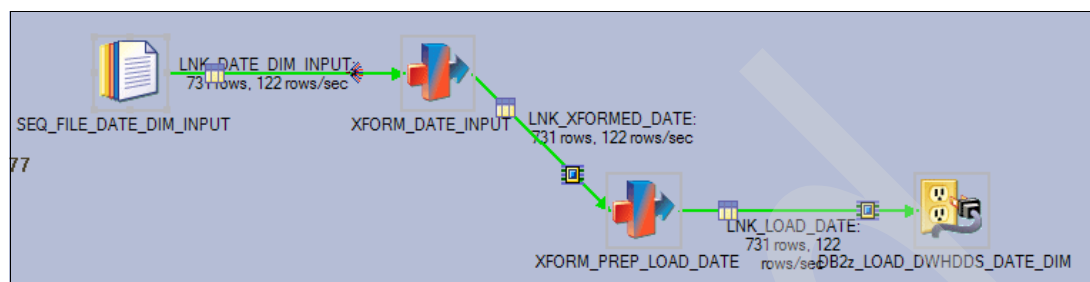


Figure 10-11 Date dimension load

## 10.4.2 Surrogate key stage utilization

Figure 10-12 shows the properties of the surrogate key generator stage. Figure 10-13 illustrates the use of this stage in which the population of the Order Status dimension with its primary key column is a surrogate key. According to requirements, the Order Status dimension is unique based on the concatenated columns of L\_LINESTATUS, O\_ORDERSTATUS, L\_RETURNFLAG, O\_SHIPRIORITY, O\_ORDERPRIORITY, L\_SHIPINSTRUCT, and L\_SHIPMODE. Based on a change of all of these natural columns together, we created a surrogate key.

Other stages, such as the slowly changing dimension stage, are valuable in the movement of data to the data mart. The surrogate key stage is discussed in Chapter 11, “Incremental update with DataStage” on page 281.

Let us quickly go through the Order Status dimension population job and focus on the surrogate key generator stage. Our target is to populate the DWHDDS.ORDER\_STATUS\_DIM table. Our input is the DWHODS.LINEITEM table with additional information of order status, order priority, and ship priority coming from DWHODS.ORDERS based on a match of the orderkey and DWHODS.STATUS\_LOOKUP for the Order status description and Line item status description. We sort in the ascending order of the concatenated columns mentioned previously, because before we remove the duplicates by using the remove duplicates stage, we must apply the Sort stage.

When the load file is ready, except for the surrogate key, we pass it through the surrogate key stage. One way to create the surrogate keys is to create a state file and then set up the properties of the stage. You can open Microsoft Notepad, choose **Save As**, and create an empty text file. Then send the text file by FTP to the server on which you are going to run the surrogate key generator stage. You can run the surrogate key generator stage with no input or output and compile and run the job. The job will run successfully although you receive a message indicating that the file is empty. You can then place the file name in the Source Name property as shown in Figure 10-12 on page 268.

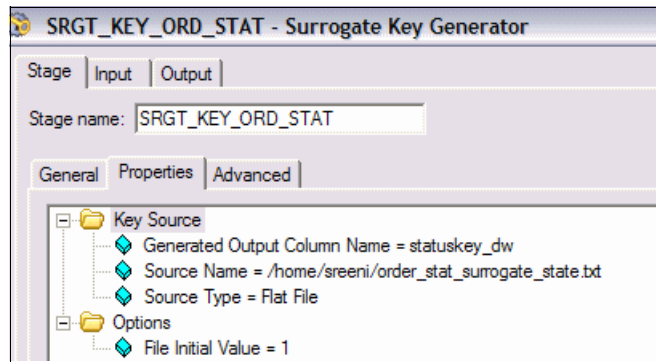


Figure 10-12 Properties of the Surrogate key generator stage

You can now run your job as shown in Figure 10-13.

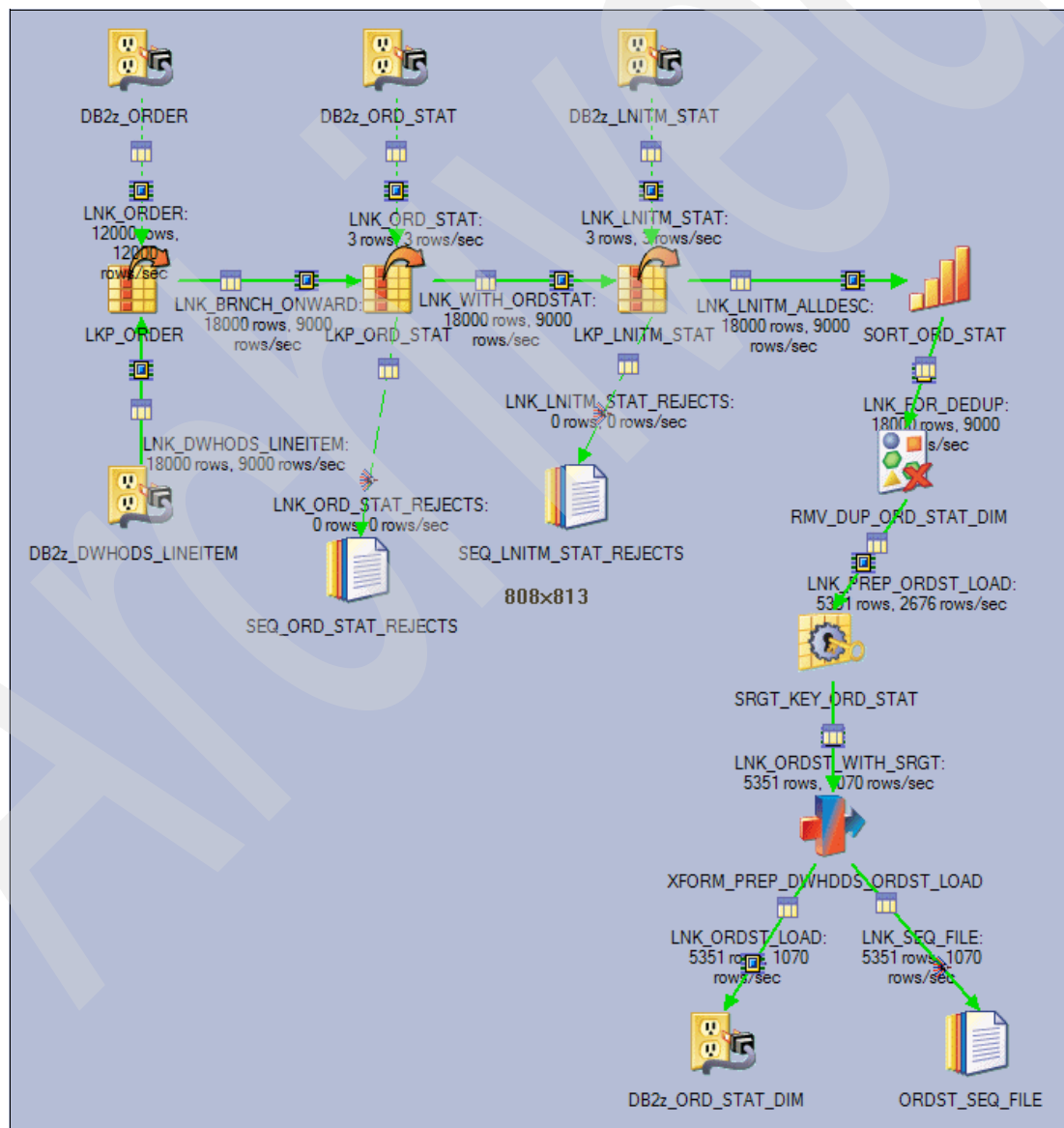


Figure 10-13 Use of the surrogate key generator stage in loading data

After the job runs, you can go to the sequential file that we created and look at the surrogate key values, which are shown in Figure 10-14.

statuskey_dw
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

Figure 10-14 Sample surrogate generator stage values

### 10.4.3 Load of the fact table

The ODS ORDERS table contains the orders that came from the branches and the Web. Similarly the ODS LINEITEM table contains the line items of the orders from both the branches and the Web. Now this order and line-item information needs to be populated in the ORDER\_TRANSACTION\_FACT table. This way the Cognos 8 BI can be used on top of it to create useful BI.

The design of the fact table in our scenario has nine keys that need to be looked up along with the retail price of a part before population of a fact table row. Line item is the granularity of this fact table. Therefore, we take all the line items, add some order information to them, make one pass at getting all the keys from the relatively small dimension tables, and push them into the fact table. We have reject files for each lookup to trap non-matching lookups. During development, you can have peek stages to provide information in the output log when something does not match in the lookups. We can place the peek stage directly on the lookup stage.

Figure 10-15 shows the job for load to the fact table.

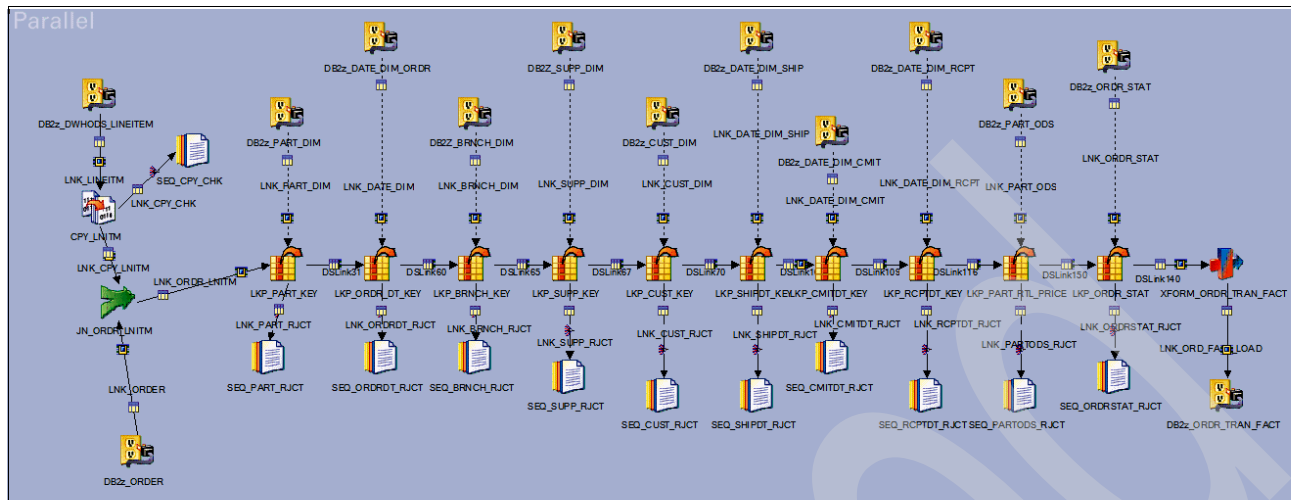


Figure 10-15 Order transaction fact table load

Upon successful completion, the job looks as shown in Figure 10-16

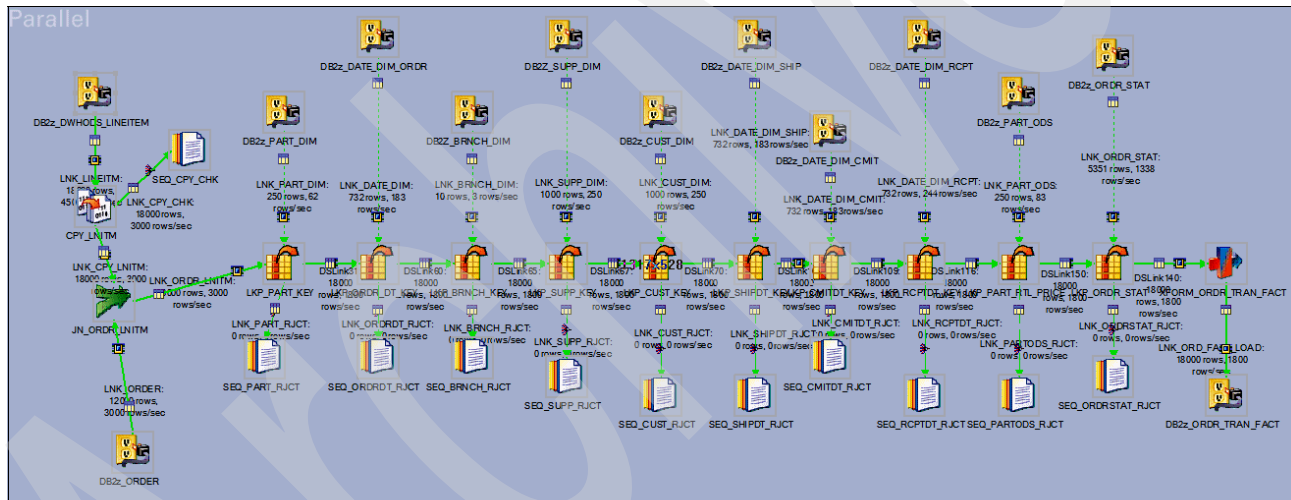


Figure 10-16 Completed Order transaction fact table load

You can verify if your reject files are empty by looking at the log as shown in Figure 10-17.

10:01:49 AM	5/5/2008	Info	SEQ_COPY_CHK,0: Export complete; 18000 records exported successfully, 0 rejected.
10:01:49 AM	5/5/2008	Info	DB2z_ORDER_TRAN_FACT,0: 1DSNU0001 126 10:01:45.74 DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = DB2ZLOAD (...)
10:01:52 AM	5/5/2008	Info	SEQ_PART_RJCT,0: Export complete; 0 records exported successfully, 0 rejected.
10:01:52 AM	5/5/2008	Info	SEQ_ORDRDT_RJCT,0: Export complete; 0 records exported successfully, 0 rejected.
10:01:52 AM	5/5/2008	Info	SEQ_BRANCH_RJCT,0: Export complete; 0 records exported successfully, 0 rejected.
10:01:52 AM	5/5/2008	Info	SEQ_SUPP_RJCT,0: Export complete; 0 records exported successfully, 0 rejected.
10:01:52 AM	5/5/2008	Info	SEQ_CUST_RJCT,0: Export complete; 0 records exported successfully, 0 rejected.
10:01:52 AM	5/5/2008	Info	SEQ_SHIPDT_RJCT,0: Export complete; 0 records exported successfully, 0 rejected.
10:01:52 AM	5/5/2008	Info	SEQ_CMIDT_RJCT,0: Export complete; 0 records exported successfully, 0 rejected.
10:01:52 AM	5/5/2008	Info	SEQ_RCPTDT_RJCT,0: Export complete; 0 records exported successfully, 0 rejected.
10:01:52 AM	5/5/2008	Info	SEQ_PARTODS_RJCT,0: Export complete; 0 records exported successfully, 0 rejected.
10:01:52 AM	5/5/2008	Info	SEQ_ORDRSTAT_RJCT,0: Export complete; 0 records exported successfully, 0 rejected.
10:01:52 AM	5/5/2008	Info	main_program: Step execution finished with status = OK.
10:01:52 AM	5/5/2008	Info	main_program: Startup time, 0:03; production run time, 0:07.
10:01:52 AM	5/5/2008	Info	Parallel job reports successful completion

Figure 10-17 Job log report of reject files during Order transaction fact table load

## 10.5 Accessing WebSphere Classic Federation in DataStage jobs

DataStage offers a dedicated stage to include data from legacy data sources in ETL jobs. With the proper setup of the WebSphere Classic Federation client, refer to the relational data that is exposed by WebSphere Classic Federation on z/OS. For more information, see 9.4.3, “Installing and configuring the WebSphere Classic Federation client on Linux on System z” on page 226.

For our scenario, we have data for parts and suppliers stored as flat files on z/OS. We make them available as relational data in table structures CAC.PART\_SEQ and CAC.SUPPLIER\_SEQ respectively.

The purpose of the ETL job shown in Figure 10-12 is to read data for suppliers and populate the supplier dimension table in our dimensional data store. The layout in the DDS assumes that the nation and regional names are being resolved. Therefore, we use two lookup stages to add these names from the respective database tables.

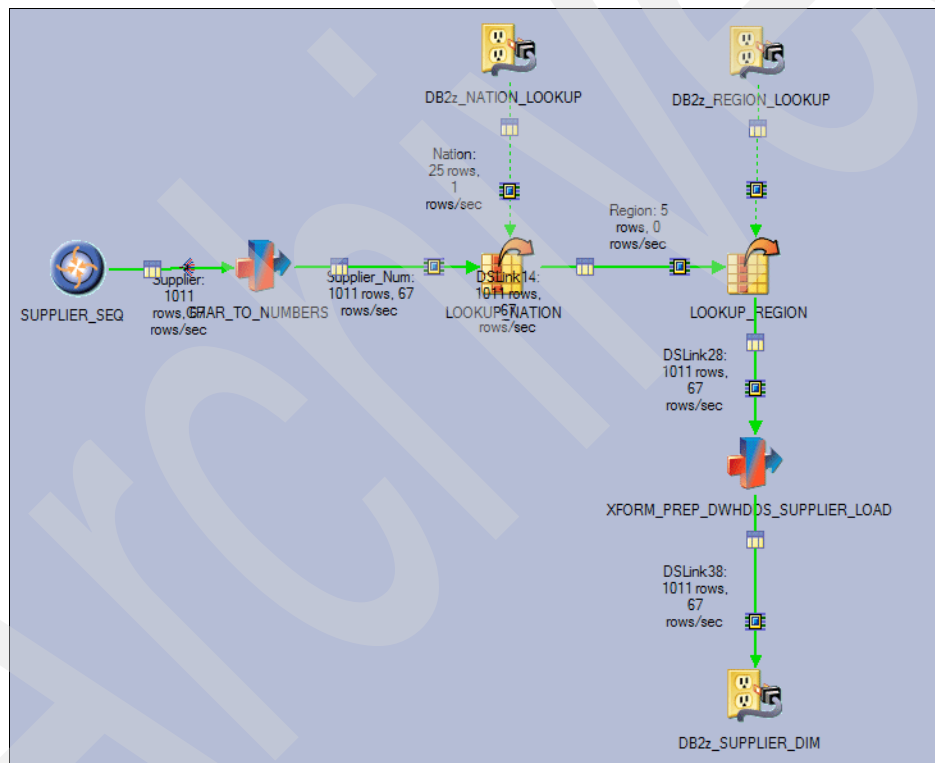


Figure 10-18 Accessing legacy data in a DataStage job

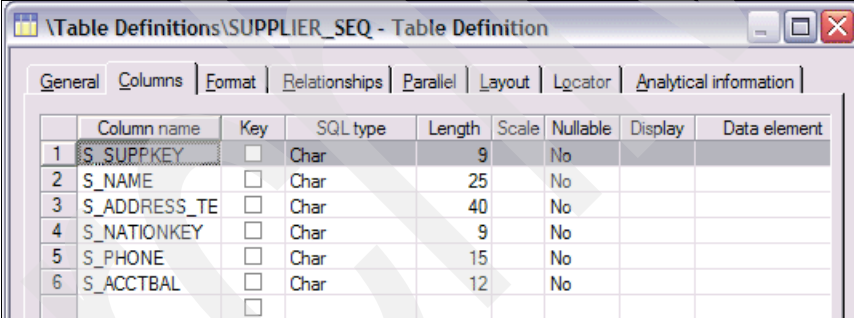


Table 10-1 shows the parameters of the WebSphere Classic Federation stage in Figure 10-18 on page 271.

Table 10-1 WebSphere Client Federation properties

Property name	Value	Comment
Source - Read Method	Table	Specifies that we want to read the entire table and not just a subset of the columns or rows.
Source - Table	CAC.SUPPLIER_SEQ	The name of the table configured in WebSphere Classic Federation on z/OS.
Connection - DataSource	REDDWHZ	The data source name configured in the cac.ini file.
Password	#DatabaseUserPassword#	If WebSphere Classic Federation is set up with security, the user password is required. We use a job parameter here.
UserId	#DatabaseUser#	The user ID that connects to WebSphere Classic Federation. We use a job parameter here.

We create the table definition for the structure of the CAC.SUPPLIER\_SEQ table manually by using the definition shown in Figure 10-19.



	Column name	Key	SQL type	Length	Scale	Nullable	Display	Data element
1	S_SUPPKEY	<input type="checkbox"/>	Char	9		No		
2	S_NAME	<input type="checkbox"/>	Char	25		No		
3	S_ADDRESS_TE	<input type="checkbox"/>	Char	40		No		
4	S_NATIONKEY	<input type="checkbox"/>	Char	9		No		
5	S_PHONE	<input type="checkbox"/>	Char	15		No		
6	S_ACCTBAL	<input type="checkbox"/>	Char	12		No		

Figure 10-19 Table definition for supplier data

Due to the way in which the file is created and that WebSphere Classic Federation makes the data available, we choose all column types to be of type *character*, even if the values represent numbers of type decimal and integer. Right after the WebSphere Classic Federation, a transformer stage (CHAR\_TO\_NUMBERS in Figure 10-18 on page 271) converts the character values in columns S\_SUPPKEY, S\_NATIONKEY, and S\_ACCTBAL to their corresponding numeric representations. The S\_ACCTBAL column, for example, is transformed by using a function such as StringToDecimal(Supplier.S\_ACCTBAL).

From the transformer stage, the rest of the job looks the same as though we were reading supplier data from a database table.



## 10.6 Running and monitoring an ETL job in DataStage Director

You can compile and run the job in the DataStage designer. You can see the output of your job execution by clicking **Tools** → **Run Director**. During the development and debug phase, you can use stages, such as the peek stage, to look into the data. By using the peek stage, the information can be written to the job log as you see a bit more in the next section.

### 10.6.1 Typical load challenges

While developing jobs, it is likely that you may encounter some challenges. In the following sections, we explain some of the challenges that are encountered during population of the jobs described previously.

#### Intermediate dataset space allocation issues

When you use a DB2z stage on DataStage, some intermediate data sets are produced. You mention these in the properties of the target DB2z stage. The name of the data set has the same prefix that you specify as a DSN Prefix option on the Properties page of the DB2z stage. When you are not using BatchPipes, but are using FTP for the files that are created on System z, you see the message shown in Example 10-1.

*Example 10-1 Error message with DB2z stage and FTP to a data set*

---

```
DB2Z_30,0: The runLocally() of the operator failed. Message Id.IIS-DSEE-TFOR-00089
DB2Z_30,0: Input 0 consumed 1 records. Message Id. IIS-DSEE-TFOR-00163
DB2Z_30,0: Operator terminated abnormally: runLocally did not return APT_StatusOk. Message
Id. IIS-DSEE-TFPM-00040
main_program: Step execution finished with status = FAILED. Message Id. IIS-DSEE-TFSC-00011
```

---

When you see this message, increase the row count estimate for the target of the load job. In this case, the target was a DB2z stage. Therefore, you double-click **DB2z** and then click **Properties** and then **Options** to choose the Row Count estimate and change it. If this estimate is too high, the job may fail again because it tries to create a data set on System z that is too large. We recommend that you use BatchPipes.

#### BatchPipe usage considerations

When using BatchPipes, you must enter the BatchPipes system ID as one of the options on the Property page for the target DB2z stage to where you are writing. We used the identifier BP01 in this project. When you use this option, you are likely to receive the error message shown in Example 10-6. For information about fixing this issue, refer to “Troubleshooting BatchPipes” on page 208.

*Example 10-2 BatchPipe related error messages*

---

```
DB2z_DWHODS_CUSTOMER,0: 1DSNU000I    107 08:37:46.35 DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = DB2ZLOAD
DSNU1044I    107 08:37:46.39 DSNUGTIS - PROCESSING SYSIN AS EBCDIC
ODSNU050I    107 08:37:46.39 DSNUGUTC - TEMPLATE TMPL_IN DSN 'SREENI.IN00000' SUBSYS(BP0) RECFM(FB) LRECL(120)
DSNU1035I    107 08:37:46.40 DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
ODSNU050I    107 08:37:46.40 DSNUGUTC - TEMPLATE TMPL_WK1 DSN 'SREENI.WORK1' UNIT SYSDA DISP(NEW, DELETE, DELETE)
SPACE(878, 87) CYL
DSNU1035I    107 08:37:46.40 DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
ODSNU050I    107 08:37:46.40 DSNUGUTC - TEMPLATE TMPL_WK2 DSN 'SREENI.WORK2' UNIT SYSDA DISP(NEW, DELETE, DELETE)
SPACE(878, 87) CYL
DSNU1035I    107 08:37:46.40 DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
ODSNU050I    107 08:37:46.40 DSNUGUTC - TEMPLATE TMPL_ERR DSN 'SREENI.SYSERR' UNIT SYSDA DISP(MOD, CATLG, DELETE)
SPACE(878, 87) CYL
DSNU1035I    107 08:37:46.40 DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
ODSNU050I    107 08:37:46.40 DSNUGUTC - LOAD DATA INDDN TMPL_IN RESUME NO REPLACE LOG NO NOCOPYPEND EBCDIC SORTKEYS
```

---

```

O SORTDEVT SYSDA WORKDDN(TMPL_WK1, TMPL_WK2) ERRDDN TMPL_ERR
DSNU650I -D912 107 08:37:46.40 DSNURWI - INTO TABLE DWHODS.CUSTOMER
DSNU650I -D912 107 08:37:46.40 DSNURWI - ("CUSTKEY_DW" POSITION(1:4) INTEGER,
DSNU650I -D912 107 08:37:46.40 DSNURWI - "C_NATIONKEY" POSITION(5:8) INTEGER NULLIF(114)=X'FF',
DSNU650I -D912 107 08:37:46.40 DSNURWI - "C_CUSTKEY_SRC" POSITION(9:12) INTEGER,
DSNU650I -D912 107 08:37:46.40 DSNURWI - "C_NAME" POSITION(13:39) VARCHAR NULLIF(115)=X'FF',
DSNU650I -D912 107 08:37:46.40 DSNURWI - "C_ADDRESS" POSITION(40:81) VARCHAR NULLIF(116)=X'FF',
DSNU650I -D912 107 08:37:46.40 DSNURWI - "C_PHONE" POSITION(82:96) CHAR(15) NULLIF(117)=X'FF',
DSNU650I -D912 107 08:37:46.40 DSNURWI - "C_ACCTBAL" POSITION(97:103) DECIMAL PACKED NULLIF(118)=X'FF',
DSNU650I -D912 107 08:37:46.40 DSNURWI - "C_MKTSEGMENT" POSITION(104:113) CHAR(10) NULLIF(119)=X'FF')
DSNU1015I 107 08:37:46.41 DSNUGDYN - ERROR ALLOCATING DATA SET DSN=SREENI.IN00000 CODE=X'04B00000'
DSNU1042I 107 08:37:46.41 DSNUGDYN - START OF IDCAMS MESSAGES
IKJ56231I DATA SET SREENI.IN00000 NOT ALLOCATED, SYSTEM OR INSTALLATION ERROR+
IKJ56231I DYNAMIC ALLOCATION REASON CODE IS X'000004B0'
DSNU1043I 107 08:37:46.41 DSNUGDYN - END OF IDCAMS MESSAGES
DSNU012I 107 08:37:46.41 DSNUGBAC - UTILITY EXECUTION TERMINATED, HIGHEST RETURN CODE=8

```

---

## 10.6.2 Hints for table organization while preparing for loading data

In this section, we provide several helpful hints while loading data. First, if a job output log is open in the Run Director and you try to rename the job in the DataStage designer, the log generates a message indicating that it cannot get an exclusive lock to make the change. In this case, exit from the job log and then make the name change.

Second, for a join stage to work, the column names must be identical. If a primary key moved into a child table as a foreign key, we recommend that you name them with the same name. Otherwise you may have to use a transformer stage to rename one of the columns for join purposes.

Third, double-check referential integrity among the tables in the source system to understand the discards in the lookups that can result in later loads. For example, all ensure that the `part_key` values in `OLTP_W.LINEITEM` and `OLTP_B.LINEITEM` are contained in the `p_partkey` in the `OLTP.PART` table. If you do not consistently look into the referential integrity, the fact table checks result with rejects in lookup with the dimensions.

## 10.7 Debugging load jobs: A brief look

DataStage has several stages and options to help in debugging. In the target DataStage job to which you are writing, you can choose the option `Verbose=true` for the DB2z stage to learn more about the messages that DataStage is generating.

You can also use the Peek DataStage job to look into the reject data as shown in the `PEEK_ORD_STAT_REJECTS` peek stage in Figure 10-20 on page 275. To create this example, we updated three order status values to a value that is not in the order status table. Six line items matched the three updated orders, and all the records showed up in this job log. For every lookup, you can keep a reject file in case of tracking any anomalies for a non-match. You can also choose to ignore a non-match based upon your needs.



## 10.8 Performance considerations

For smaller jobs, performance is not an issue. However, for jobs dealing with a large amount of data, the configuration file has to be set up to use partitioning and exploit the hardware capabilities. A *performance analysis feature* is available in the DataStage designer. By clicking this feature, you see the output shown in Figure 10-22.

### 10.8.1 Performance statistics

When the DataStage harvests the performance data, you see a chart like the example in Figure 10-22.

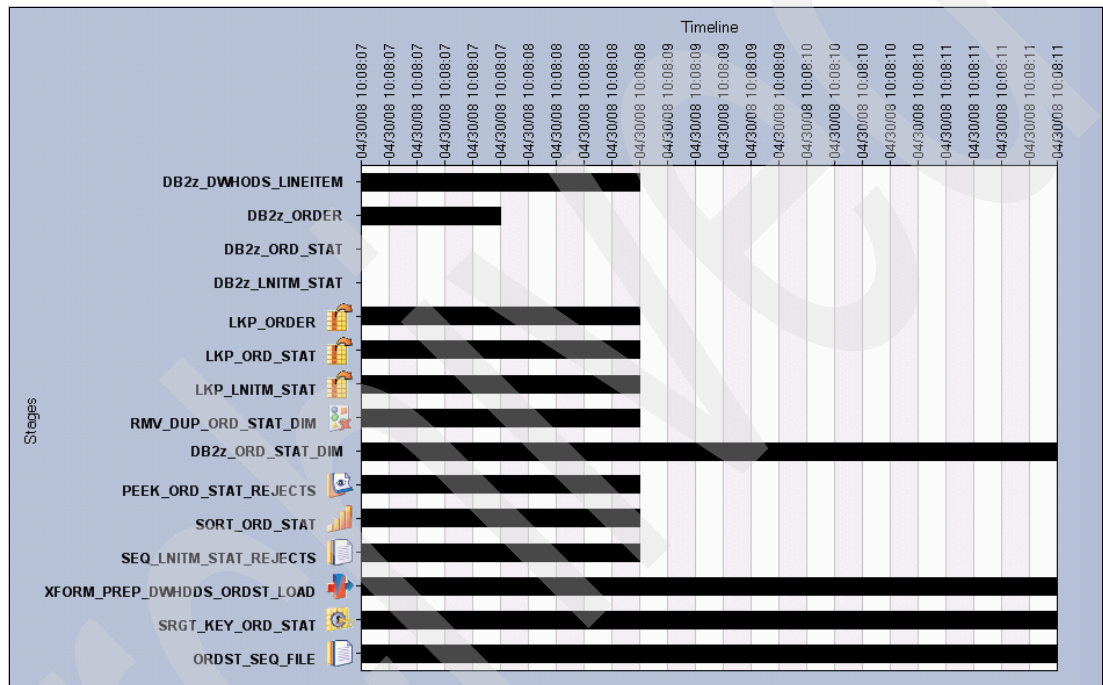


Figure 10-22 DataStage job performance statistics

In addition to the job timeline, you have the option charts, which include charts for CPU utilization, memory utilization, disk utilization, process density, and so on. By clicking Charts and the CPU, you can see CPU utilization, memory utilization, disk utilization, process density, and so on. Figure 10-23 shows the corresponding CPU utilization.

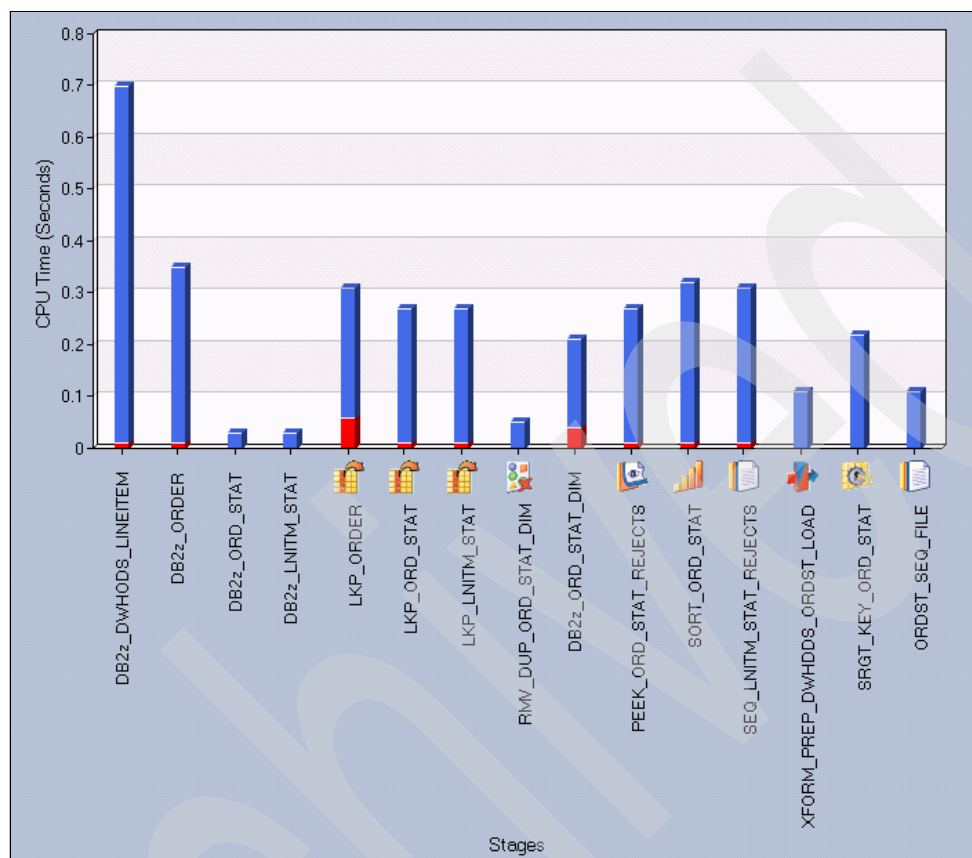


Figure 10-23 DataStage job CPU utilization

## 10.8.2 Parallel jobs versus server jobs

All the jobs that were developed for this book are parallel jobs. Parallel jobs allow parallel processing on symmetric multiprocessing (SMP), massively parallel processing (MPP), and cluster systems. The *WebSphere DataStage Parallel Job Advanced Developer Guide*, LC18-9892, has examples of configuration files that you need for SMP, MPP and cluster systems to help DataStage exploit these capabilities. In the DataStage context, there are two kinds of parallel processing: pipeline and partitioning.

### Pipeline parallelism

Suppose that you have a job such as the example shown in Figure 10-7 on page 264 where you have a Source DB2z stage, a transformer stage, and a target DB2z stage. In pipeline parallelism, where you have three processors, for example, one processor starts reading the source stage. The second processor runs the transformer stage to transform the data in some way, and the third processor writes to the target. All three processors work simultaneously without waiting for sequential completion.

## Partitioning parallelism

In partitioning parallelism, the same job is run simultaneously by several processors with each processor handling a separate subset of the data. At the end of the job, all the subsets of data can be collected back and written to a single target.

## Pipeline and partitioning parallelism

You can combine pipeline and partitioning parallelism by having all stages run simultaneously and for subsets of data to be worked on by various processors. Figure 10-24 from the *WebSphere DataStage Parallel Job Advanced Developer Guide*, LC18-9892, shows this combination.

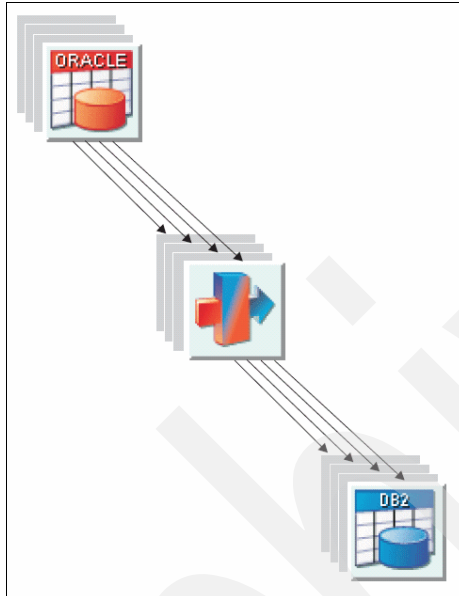


Figure 10-24 Combining pipeline and partitioning parallelism

## Server jobs

Server jobs are compiled and run on the IBM WebSphere DataStage server sequentially.

### 10.8.3 Choice of stage for performance improvement

The join stage must be preceded by a sort, which can be accomplished by using an ORDER BY clause when the source data is extracted. In the case that the source stage is a DB2z stage, then it uses the sort area on the System z server to do the sorting. If there is a funnel stage after the source stage, choose a Funnel type of Sort Funnel to ensure that the ordering is maintained. Sort data using the database is a best practice.

When doing the lookup in DataStage, the size of the reference table used in the lookup matters. If it is small, it does not impede in the running of the job from the lookup point of view. If it is large, consider the use of a join stage instead.

In instances where the population of a fact table by one job is taking a long time, explore the possibility of slicing the job so that parts of the job can produce output that can be dumped into data set stages that do not use as much memory locally. Having a slowly changing dimension stage has an overhead in that it involves the creation and lookup of a key.

A best practice is to use the database name, user ID, and password as a job parameter so that it is not tied to one individual developer. To do this, you can select **Edit** → **Job Properties** and click the **Parameters** tab (Figure 10-25). Then when you establish your access to the database for your DB2z stage either for reading or writing, you can insert a job parameter instead of hard coding a value into it.

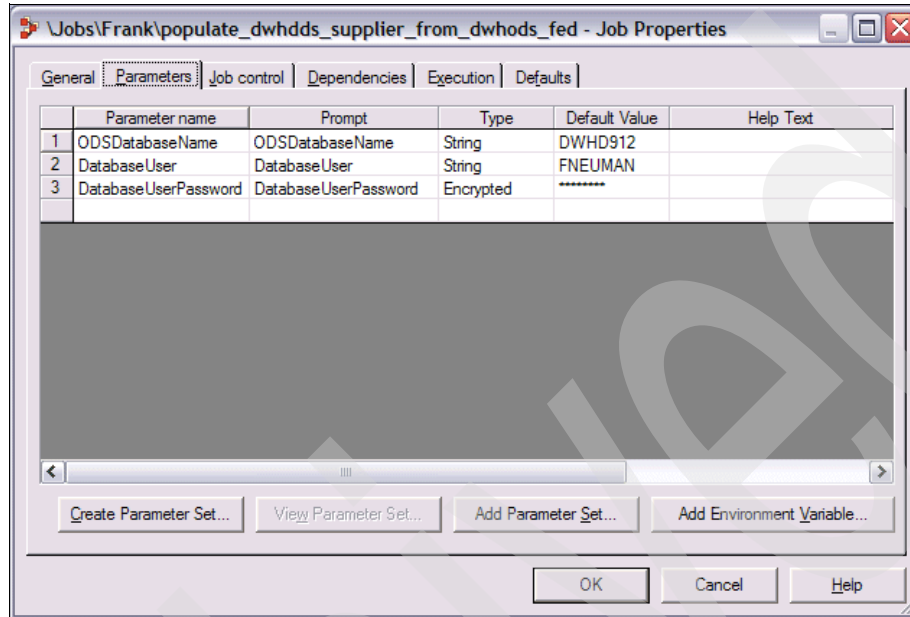


Figure 10-25 DataStage job parameters for database name and credentials

## 10.9 Naming standards

We used naming standards for the purposes of this book. You can follow a standard consistent with your company standards. In our jobs, we tried to have all DB2z stages starting with DB2z, all lookup stages starting with LKP, all links beginning with LNK, funnels beginning with FNL and so on. However our main task was to show how DataStage can be used quickly and relatively easily to populate all the ODS and DDS tables on DB2 for z/OS.

## 10.10 Data quality implementation during full load

The implementation of the load in this book is low in complexity compared to data warehouse loads in the real world with which companies struggle all the time. The reasons for the complexities are well known in the industry such as the integration of data from different online transaction systems and the lack of data conforming to the required business specifications.

For the purposes of the load required for the business scenario in this book, we used only simple features, such as peek stages for debugging, to keep a sequential reject file for lookup rejects and carefully monitor load counts. In a real world situation, it is important to perform analysis of the data to be loaded before the design and implementation of DataStage jobs. In this context, an additional tool called *QualityStage* is available from IBM. QualityStage shares the same development environment with DataStage.

You can use QualityStage stages in your DataStage and vice versa. QualityStage helps in various data quality processes such as the following examples:

- ▶ Data investigation
- ▶ Data standardization
- ▶ Data matching
- ▶ Data survivorship

Discussing these processes is beyond the scope of this book. The following manuals are available for those of you who are interested in using the QualityStage tool along with DataStage:

- ▶ *WebSphere QualityStage User Guide*, SC18-9922
- ▶ *WebSphere QualityStage Tutorial*, SC19-9925

The most current information about the IBM Information Server suite, which includes DataStage and QualityStage functions, is available in the IBM Information Server information center at the following address:

<http://publib.boulder.ibm.com/infocenter/iisinfsv/v8r0/index.jsp>



## Incremental update with DataStage

Chapter 10, “Full load using DataStage” on page 257, shows the initial load into the data warehouse. After the initial load is done, you must incrementally load the data warehouse to reflect the updates that are taking place in the online transaction system. The ability to update your data warehouse environment with low latency enables near real-time business intelligence (BI).

Therefore, you must define the *frequency* of the incremental load, which mainly depends on the business requirements. In the business realm, a gradual shift is taking place. Initially data warehouses were established with a few users in mind. They generally looked at the past historical data and predict trends. This type of historical integrated look at customers, along with the products that they buy, for example, was not available to general everyday customer interacting company employees. When a customer walked into the door, an employee had no idea of any cross-selling opportunities.

To open a data warehouse to customer facing employees, current data, the ability of the database to support the requests from several users, and the mixed workload of small queries with queries that, for example, use an exact account number or social security number must be known. At the same time, larger queries that are used to build either data marts or extensive reports with historical data must also be catered to. This information makes the difference blurry between a data warehouse and an online transaction system. Fortunately technologies can help meet the requirements of these new groups of data warehouse users.

The products in Chapter 10, “Full load using DataStage” on page 257, and in this chapter are product modules of the IBM Information Server for System z. In Chapter 10, we look at IBM InfoSphere DataStage for Linux on System z and IBM WebSphere Federation Server and talk about IBM InfoSphere QualityStage for Linux on System z. In this chapter, we look at how we can use DataStage to read the messages that have been written by the IBM WebSphere Data Event Publisher based on online transaction processing (OLTP) transactions. This Event Publisher is also a module of the IBM Information Server for System z. We also point to tools, such as Workload Manager, that can be tailored to cater to mixed query workloads such as where real-time BI or operational BI is enabled.

This chapter includes the following topics:

- ▶ 11.1, “Operational BI revisited” on page 282
- ▶ 11.2, “Reduction of a batch window” on page 283
- ▶ 11.3, “Usage of queues as sources of data” on page 283
- ▶ 11.4, “Anatomy of a queue” on page 284
- ▶ 11.5, “DataStage job to read a queue” on page 285
- ▶ 11.6, “Automated reading of the queues and updating of the data warehouse” on page 286
- ▶ 11.7, “Concurrency considerations” on page 292
- ▶ 11.8, “Summary” on page 293

## 11.1 Operational BI revisited

The first time you go to a doctor’s office, you complete a form to give the doctor access to your complete medical history. If you go to this doctor again, a nurse records on a chart your weight, temperature, blood pressure, and most recent health issue for which you are at the doctor’s office and adds this information to your medical history. The doctor reviews your chart for your previous history and current issues in order to have the right context to perform an examination and provide proper medical guidance at the time of service.

Whenever a history of information can be applied to optimize service and care, it can be construed as *operational BI*. You must determine whether you can provide your customer-facing personnel with the ability to look at the history of a customer’s interaction with your company. You do this to satisfy day-to-day tasks and to see the interaction as an opportunity to look through their history. For example, you might look at past complaints to avoid a particular problem or problems again. You might look at the products a customer likes to use as a base for cross-selling other products. Or you might look at any consistent loss of business and pass that information to the relevant product development group. If you can make these determinations, then you are employing operational BI by empowering your customer service personnel to ensure customer satisfaction. How well you do this differentiates you from the competition.

The users of the operational BI can be both the customer facing personnel and the customers themselves, which results in a massive increase of users against the data warehouse. Operational BI is about optimizing the day-to-day business and not a replacement of the strategic BI for which the warehouses were initially developed.

Senior executives need strategic requirement to look at corporate-wide trends and to set a company-wide direction based on that information. For example, a government executive working at the governor’s office in the state of Ohio (U.S.) might ask questions related to alcohol and drug addiction support, for example: How many dollars per successful outcome is the government spending by county and over time? A successful outcome is the abstinence of alcohol and drugs along with employment of a person who availed the government’s care during their time of addiction.

With operational BI, you learn about the precursors to a behavior, such as alcohol addiction in this case, when providing services, such as rehabilitation. You can assess the likelihood that a person will repeat this behavior again, such as to abuse alcohol again, and provide basic preventive measures. There can be other examples for transforming a business to be more dynamic *operationally*.

## 11.2 Reduction of a batch window

The advent of operational BI has reduced the batch window due to the need for real-time data for some source systems to percolate to the data warehouse environment. Even if service-level agreements (SLAs) with IT customers do not change, the nature of a tiered data warehouse environment contributes to the batch window. Also as more source systems are added and, therefore, more data is in the data warehouse, there is a concern of not meeting the SLAs. In addition, upgrades are made to source systems that may prevent you from getting a monthly load file on time, assuming a monthly feed from that source.

You need an array of solutions, such as a combination of technical hardware, a database, and related software, and a clear delineation of business requirements of priority that is reflected in the data acquisition and distribution approach. In previous chapters, we show the performance, architecture, and continuous availability of System z hardware along with DB2 for z/OS and related software, such as DataStage, working together to account for limited batch windows.

## 11.3 Usage of queues as sources of data

Message queues as a source of data are pivotal in a real-time environment. In Chapter 8, “Q replication and event publishing” on page 159, we explain how you can use WebSphere Data Event Publisher for z/OS to read OLTP database recovery logs and publish these events as delimited or XML messages into a queue. DataStage can then read these messages and use them to update the data warehouse tables. These messages are the updates made to the source transaction system. By reading the messages and applying them to the data warehouse, you enable your data warehouse with low latency updates. Not all subject areas of your warehouse may need low latency. Therefore, you can select to which data warehouse areas you need to apply OLTP updates as soon as possible and so on. For those for which a batch update to their data warehouse suffices, then you can use DataStage to write to a staging table.

The IBM WebSphere Replication Server and IBM WebSphere Data Event Publisher are both modules of the IBM Information Server for System z. They can be used to read the recovery logs via the Q Capture program and write to WebSphere MQ queues. In Replication Server, the Q Apply program can read the WebSphere MQ queue and apply the changes to a target table. In Event Publisher, there is no Q Apply program. There is only the Q Capture program that writes to the WebSphere MQ queue in delimited or XML format.

We use Q Capture to satisfy the business scenario in this book. After the queue information is read, the data must be populated first in the operational data store (ODS) and then in the DDS. Sometimes lookups are done on existing data to keep track of history requirements and then as needed on end-date old information and inserted new data. Since it is not a straight insert into the target tables, we used DataStage.

Changes that are made to your source tables are captured by the Q Capture program from the recovery log. The Q Capture program runs on the server on which the source tables are located. Since the OLTP tables in the scenario for this book reside on DB2 for z/OS, the server refers to the DB2 subsystem on z/OS. You can choose, via the Q Capture program, whether you want only new data values in your message or to have both old and new data values in your message. Similarly you can also choose changed columns only or both changed and unchanged columns for Q Capture to use to build the message. For Inserts, you have only new information.

The Q Capture program uses a set of control tables, known as the *Q Capture control tables*, to store the information that the Q Capture program needs in order to build the messages in the send queue, which is ready to be read by a receiving application such as DataStage. The Q Capture control tables define the data source tables and the data to extract from the recovery log and write to the queue for each of these source tables.

In reading the message queues with DataStage, one goal is to update the data warehouse ODS or enterprise data warehouse in real time. By doing this, customer-facing employees have current information when they read from the ODS or the enterprise data warehouse. The second goal is for this information to be cascaded to a data mart.

In our business scenario, the OLTP information is written to the ODS from which the history is kept in the dimensional data store (DDS) or data mart. To highlight the ability to provide real-time BI, we read from a message queue and update the first ODS and then the DDS. We look at the format of a queue and develop a DataStage job to read the queue and update the data warehouse. Lastly we discuss any concurrency considerations that arise from writing data to the data warehouse frequently while reading from it.

## 11.4 Anatomy of a queue

In your project folder in DataStage, in the subfolder Real Time, you find a WebSphere MQ Connector and an MQ message beneath it. This MQ message is the structure of the message that you need. Whenever you start, you use the WebSphere MQ Connector stage as shown in the stage named MQ\_CUSTOMER in Figure 11-1. Then if you have a transformer stage, such as XFORM\_GRAB\_PAYLOAD (also shown in Figure 11-1), and a link between them, you can drag the MQ message definition on top of the link.

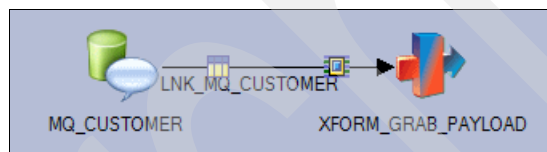


Figure 11-1 MQ message definition link

By doing this, when your MQ queue is populated on z/OS and we read that queue when we execute a DataStage job, the structure of the queue is in this definition format. For our purposes, the message that we needed was in one column, called *Payload*, at the bottom, after many columns that contain different information about the message such as QueueName and PutDate.

You can see the format of the payload column in the queue structure in the “WebSphere MQ message queues for the WebSphere Business Integration wrapper and the adapter” topic in the IBM DB2 Database for Linux, UNIX, and Windows Information Center at the following address:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp?topic=/com.ibm.swg.im.iis.ep.xmlpubsguide.doc/topics/iixexpmgrdelimstruct.html>

You can also see the structure of the payload column in Figure 11-5 on page 288. Although we named the columns, they are patterned after the metadata of the column structure shown in the information center at the previous Web address.

In the sections that follow, we explain how we use the Payload column information to move it to the data warehouse environment.

## 11.5 DataStage job to read a queue

The most important new DataStage stages that we need in an incremental load, as compared to a full load, are the WebSphere MQ Connector stage and the Open Database Connectivity (ODBC) Enterprise stage. The MQ Connector stage is used to read message queues that have been written by Event Publisher. The ODBC Enterprise stage is used in an upsert mode to update rows or insert rows based on a match, or lack thereof, of the key column that is identified.

### 11.5.1 WebSphere MQ Connector stage

Figure 11-2 shows a sample MQ Connector stage entry. This entry is used to read the queue for the updates that are made to the OLTP.CUSTOMER table.

LNK_MQ_CUSTOMER	
Properties   Columns   Advanced	
▼ Connection	
Mode	Client
Queue manager	Q801
Username	
Password	
▼ Client channel definition	
Channel name	DWHCHNL
Transport type	TCP
Connection name	10.10.10.2(1414)
▼ Usage	
Queue name	REDDWH.D911.OLTP.EVPUB.DEL
Access mode	As in queue definition
► Other queue settings	No
Wait time	-1
Message quantity	10
End of data message type	
► Refresh	No
Message read mode	Keep
▼ Transaction	
Record count	2
► End of wave	None
► Message options	No
► Error queue	No
► Filter messages	No
► Publish/Subscribe	No

Figure 11-2 Sample WebSphere MQ Connector stage entries

## 11.5.2 ODBC Enterprise stage

The ODBC Enterprise stage is used to update and then insert those rows that do not match the update criteria in the ODS customer table. In order for this to happen, the *upsert mode* of “Update then Insert” is specified. Subsequent to this stage, we might account for all the rows that came in via the message queue. We must be careful that the columns coming in from the message queue are renamed just before this ODBC Enterprise stage to match the column names exactly as in the target ODS Customer table. The upsert does not work if the column names that come in are different. We accomplish this name alignment in the XFORM\_FOR\_UPSERT transformer stage. Figure 11-3 shows the settings of this ODBC Enterprise stage

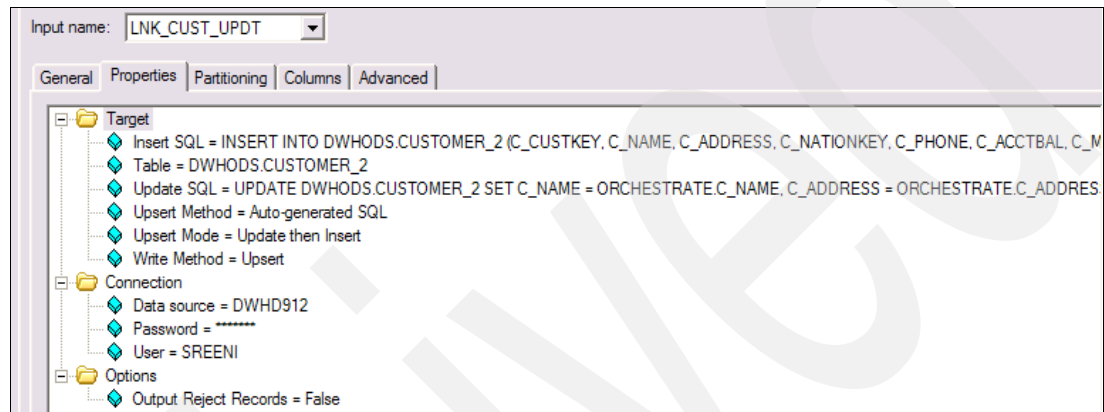


Figure 11-3 ODBC Enterprise stage settings

## 11.6 Automated reading of the queues and updating of the data warehouse

Our immediate goal is to read the message queues that are populated based on the changes to the OLTP tables and update the ODS and then the DDS. This shows the possibility of real-time BI in action. We tested this for one table, the OLTP.CUSTOMER table. In Chapter 8, “Q replication and event publishing” on page 159, we explain how the message queue is populated. In this section, you see the transport of this message first from the queue to the DWHODS Customer table and then to the DWHDDS Customer table. During development, the messages were first written to a CUSTOMER\_2 table, which is a mirror image of CUSTOMER table for testing purposes.

### 11.6.1 Update of the DWHODS Customer table

All the data needed for the update and insert is provided in the Payload column of the message queue. Therefore, as soon as the message is read from the WebSphere MQ Connector stage, named MQ\_CUSTOMER in Figure 11-4, the message is passed into a transformer stage, XFROM\_GRAB\_PAYLOAD, to pull just the payload column. Figure 11-4 shows the entire job that was developed to read the queue and populate the ODS Customer table.

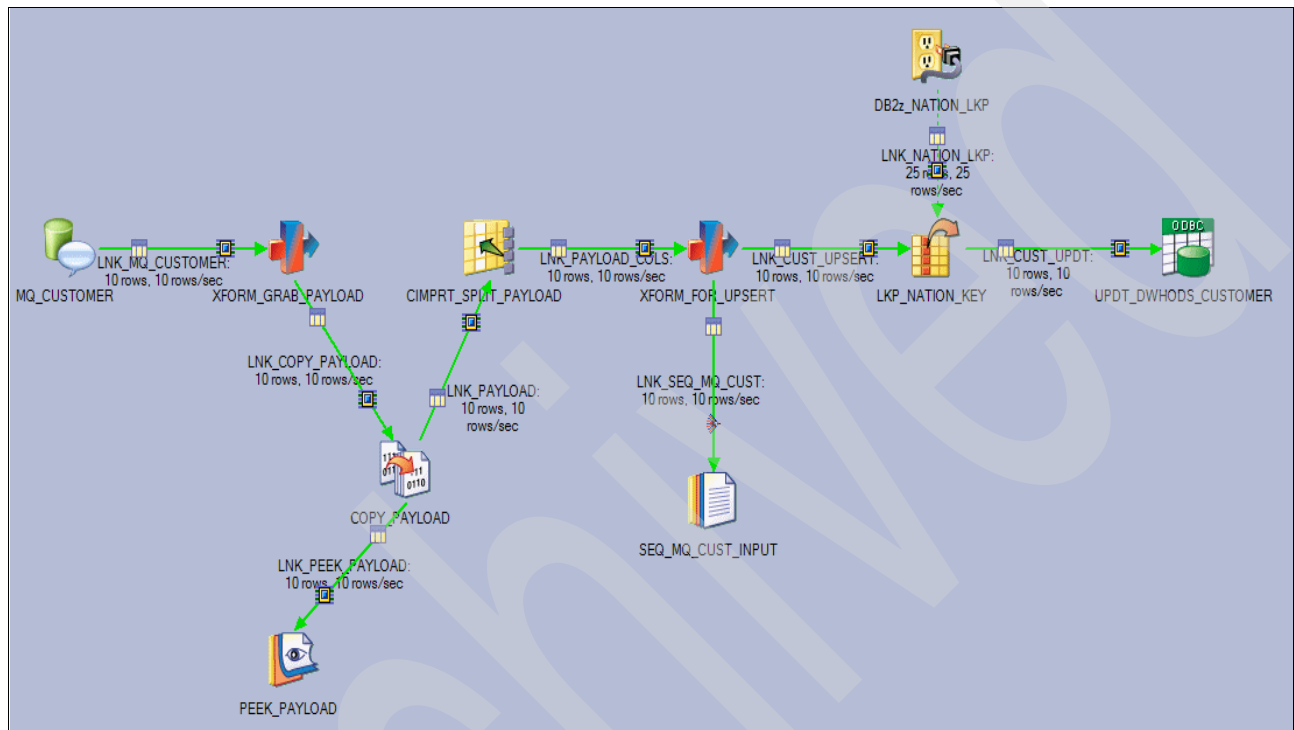


Figure 11-4 Read message queue to update ODS Customer table

onsisting columns

the log is created. This log shows the anticipated number of rows for the

10:49:05 AM	5/3/2008	Control	Sta
10:49:06 AM	5/3/2008	Info	Enr
10:49:06 AM	5/3/2008	Info	Pa
10:49:06 AM	5/3/2008	Info	ma
10:49:06 AM	5/3/2008	Info	ma
10:49:07 AM	5/3/2008	Info	MC
10:49:07 AM	5/3/2008	Info	UP
10:49:07 AM	5/3/2008	Info	ma
10:49:07 AM	5/3/2008	Info	MC
10:49:07 AM	5/3/2008	Info	PE
10:49:07 AM	5/3/2008	Info	CIF
10:49:07 AM	5/3/2008	Info	UP
10:49:07 AM	5/3/2008	Info	UP
10:49:07 AM	5/3/2008	Info	UP
10:49:07 AM	5/3/2008	Info	UP
10:49:07 AM	5/3/2008	Info	UP
10:49:07 AM	5/3/2008	Info	UP
10:49:07 AM	5/3/2008	Info	SE
10:49:07 AM	5/3/2008	Info	ma
10:49:07 AM	5/3/2008	Info	ma
10:49:07 AM	5/3/2008	Info	Pa

the log is created. This log shows the anticipated number of rows for the

10:49:05 AM	5/3/2008	Control	Sta
10:49:06 AM	5/3/2008	Info	Enr
10:49:06 AM	5/3/2008	Info	Pa
10:49:06 AM	5/3/2008	Info	ma
10:49:06 AM	5/3/2008	Info	ma
10:49:07 AM	5/3/2008	Info	MC
10:49:07 AM	5/3/2008	Info	UP
10:49:07 AM	5/3/2008	Info	ma
10:49:07 AM	5/3/2008	Info	MC
10:49:07 AM	5/3/2008	Info	PE
10:49:07 AM	5/3/2008	Info	CIF
10:49:07 AM	5/3/2008	Info	UP
10:49:07 AM	5/3/2008	Info	UP
10:49:07 AM	5/3/2008	Info	UP
10:49:07 AM	5/3/2008	Info	UP
10:49:07 AM	5/3/2008	Info	UP
10:49:07 AM	5/3/2008	Info	UP
10:49:07 AM	5/3/2008	Info	SE
10:49:07 AM	5/3/2008	Info	ma
10:49:07 AM	5/3/2008	Info	ma
10:49:07 AM	5/3/2008	Info	Pa

Figure 11-6 Job log from update operation

This log also shows that the data from the recovery log and placed in a



them into a data warehouse environment. In this case, the near real-time possible updates are applied to the ODS Customer table.

## 11.6.2 Update of the DWHDDS Customer table

Our primary intention is to show the low latency update of the data warehouse environment. We want to accomplish this by reading the message queues that are populated from the recovery log. In the previous section, we show that it is possible to read the queue by using DataStage with the OLTP updates and reflect that information in the ODS. We choose customer information as an example. Therefore, we must push the same customer information into the data mart/DDS to make it go through the complete decision support loop.

The business requirement is that, whenever the business users look at the measures for orders and order line items, they need the corresponding customer information to be relevant at that time. Technically we must store historical customer information of the customer dimension table. As we explain in “Slowly changing dimensions” on page 12, we use the SCD Type 2 pattern when we store history information in a dimension table. The DataStage tool that we use has an SCD stage that you can set to Type 2 to help implement this business requirement.

The source DWHODS Customer table provides the same source customer identification key for updated information. When we pick it up, we need to store it in the CUST\_KEY\_SRC column in the Customer dimension table. We then assign a new surrogate key for the new changed information. An end date is required for the effectiveness of the older customer row. We set SCD\_RECENTFLAG to N for the older information row of the customer, and we set SCD\_RECENTFLAG to Y for newer information of the same customer. We do this as part of an SCD Type 2 implementation.

Figure 11-7 shows the DataStage job that we developed.

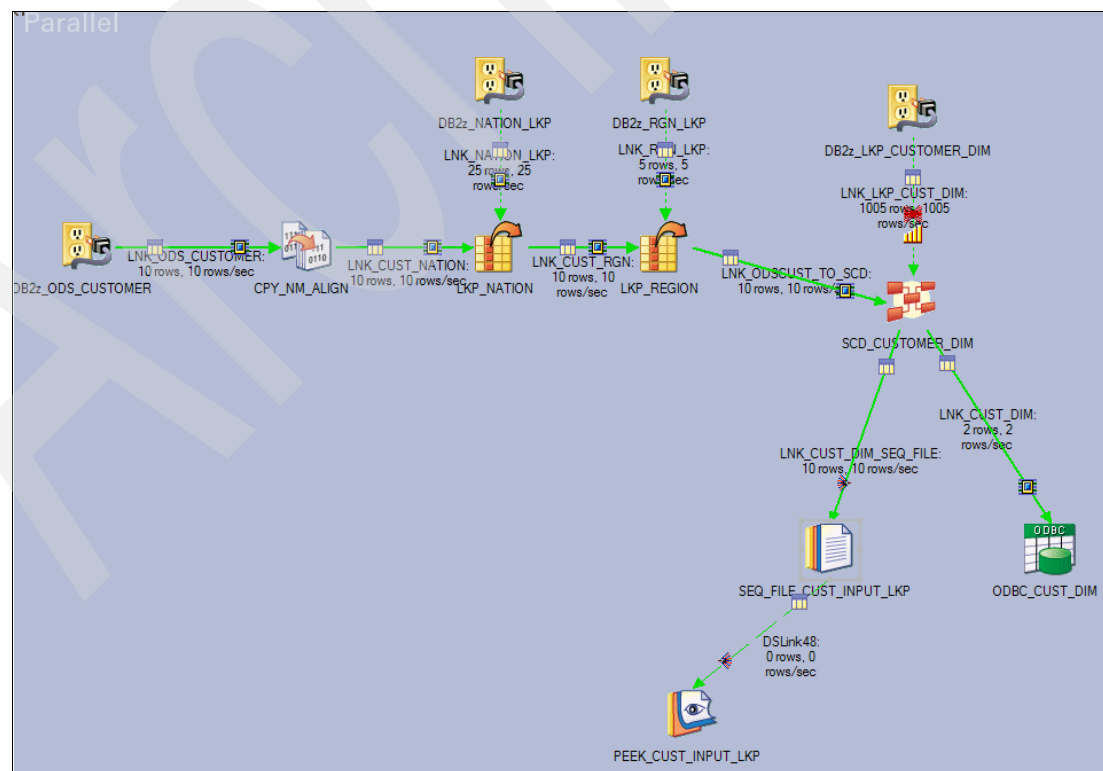


Figure 11-7 SCD implementation example

Let us walk through the steps and look at the results. Since our source ODS data is on DB2 for z/OS, we use a DB2z stage. The names that you pass from this table must match the target column names in the dimension table for the upsert mode to work in the ODBC Customer dimension stage. For this purpose, when we send data to LNK\_ODSCUST\_TO\_SCD, we change the column names from the source columns to match the target columns.

To define the SCD stage, you need two inputs: the ODS input of customer information and a lookup to the dimension table. This dimension is also what you are going to update. The lookup is needed in order that DataStage can first see if there are any changes for the input rows that you are getting for which you must create history rows in the dimension table. The lookup is based on the customer source key that comes from the ODS Customer table and is matched to the customer source key in the DDS Customer dimension table, as shown in Figure 11-8.

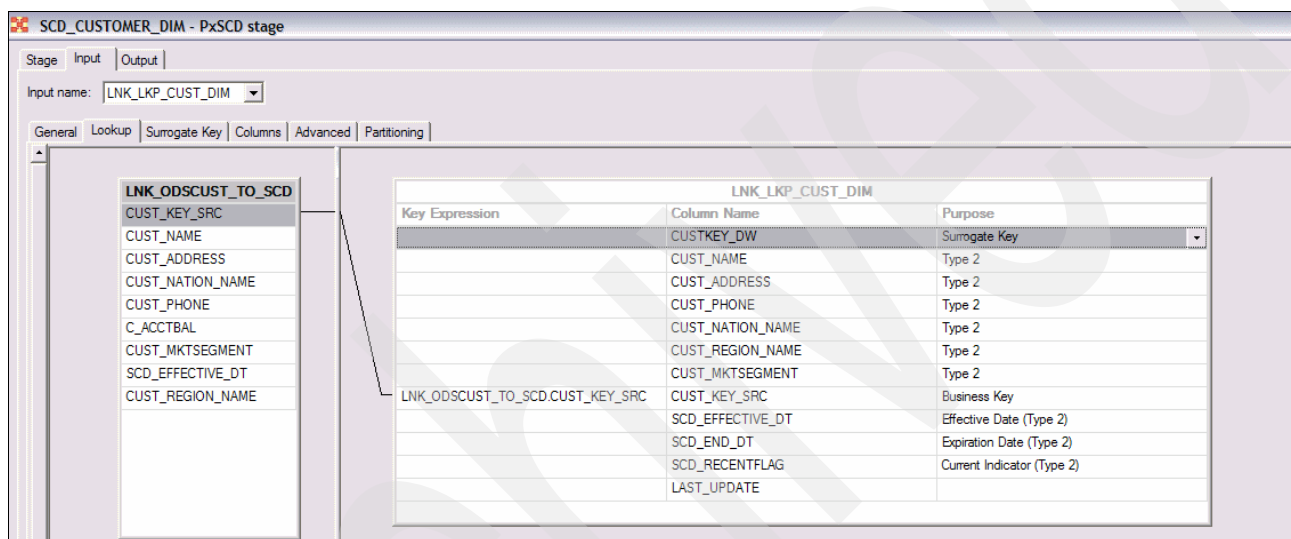


Figure 11-8 SCD Lookup of dimension table

For the target Customer dimension table, you also must create a surrogate key, which is defined as shown in Figure 11-9.

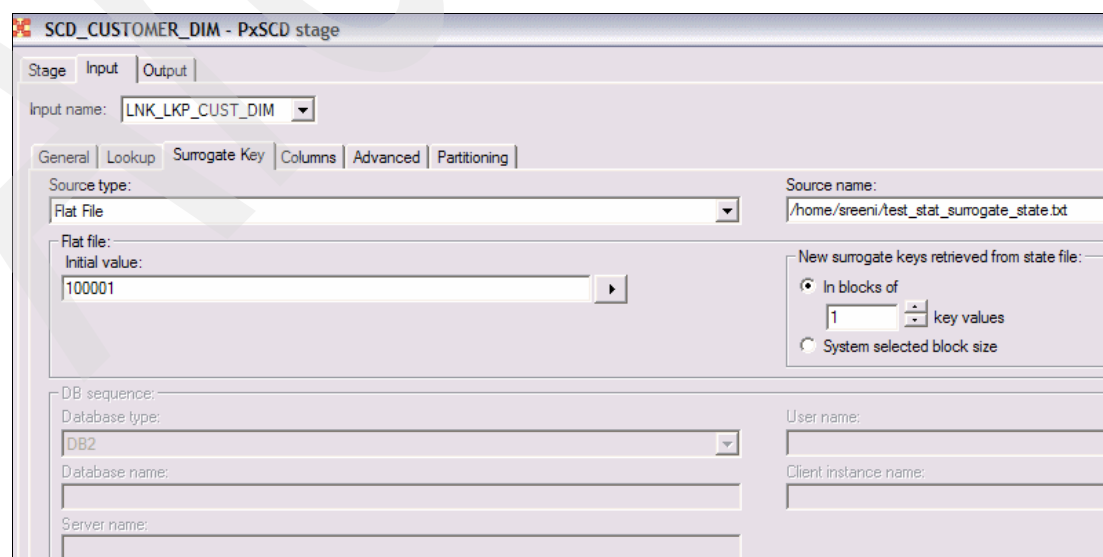


Figure 11-9 SCD Surrogate Key settings

Next you specify the output from the SCD stage. Figure 11-10 shows an example of the mapping that updates the Customer dimension table.

Derivation	Column Name	Purpose	Expire
NextSurrogateKey()	CUSTKEY_DW	Surrogate Key	
LNK_ODSCUST_TO_SCD.CUST_NAME	CUST_NAME	Type 2	
LNK_ODSCUST_TO_SCD.CUST_ADDRESS	CUST_ADDRESS	Type 2	
LNK_ODSCUST_TO_SCD.CUST_PHONE	CUST_PHONE	Type 2	
LNK_ODSCUST_TO_SCD.CUST_NATION_NAME	CUST_NATION_NAME	Type 2	
LNK_ODSCUST_TO_SCD.CUST_REGION_NAME	CUST_REGION_NAME	Type 2	
LNK_ODSCUST_TO_SCD.CUST_MKTSEGMENT	CUST_MKTSEGMENT	Type 2	
LNK_ODSCUST_TO_SCD.CUST_KEY_SRC	CUST_KEY_SRC	Business Key	
CurrentTimestamp()	SCD_EFFECTIVE_DT	Effective Date (Type 2)	
SetNull()	SCD_END_DT	Expiration Date (Type 2)	CurrentTimestamp()
"Y"	SCD_RECENTFLAG	Current Indicator (Type 2)	"N"

Figure 11-10 SCD dimensional update mapping

If you want to look at the input rows that are being passed into the SCD stage, then you can define a text file to capture this data. Figure 11-11 shows the mapping for this. As in creating regular SQL scripts and programming, watch for handling NULL values, especially during matching data for lookups and the like.

Derivation	Column Name	Expire
LNK_LKP_CUST_DIM.CUSTKEY_DW	CUSTKEY_DW	
LNK_LKP_CUST_DIM.CUST_NAME	CUST_NAME	
LNK_LKP_CUST_DIM.CUST_ADDRESS	CUST_ADDRESS	
LNK_LKP_CUST_DIM.CUST_PHONE	CUST_PHONE	
LNK_LKP_CUST_DIM.CUST_MKTSEGMENT	CUST_MKTSEGMENT	
LNK_LKP_CUST_DIM.CUST_KEY_SRC	CUST_KEY_SRC	
LNK_LKP_CUST_DIM.SCD_EFFECTIVE_DT	SCD_EFFECTIVE_DT	
LNK_LKP_CUST_DIM.SCD_END_DT	SCD_END_DT	
LNK_LKP_CUST_DIM.SCD_RECENTFLAG	SCD_RECENTFLAG	
LNK_LKP_CUST_DIM.CUST_NATION_NAME	CUST_NATION_NAME	
LNK_LKP_CUST_DIM.CUST_REGION_NAME	CUST_REGION_NAME	

Figure 11-11 SCD output to file

## Testing the functioning of SCD Type 2

To test the functioning of this SCD Type2 job for a source key of 211611, we had one row in the Customer dimension with a name. For the same source key of 211611, we had a different name come in the input from the ODS. Now we must see the different names on two different

rows. We must also see the end date and the SCD\_RECENTFLAG updated for the older row. This is shown in Figure 11-12 and Figure 11-13. SCD\_END\_DT is populated and SCD\_RECENTFLAG is set to N for the older row. The new row shows N in the SCD\_END\_DT and Y for the SCD\_RECENTFLAG.

CUSTKEY_DW	CUST_NAME	CUST_KEY_SRC	SCD_EFFECTIVE_DT
100007	SREENIVASA JANAKI	211611	2008-05-06-13.44.01.000000
100008	Andrew J C	211611	2008-05-06-13.53.41.000000

Figure 11-12 SCD result with the effective date

CUSTKEY_DW	CUST_KEY_SRC	SCD_END_DT	SCD_RECENTFLAG
100007	211611	2008-05-06-13.53.41.000000	N
100008	211611		Y

Figure 11-13 SCD result continued with the end date and recent flag

## 11.7 Concurrency considerations

To enable real-time BI, we populate the data warehouse environment with low latency. This means that we are inserting rows into the tables while users may be reading the data from those tables. By using such tools as z/OS Workload Manager, you can set up the priorities for your work. Refer to 3.2.2, “Workload management” on page 35, for more information.

For the considerations here, in incremental loading, you can use Workload Manager to put together a policy to favor the queries to read the data when such a need is more pronounced. You can also switch the policy to refresh the data when such a need takes precedence. Figure 11-14 illustrates these capabilities.

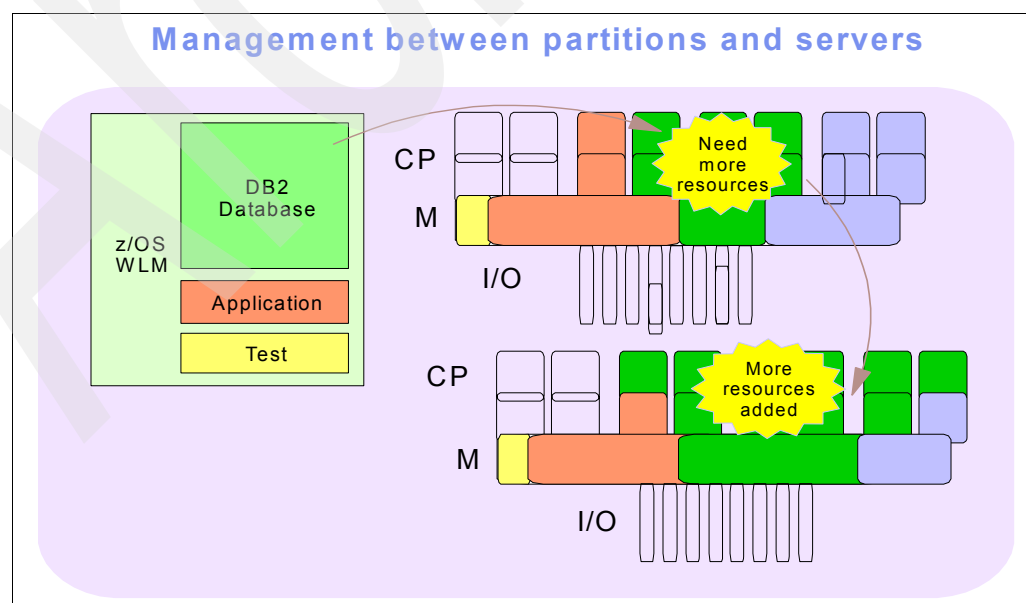


Figure 11-14 Workload Manager helping to prioritize for operational BI

## 11.8 Summary

This chapter showed that, from a technology standpoint, opportunities for real-time BI are opening up for managers who work on a daily basis with customers. They can use these opportunities to cross-sell and cement the bond with customers for long-term relationships. For example, you can open a report each day to see if the top 100 customers in your area submitted any complaints. You can reach out to them and placate them as necessary to void a difficult situation.

Depending on a data warehouse implementation, sometimes data fails data quality checks and is not written to the data warehouse. This can result when the data warehouse does not have the updated code tables when new codes are entered in the online transaction system. Therefore, the transactions in OLTP have these codes, such as General Ledger codes in a banking environment, but the data warehouse does not. In such cases, you can either use the logs of the transaction system or tools to write to message queues from which they can be used to update the data warehouse with relatively low latency.

There can also be help by using federation, which allows for reading a text file, for example, on a source application system and using the data as though it were in a table. We discussed reading the queues in this chapter, but explain how to use federation in 10.5, “Accessing WebSphere Classic Federation in DataStage jobs” on page 271.

Archived

# An operational business intelligence implementation

The advent and requirement for operational business intelligence (BI) is one of the important reasons why you should consider a data warehouse to run on the System z server.

As outlined in 2.4.1, “Operational business intelligence” on page 26, one aspect of operational BI is adding embedded analytics to online transactional processing (OLTP) systems. In our scenario, we run a transactional processing system, dealing with order processing. In this chapter, we show implementation examples for an application with embedded analytical information, derived from the operational data store (ODS) and dimensional data store (DDS) of our data warehouse.

Combining both OLTP data and historical data from the data warehouse in the same application and making them available to (potentially many) users emphasizes that both systems must comply to the same high standards and quality of service. If both data sources are needed for tactical and business-critical applications, then the availability and quality of service for the data warehouse must be comparable with the one for OLTP data.

This chapter includes the following topics:

- ▶ 12.1, “OLTP application with embedded analytics” on page 296
- ▶ 12.2, “The order processing Web application” on page 297
- ▶ 12.3, “Implementation considerations” on page 299
- ▶ 12.4, “Improving response times by using materialized query tables” on page 300

## 12.1 OLTP application with embedded analytics

For this book, we simulate an order processing system with the help of a J2EE application running on Linux on System z. Figure 12-1 shows our OLTP data (by using schema OLTP, OLTP\_W, and OLTP\_B) and the data warehouse data on the right side. Both are managed by DB2 for z/OS. On the left side, the order processing Web application runs in an instance of WebSphere Application Server 6.1 on Linux on System z.

Potentially many users who work in order processing can access the Web application by using their Web browser and submit new orders to the system. We use call center agents as an example of users of these applications. However, the same conceptual architecture can also be used to run a Web application that is available directly for customers through the Internet.

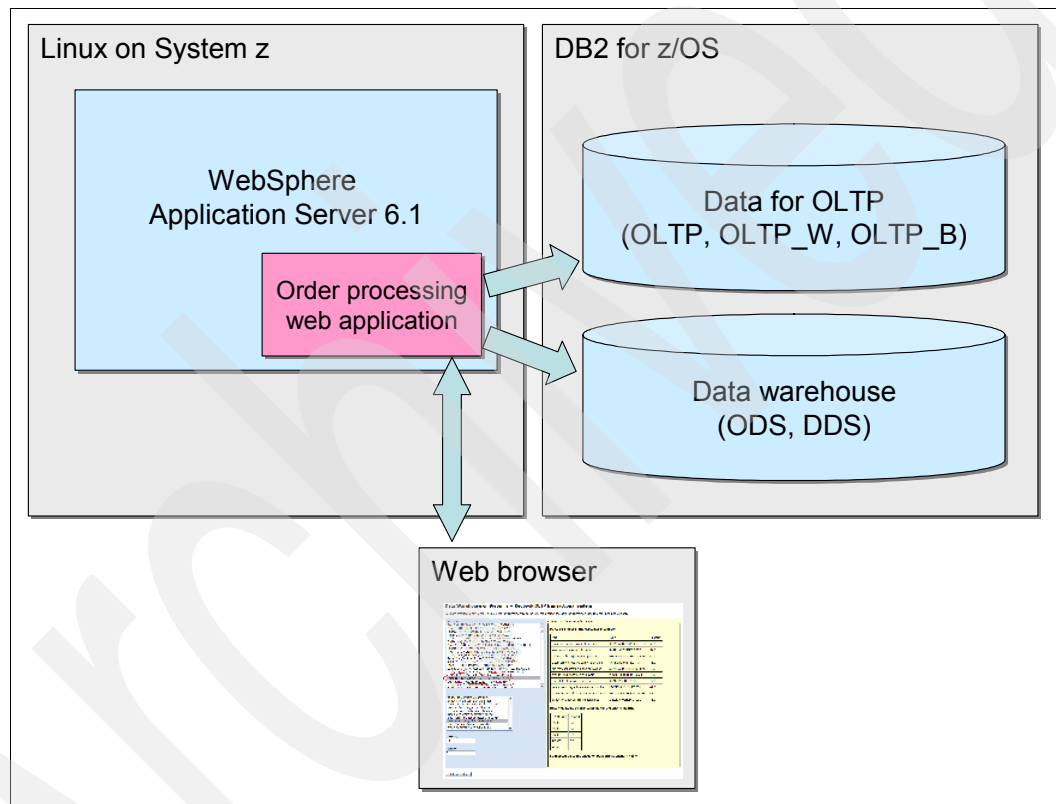


Figure 12-1 Order processing application that accesses both OLTP and data warehouse data

The Web application does not have to run on Linux on System z. We placed it there because we already had an WebSphere Application Server installed there. Running on WebSphere Application Server for z/OS works as well.

To achieve the best performance when accessing DB2 for z/OS, placing the Web application on System z (either Linux on System z or z/OS) is preferred over running the application on a distributed system.



## 12.2 The order processing Web application

Figure 12-2 shows a sample of a simple J2EE Web application that a call center agent might use to enter new orders for existing customers. On the left side (in blue), the agent first selects a customer from the list, then interactively adds the items (parts), and finally clicks the Submit Order button to complete the creation of the order in the transactional system. A *real* application requires additional functions for order processing that are not implemented in our application.

The order processing application contains a context-sensitive BI pane on the right side (in yellow). Based on a selected customer, this pane shows additional information that can help the call center agent to complete the order.

**Data Warehouse on System z -- Redbook OLTP transactional system**

To start creating a new order, first select the customer from the list, then select the part the customer likes to order and add quantity

**Customer**

- Andrew J C (EQ Park Lane12, 156872 A Town, CHINA )
- Khadja N A (BE Highway39, 170069 S Town, MOROCCO )
- Cris I M (JW Park Lane16, 140281 J Town, ETHIOPIA )
- Robert B J (QK Highway38, 109274 K Town, BRAZIL )
- Gaurav K G (JH Way8, 135493 O Town, GERMANY )
- Andrew X E (HF Place68, 101045 F Town, RUSSIA )
- Robert H A (PG Highway35, 132460 D Town, ETHIOPIA )
- Cris N O (OJ Crossing38, 162493 V Town, RUSSIA )
- Paolo T H (IT Place99, 184815 X Town, ARGENTINA )
- Khadja K T (FS Street75, 143045 G Town, KENYA )
- Cristian P Q (RS Place70, 142791 Q Town, FRANCE )
- Khadja F J (TM Ave55, 173728 B Town, MOROCCO )
- Sreeni P A (CE Place31, 157542 G Town, FRANCE )
- Gaurav X J (KV Highway57, 187376 D Town, ARGENTINA )
- Kirk A E (OH Blvd67, 175771 C Town, ARGENTINA )
- John V J (JT Blvd54, 187878 Y Town, SAUDI ARABIA )
- Michael G O (WD Ave2, 140368 P Town, FRANCE )
- Paul D C (MP Highway53, 102042 X Town, SAUDI ARABIA )
- Peter J A (KF Crossing49, 189724 K Town, IRAQ )

**Parts**

- goldenrod lace spring peru powder
- blush rosy metallic lemon navajo
- dark green antique puff wheat
- chocolate metallic smoke ghost drab
- forest blush chiffon thistle chocolate
- white ivory azure firebrick black
- blue blanchd tan indian olive
- ivory khaki cream midnight rosy
- thistle rose moccasin light floral
- floral moccasin royal powder burnished

Quantity

Priority

**Submit Order**

**Business Intelligence Information**

**Parts sold most in the customer's country**

Part	Type	Quantity
------	------	----------

**Returned items by this customer in the past 4 months**

Month/Year	Returns
------------	---------

**Suggested Discount based on past transactions: 0.0 %**

Figure 12-2 OLTP application for order processing

In our sample scenario, we decided to add the following information based on data stored in our data warehouse environment:

- In the first example, based on the country where the selected customer lives, the most sold parts in past transactions are derived. The call center agent might use this information to offer additional parts when talking to the customer. While this is simply an example, it is easy to imagine other kinds of data (for example, promotions based on region, showing parts that other customers combined) that can be derived and displayed to support the call center agent.

Example 12-1 shows the query for our DDS that we use to retrieve this information. We join the ORDER\_TRANSACTION\_FACT table with the PART\_DIM and CUSTOMER\_DIM dimensions and filter based on values in CUST\_NATION\_NAME for the selected customer. The value 211629 is an example, and the order processing applications run the query with the customer number of the selected customer.

---

*Example 12-1 Query for most sold parts in the country of a given customer*

---

```
SELECT P.PART_NAME, P.PART_TYPE, SUM(O.QUANTITY) AS SUM
FROM
    DWHDDS.ORDER_TRANSACTION_FACT O, DWHDDS.PART_DIM P, DWHDDS.CUSTOMER_DIM C
WHERE
    O.PARTKEY_DW = P.PARTKEY_DW AND
    O.CUSTKEY_DW = C.CUSTKEY_DW AND
    C.CUST_NATION_NAME =
        (SELECT CUST_NATION_NAME FROM DWHDDS.CUSTOMER_DIM WHERE CUSTKEY_DW=211629)
GROUP BY PART_NAME, PART_TYPE
ORDER BY SUM DESC
FOR READ ONLY
FETCH FIRST 10 ROWS ONLY
```

---

- The second example for BI data in our application is derived from the customer's past behavior. In our implementation, we query the number of parts that the selected customer returned in the past months and display an ordered list of month/year along with returned items. Again, this simplistic example can easily be enhanced to display a more sophisticated analysis on purchase behavior in the past. A call center agent might use this data when talking to the customer and ask how the company can improve the service in order to avoid returns. Or the agent might give a higher discount based on transactions done in the past.

Example 12-2 shows the query that we use to obtain the quantity of returned items per month. The customer number (211613 in this example) is again replaced with the value for the selected customer. The real code in the application uses a prepared statement with a parameter marker in this place.

---

*Example 12-2 Query for returned items per customer*

---

```
SELECT D.MONTH_VAL, D.YEAR_VAL, SUM( O.QUANTITY_RETURNED ) AS QTY
FROM
    DWHDDS.ORDER_TRANSACTION_FACT O, DWHDDS.DATE_DIM D
WHERE
    O.ORDERDATEKEY = D.DATEKEY_DW AND O.CUSTKEY_DW = 211613
GROUP BY MONTH_VAL, YEAR_VAL
ORDER BY QTY DESC
FOR READ ONLY
FETCH FIRST 5 ROWS ONLY
```

---

- The third example at the bottom of the BI panel is a suggestion for the discount that the customer might receive. Various information from the data warehouse and from transactional systems can be included to create this option. In our example, we choose the average discount that the customer received in the past.

Figure 12-3 on page 299 shows the retrieved values for a selected customer, Frank N Q (living in Japan). Upon selecting the customer, the BI information on the right side is instantly updated with the information described earlier. The first table, for example, shows that lemon lace khaki powder blanched is the most popular part in Japan and has been sold with a quantity of 55.0 so far.

## Data Warehouse on System z -- Redbook OLTP transactional system

To start creating a new order, first select the customer from the list, then select the part the customer likes to order and add quantity

Customer

- Sam B G (TM Road90, 182844 Z Town, JORDAN )
- Bob D E (NK Blvd36, 136045 L Town, RUSSIA )
- Gaurav R N (YF Road31, 141269 W Town, IRAN )
- Kirk M C (PP Ave60, 151726 F Town, CANADA )
- John I T (VT Pkway52, 187262 K Town, SAUDI ARABIA )
- Cristian X Q (EA Blvd12, 174090 Q Town, IRAQ )
- Paolo X L (MH Highway41, 143820 X Town, UNITED STATES )
- Robert M P (SV Park Lane49, 124829 R Town, IRAN )
- Mike Y C (GY Crossing53, 117507 K Town, FRANCE )
- Khadja Q Y (IE Highway87, 122295 I Town, IRAN )
- Peter R X (IW Ave65, 144717 D Town, CANADA )
- Kirk F Y (II Ave76, 177672 R Town, SAUDI ARABIA )
- Paolo J T (BI Highway30, 171855 Q Town, INDIA )
- Gaurav Q K (WR Crossing36, 150696 F Town, MOZAMBIQUE )
- Khadja M C (QQ Crossing51, 159416 I Town, ETHIOPIA )
- Greeni P P (JJ Place23, 146040 M Town, PERU )
- Frank N Q (XZ Way94, 158481 X Town, JAPAN )**
- Chris R E (RV Road36, 161123 T Town, IRAN )
- Khadja S M (AR Place4, 184814 S Town, BRAZIL )

Parts

- dodger forest floral cream black
- bisque salmon dark blanchd linen
- lemon aquamarine firebrick floral almond
- medium floral beige cornsilk olive
- firebrick bisque slate rose blanchd
- saddle dim white honeydew spring
- blush forest magenta metallic turquoise
- lawn forest misty chartreuse snow**
- plum drab cornflower brown dim
- green honeydew dim cyan bisque

Quantity

Priority

Business Intelligence Information

**Parts sold most in the customer's country**

Part	Type	Quantity
lemon lace khaki powder blanchd	STANDARD BURNISHED TIN	55.0
lawn forest indian orchid peru	SMALL ANODIZED STEEL	54.0
goldenrod lace spring peru powder	PROMO BURNISHED COPPER	45.0
cream almond papaya salmon blanchd	SMALL BURNISHED TIN	45.0
tan ivory aquamarine indian burlywood	STANDARD BRUSHED BRASS	42.0
orange khaki papaya cyan navajo	PROMO BRUSHED NICKEL	42.0
cornsilk blush powder tan rose	SMALL PLATED STEEL	42.0
sandy misty magenta chocolate blanchd	PROMO BURNISHED TIN	41.0
brown burnished turquoise moccasin olive	MEDIUM BRUSHED COPPER	41.0
bisque pink dodger cream goldenrod	LARGE BRUSHED STEEL	41.0

**Returned items by this customer in the past 4 months**

Month/Year	Returns
2/2007	9.0
1/2007	7.0
5/2007	6.0
4/2007	6.0
3/2007	0.0

**Suggested Discount based on past transactions: 4.466 %**

Figure 12-3 OLTP application with BI information

## 12.3 Implementation considerations

We create our order processing Web application by using JavaServer™ Pages (JSPs) and connect it to DB2 for z/OS by using Java Database Connectivity (JDBC) and the DB2 Universal JDBC driver (Type 4). The same application connects to two DB2 for z/OS data sources. Depending on the requirement of the scenario and the setup of the two data sources, different techniques are required for an implementation.

### 12.3.1 Handling multiple data sources

In real-life scenarios, data that has been replicated might need to be enriched by combining it with data that resides in other tables of the OLTP or data warehouse database. Table 12-1 on page 300 illustrates different cases where OLTP and data warehouse data reside in the same subsystem or in different subsystem. It also indicates which techniques can be used to implement joins between data or if there is a need to access data in the same transaction. Typically, there is no need to combine data in the same transaction or to join them. Our sample application starts a new transaction for reading BI data from the data warehouse after the customer retrieval from the transactional system is completed.

Chapter 12. An operational business intelligence implementation 299

However, if there is a requirement to either join data, access data, or do both in the same transaction, then this can be achieved by using different subsystems.

*Table 12-1 Handling application requirements when using multiple DB2 for z/OS subsystems*

	Same DB2 subsystem	Different DB2 subsystems
Join data	No issue	Use federation or application logic (for example, temporary table)
Access data in the same transaction	No issue	Use global transactions and XA data source in WebSphere Application Server
Access data in a different transaction	No issue (one data source)	No issue (two data sources)

### 12.3.2 Leveraging tools to render business intelligence data

In a professional application, consider leveraging available products for both doing analysis and rendering of BI data. Examples for such applications include Cognos 8 BI, AlphaBlox and DataQuant. For more details, see Chapter 13, “Reporting and analysis with Cognos 8 BI” on page 305, and Chapter 14, “Reporting with DataQuant, QMF, and AlphaBlox” on page 397.

When building a Web application, also consider using WebSphere Portal. With WebSphere Portal, you can render BI data in additional (supporting) portlets on the same portal page where your transactional order processing system is displayed. In our example, communication between portlets can be used to send information about the selected customer to other portlets on the page and show supporting information there.

## 12.4 Improving response times by using materialized query tables

Because our data warehouse queries are run in context or in a transactional environment with a large number of users, the query response time for our SQL query in Example 12-1 on page 298 (popular parts in a country) might not be satisfying anymore. Therefore, we consider the definition of a materialized query table (MQT) to improve query response times and CPU costs.

We use IBM Optimization Service Center to review the access plan for the query (Figure 12-4 on page 301). It shows the accessed tables during query execution and also the join and sort operations to return the result for each individual request.

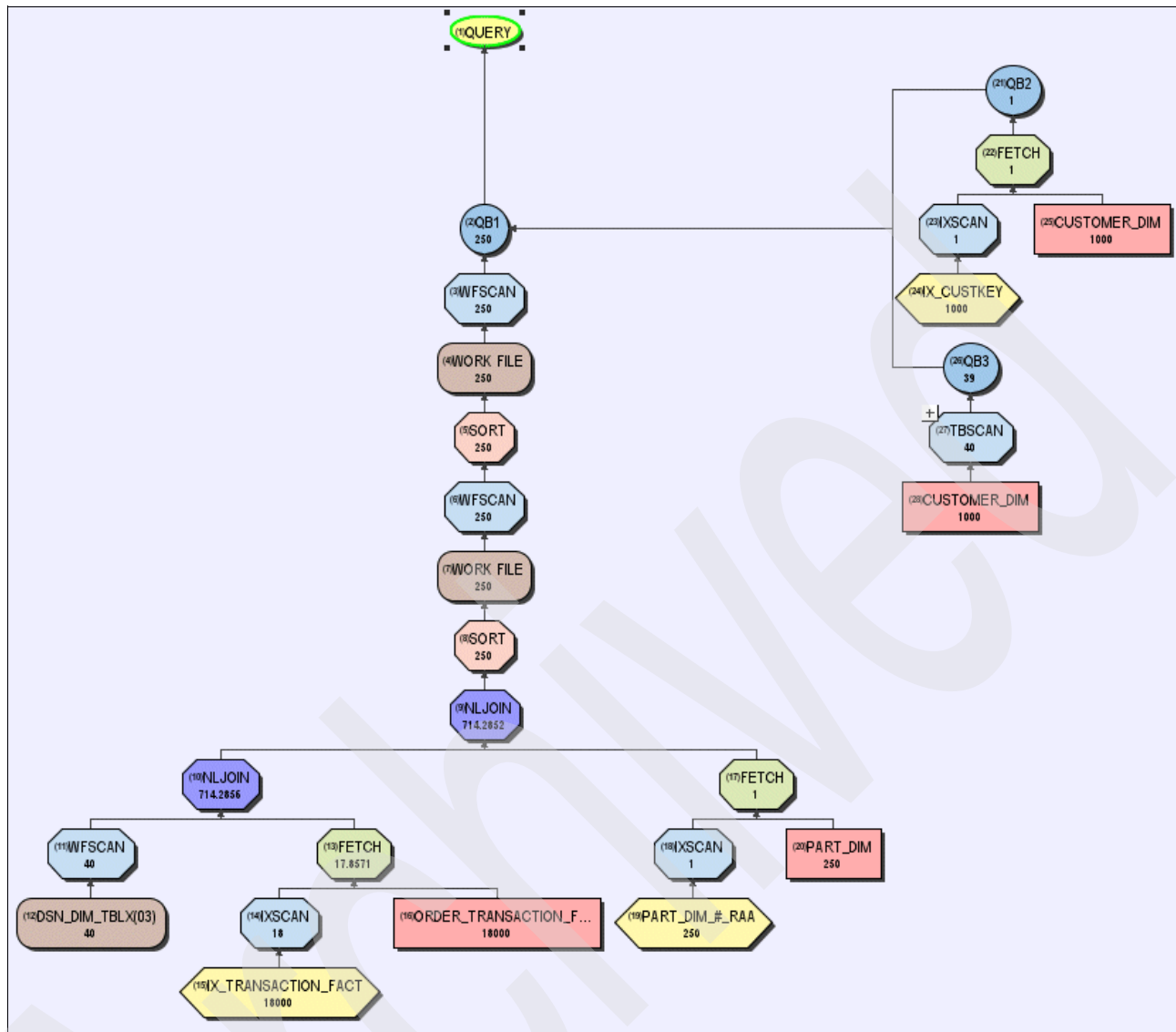


Figure 12-4 Access plan for data warehouse query in order processing application without MQT

We also get the estimated costs based on the current statistics for the tables (Figure 12-5).

Name	Value
Type	SELECT
CPU Cost (ms)	185
CPU Cost (su)	4991
Cost Category	B
Reason	TABLE CARDINALITY
Timestamp	2008-05-08 14:37:40.1

Figure 12-5 Costs of the data warehouse query without MQT

**Number of rows:** The number of rows in the tables that we used for our implementation is not representative for a customer environment. We have a few thousand rows stored in the tables, while a *real* environment might have millions of rows. The benefits of using MQTs are even greater with a larger amount of data.

In order to get to the definition of a suitable MQT, we revisit the SQL statement for the original query in Example 12-1 on page 298. It uses a where predicate to filter based on the nation name, which in turn is retrieved from a given customer key. We decide to define an MQT that contains nation name, part name, part type, and the sold quantity for this combination.

Another option is a definition with cust\_key instead of nation name. Because there are potentially many customers with the same nation but only (comparable) few nations, this approach can significantly increase the rows in the MQT. Furthermore, a lookup of the nation for a given customer key is considered to be index only access. Therefore, index only access, rather than index and data access, is much better performing.

Example 12-3 shows the final definition of our MQT. The nation column is added as an additional column, and the sub select clause to retrieve the customer's nation from the customer key is removed.

*Example 12-3 MQT definition that supports popular parts query*

---

```
CREATE TABLE POPULAR_PARTS (NATION, PART_NAME, PART_TYPE, QTY) AS
  (SELECT C.CUST_NATION_NAME, P.PART_NAME, P.PART_TYPE, SUM(O.QUANTITY) AS SUM
   FROM
     DWHDDS.ORDER_TRANSACTION_FACT O, DWHDDS.PART_DIM P, DWHDDS.CUSTOMER_DIM C
   WHERE
     O.PARTKEY_DW = P.PARTKEY_DW AND
     O.CUSTKEY_DW = C.CUSTKEY_DW
   GROUP BY CUST_NATION_NAME, PART_NAME, PART_TYPE
  )
DATA INITIALLY DEFERRED
REFRESH DEFERRED
MAINTAINED BY SYSTEM
ENABLE QUERY OPTIMIZATION
```

---

**Note:** Our objective is to leave the implementation of our order processing untouched and use the automatic query rewrite mechanism for MQTs in DB2 for z/OS.

We define an MQT that is maintained by the system and enable query optimization. Both properties are required to make the materialized query table eligible for query rewrites. MQTs support the definition of indexes on the table. Because our original query filters based on country name, we introduce the following index to support this query:

```
CREATE INDEX POP_NATION ON POPULAR_PARTS (NATION)
```

After the definition and the data is loaded into our DDS, we refresh the MQT by using the following command:

```
REFRESH TABLE POPULAR_PARTS
```

The query can be manually rewritten and used in our order processing Web application, or DB2 for z/OS can rewrite the query automatically. Example 12-4 shows the effective query.

*Example 12-4 Rewritten query using the defined MQT*

---

```
SELECT PART_NAME, PART_TYPE, QTY
FROM
  POPULAR_PARTS
WHERE
  NATION =
    (SELECT CUST_NATION_NAME FROM DWHDDS.CUSTOMER_DIM WHERE CUSTKEY_DW=211629)
FETCH FIRST 10 ROWS ONLY
```

---

The value of the CURRENT REFRESH AGE register is an important criteria for DB2 for z/OS when considering an MQT for automatic query rewrite. The initial value of CURRENT REFRESH AGE is determined by the value of field CURRENT REFRESH AGE on installation panel DSNTIP8. The default installation for the initial value of that field is 0, unless your installation has changed it to ANY by modifying the value of that field.

We verified that the value for CURRENT REFRESH AGE in our DB2 for z/OS subsystem is indeed 0 and, therefore, change it to ANY in the corresponding ZPARM.

**Important:** Use care when changing CURRENT REFRESH AGE to ANY for the entire subsystem. The benefit is that qualifying queries are automatically rewritten to leverage eligible materialized queries. However, if data in the MQTs is outdated, the query rewrite might be undesired.

In our scenario, we make sure that the MQT is refreshed whenever we change data in the underlying tables of the DDS. We also verify that no other negative impact is imposed by changing this ZPARM value.

We define our MQT with the option MAINTAINED BY SYSTEM. Therefore, we do not have to change register CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION, which is used to determine the tables that should be considered by the optimizer for query rewrites.

Another option for setting the desired registry values is to modify the application program and add additional JDBC statements. See Example 12-5.

*Example 12-5 JDBC code fragment to set the current refresh age value*

---

```
Connection con = ...;
Statement stmt = con.createStatement();
stmt.execute("SET CURRENT REFRESH AGE ANY");
stmt.close();
```

---

After the MQT is defined and we changed the ZPARM value, we again use IBM Optimization Service Center to verify whether the query is being rewritten. DB2 for z/OS now accesses our newly defined MQT. In Figure 12-6 on page 304, we see that this is the case. The part of the access graph that represents retrieving the nation name from the customer key is unchanged. Access to the fact table and the part dimension, however, is completely replaced by accessing the MQT.

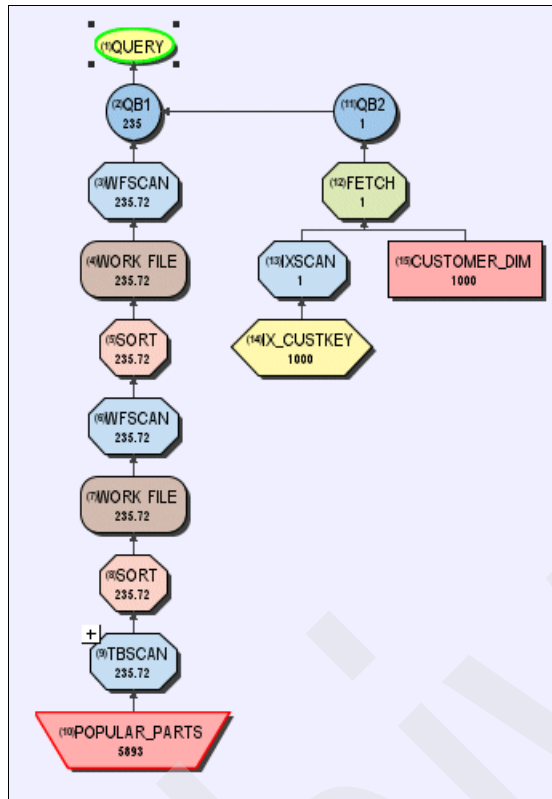


Figure 12-6 Access plan for data warehouse query in order processing application using MQT

Comparing CPU cost estimates for the previous plan (Figure 12-5 on page 301) with the one using MQTs (Figure 12-7) shows an improvement factor of 30. As usual, these are estimates, but they give an indication of the kind of improvement that can be achieved. This example shows how MQTs can significantly support an existing operational BI application, such as our order processing Web application.

Name	Value
Type	SELECT
CPU Cost (ms)	6
CPU Cost (su)	158
Cost Category	B
Reason	TABLE CARDINALITY
Timestamp	2008-05-08 14:38:27.26

Figure 12-7 Costs of the data warehouse query using MQT

With a more complex application, after analyzing expensive data warehouse queries, a set of multiple data warehouse queries can be defined to support these queries. This will likely become an iterative process with a trade-off where the cost of storing and maintaining materialized queries must be balanced against the benefit of improved CPU and response time.



## Reporting and analysis with Cognos 8 BI

In this chapter, we give an overview of our Cognos software implementation by using Cognos 8 Business Intelligence (BI) for Linux on System z. The implementation has been delivered to fulfill the requirements for the RedParts Distribution scenario, which is documented in 5.2, “Business requirements” on page 72, and to complete the enterprise data warehouse environment that was developed by using DB2 for z/OS and System z tools. Cognos 8 BI is available from Linux, UNIX, and Windows.

**Note:** For more information about Cognos 8 BI for Linux on System z, see “Cognos 8 Business Intelligence for Linux on System z” at the following address:

<http://www.ibm.com/software/data/cognos/cognos-8-business-intelligence/system-z/>

The Cognos 8 BI components that we demonstrate include Query Studio, Report Studio, Analysis Studio, PowerCubes, Transformer, and Framework Manager. Other components of Cognos 8 BI, such as Metric Studio and Event Studio, are available that we have not demonstrated.

For more information about the full range of capabilities that are available with Cognos 8 BI, visit the following Web site:

<http://www.cognos.com>

This chapter contains the following topics:

- ▶ 13.1, “Overview of Cognos 8 BI components” on page 306
- ▶ 13.2, “Installing Cognos 8 BI Server on Linux on System z” on page 308
- ▶ 13.3, “Building Cognos data models and packages” on page 335
- ▶ 13.4, “Reports with Report Studio” on page 367
- ▶ 13.5, “Ad hoc queries with Query Studio” on page 374
- ▶ 13.6, “Multidimensional analysis with Analysis Studio” on page 385

## 13.1 Overview of Cognos 8 BI components

Cognos 8 BI is a powerful suite of components that delivers a complete range of BI and performance management capabilities on a single, service-oriented architecture (SOA). These components share one common infrastructure for creating, managing, and deploying queries, reports, analyses, dashboards, scorecards, and alerts. Cognos 8 BI helps to meet an organization's BI objectives and the needs of every kind of user from simple report consumers to professional authors and developers.

In the following sections, we highlight some components and capabilities of Cognos 8 BI.

### Reporting

Reporting with Cognos 8 BI gives users the ability to author, share, and use reports that draw on data across all enterprise sources for better business decisions. Cognos 8 BI delivers all report types (ad hoc queries, simple inventory lists, managed business, bills, statement style, invoice, and business dashboards).

The Report Studio component provides the report authoring capability. Reports can include charts, cross-tabs, and lists as well as non-BI components such as images, logos, and live embedded applications that can be linked to information. Users can drag report objects into the report authoring window. The report layout is automatically rearranged correspondingly. The task-based interface offers flexibility and simplifies report authoring and modification.

Cognos 8 BI provides complete enterprise reporting coverage as it lets each group work with information as they want. Reports can be accessed and shared by the recipients in their own work language. Reporting with Cognos 8 BI helps business managers make reports, analyze, follow performance metrics, and collaborate with other users. It also helps executives get business dashboards for at-a-glance summaries of critical information and understanding of the business.

With the open data strategy provided by Cognos 8 BI, user can deliver complex reports against either relational or multidimensional sources. Reporting with dimensional data includes automatic recognition of hierarchies and automatic drill for text reports and charts.

Cognos 8 BI delivers a zero-footprint browser based easy-to-use interface for all reporting functionality and provides high usability, interaction, high scalability, and complex formatting.

Self-service reporting with Cognos 8 BI enables any user in the organization to create reports quickly and distribute information faster by reusing report objects and queries created by others.

### Analysis

Analysis Studio is a Cognos 8 BI component that delivers the capability to analyze information presented by business dimensions and measures independently from where the operational data is stored. This capability uses online analytical processing (OLAP) technologies or dimensionally modeled relational data.

Complex data analysis becomes easier through elaborate functions and features that allow sophisticated filtering and exclusion of unimportant information through business-oriented calculations. The asymmetrical analysis feature enables the integration of different rows and columns of data for an easier recognition of business issues.

The analysis capability is characterized by its ease of use, thanks to drag-and-drop techniques, and the possibility to drill down through increasing detail levels, rank, sort, forecast, and nest information. These techniques can offer a better insight into trends,

causes, and effects. The user-friendly interface, the use of graphics to analyze data relationships, and the possibility to change displays easily make multidimensional analysis simple for all users independently of their business or technical level.

## Dashboards

Business dashboards are designed to give a broad overview of how well your business is running and an easy-to-understand view of the numbers that matter most. Dashboards use visuals that include graphs, charts, maps, and other at-a-glance displays to provide critical information about processes, performance, and progress toward goals. In case of poor performance, managers and executives can issue alerts and warnings and make decisions to keep the business running smoothly. Figure 13-1 shows an example dashboard that has been built with Cognos 8 BI portlets.

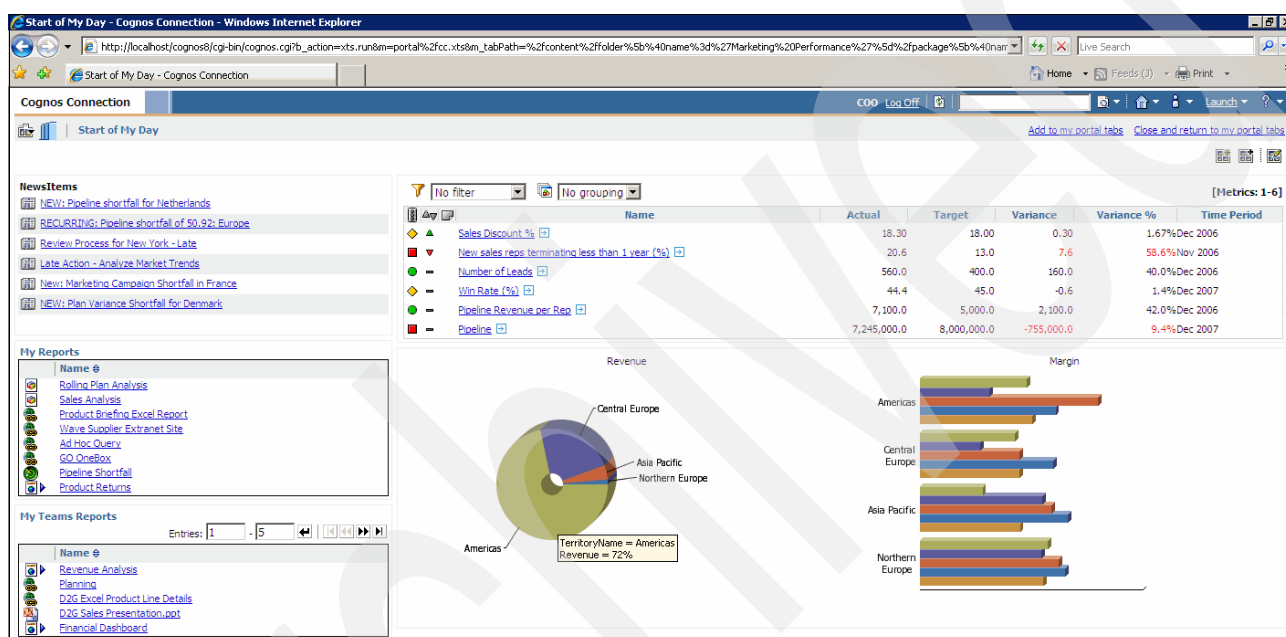


Figure 13-1 Example Cognos 8 BI dashboard portlets<sup>1</sup>

With Cognos dashboards, business users can drill into or through related reports and analysis cubes to obtain additional information or a better understanding of trends and issues. They can schedule and define criteria to receive critical information by creating personalized alerts based on data conditions. Using global dashboard filters allows for smooth orchestrated changes across multiple objects in a portal page.

Tactical dashboards deliver analysis and tracking of departmental activities, processes, and projects. They run against data marts or data warehouses. Managers and analysts can get visibility into monitored performance on a daily, weekly, or monthly basis. They can display results either as a stand-alone report or in a corporate portal.

By using strategic dashboards and scorecards, executives and senior staff can measure their progress against the associated target and strategic objectives at each level of an organization. Executives can also drill into supporting details, such as reports or analysis, to understand why a metric is performing a certain way. Scorecards perform an important role in the performance management of an organization.

Operational dashboards have the unique need for real-time updates to be displayed. Operational dashboards are provided by the Cognos Now! product line.

<sup>1</sup> Dashboard image source: Cognos Web site: <http://www.cognos.com>

## BI modeling tools

Framework Manager is a metadata modeling tool for building models that present data source information in a business perspective. From these models, a number of packages that contain business metadata can be published with appropriate security information. Published packages can then be used by many different groups of users as the source within Cognos web studios to perform their BI needs.

Transformer is the OLAP modeling component of Cognos 8 BI and is used to define and model OLAP PowerCubes. With this tool, users can define their organization reporting structures as dimensional hierarchies and levels. These reporting structures are then related to defined organization facts or measures such as “dollars spent.” The resulting model is used to create an OLAP PowerCube®, which when deployed to the Cognos server, can be used for reporting and drill analysis within the Cognos web studios.

## 13.2 Installing Cognos 8 BI Server on Linux on System z

In this section, we explain how to install the Cognos 8 BI Server components on Linux on System z.

**Note:** At the time when we wrote the book, Cognos 8 BI Server had only been announced to become available for Linux on System z, but was not actually available. We used a beta version of the product code for our book. Be aware that windows shown in this section and in the installation steps may change with the final version.

### 13.2.1 Topology overview

Figure 13-2 illustrates the technical components for Cognos BI and the platform on which they are installed.

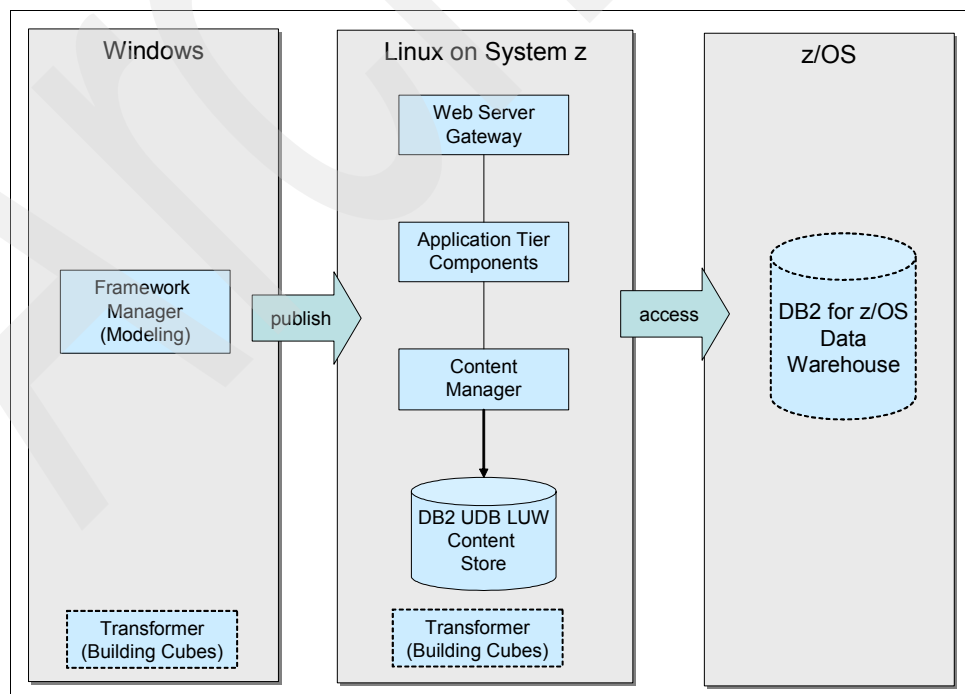


Figure 13-2 Topology overview for Cognos installation

On Windows, we install Framework Manager, the Cognos BI modeling tool to define packages that are then published to the Cognos server, running on Linux on System z. The server components include a Web server gateway, which accepts HTTP requests from (Web) clients, and an application tier layer that processes the requests. The content manager accesses the content store database, which maintains metadata, such as the published packages and stored reports.

The Cognos server components access data sources, particularly the data warehouse as in our case, that are maintained by DB2 for z/OS. For this book, we install the Cognos BI Server and the Cognos Transformer product components both on Linux on System z and Windows. We verified building PowerCubes on both platforms.

The server components, for example, the Web server gateway, can be distributed among multiple physical machines for load balancing and high availability. For our implementation, we choose to run all the Cognos server components on a single installation of Linux on System z. For this book, we installed *Cognos BI Server 8.x* along with the product versions and components listed in Table 13-1.

Table 13-1 Product components

Component	Product and version
Web Server Gateway	IBM HTTP Server 6.1
Application Tier	Tomcat 4.1.27 (default) WebSphere Application Server 6.1.0.17 (optional)
Content Store	DB2 for Linux, UNIX, and Windows Enterprise Server Edition, V9.5, FP1

We do not describe installation of IBM HTTP Server, WebSphere Application Server, or DB2 for Linux, UNIX, and Windows in this book. In the following section, we assume that these components are installed and configured properly.

**Notes:**

- ▶ We install DB2 for Linux, UNIX, and Windows Enterprise Server Edition on Linux on System z because we need the DB2 installation both for the local Content Store and for connecting to our remote DB2 for z/OS data warehouse.
- ▶ We do not use an LDAP server for a user registry and do not implement user authentication for Cognos BI Server.

## 13.2.2 System considerations and version checks

Before installing Cognos BI Server, apply the following system considerations and checks to make sure that the environment is properly setup.

**Note:** For current information about active and compatible Cognos 8 software environments, refer to the *Cognos 8 Installation and Configuration Guide* and the software environments topic at the following Web address:

<http://support.cognos.com/en/support/products/environments.html>

## Ports

Cognos BI Server requires, by default, access to three ports. You must ensure that these ports are “open”. Firewalls and other security components may prevent desktop machines from accessing certain ports on servers. Therefore, make sure that the following planned ports can be used in your environment:

- ▶ The main port is the *dispatcher port*, which by default, is set to 9300.
- ▶ You also need a *shutdown port* (9399) and a *logging port* (9362).
- ▶ Additionally, for access to the Web server, you need port 80 (or whichever port the Web server runs on) to be open as well.

To avoid conflicting port assignments, you can use the **netstat** command to list the ports that are already in use. In Example 13-1, we use **netstat -a** and check if port 9080 is in use.

*Example 13-1 Using netstat to list used ports*

---

```
cognos@lnxdwh1:~> netstat -a | grep 9080
tcp        0      0  *:9080               *:*                  LISTEN
```

---

## Free disk space

As a rule of thumb, plan for the following required free disk space. Depending on selected components, disk space requirements may vary.

- ▶ 3 GB for Cognos BI Server
- ▶ 1 GB for DB2 for Linux, UNIX, and Windows
- ▶ 3 GB for WebSphere Application Server

## Checking the DB2 for Linux, UNIX, and Windows version

To verify that you run the required DB2 version for Linux on System z, run the **db2level** command in a shell with a logon user who has the DB2 environment setup. The result should look as shown in Example 13-2. You should see that, at a minimum, version 9.5.0.1 is installed.

*Example 13-2 Using db2level to display DB2 for Linux, UNIX, and Windows version information*

---

```
db2inst1@lnxdwh1:~> db2level
DB21085I  Instance "db2inst1" uses "64" bits and DB2 code release "SQL09051"
with level identifier "03020107".
Informational tokens are "DB2 v9.5.0.1", "s080328", "MI00227", and Fix Pack
"1".
Product is installed at "/opt/IBM/db2/V9.5".
```

---

## Checking the WebSphere Application Server version

To verify the version of WebSphere Application server, run the **versionInfo.sh** script in the application server's bin directory. In Example 13-3, we show the version report of a network deployment (ND) installation. Alternatively, you can use a “base” installation.

*Example 13-3 Using versionInfo.sh to check the WebSphere Application server version*

---

```
cognos@lnxdwh1:/opt/IBM/WebSphere32/AppServer/bin> ./versionInfo.sh
WVER0010I: Copyright (c) IBM Corporation 2002, 2005; All rights reserved.
WVER0012I: VersionInfo reporter version 1.15.4.1, dated 11/23/07
```

---

-----  
IBM WebSphere Application Server Product Installation Status Report  
-----

(... lines skipped ... )

#### Installed Product

Name	IBM WebSphere Application Server - ND
Version	<b>6.1.0.17</b>
ID	ND
Build Level	cf170821.07
Build Date	5/28/08

End Installation Status Report

### Checking the Java version

It is important to verify that you run a supported 31-bit Java virtual machine (JVM™). For installation with WebSphere Application Server, you can check the JVM versions by using **java -version** as shown in Example 13-4. If you use a stand-alone Java Runtime Environment (JRE) installation, call **java -version** in the installation directory of the JRE installation that you plan to use.

*Example 13-4 Output of Java version check for the supported 31-bit version*

```
cognos@lnxdwh1:/opt/IBM/WebSphere32/AppServer/java/bin> java -version
java version "1.5.0"
Java(TM) 2 Runtime Environment, Standard Edition (build pxz31dev-20080315 (SR7))
IBM J9 VM (build 2.3, J2RE 1.5.0 IBM J9 2.3 Linux s390-31 j9vmxz3123-20080315 (JIT enabled)
J9VM - 20080314_17962_bHdSMr
JIT - 20080130_0718ifx2_r8
GC - 200802_08)
JCL - 20080314
```

Example 13-5 shows the check output when the *unsupported 64-bit version* of the Java Developer Kit (JDK™) is detected. In this case, install the 31-bit version of WebSphere Application Server or the JRE.

*Example 13-5 Output of Java version check for the unsupported 64-bit version*

```
cognos@lnxdwh1:/opt/IBM/WebSphere64/AppServer/java/bin> java -version
java version "1.5.0"
Java(TM) 2 Runtime Environment, Standard Edition (build pxz64devifx-20071025 (SR6b))
IBM J9 VM (build 2.3, J2RE 1.5.0 IBM J9 2.3 Linux s390x-64 j9vmxz6423-20071007 (JIT
enabled)
J9VM - 20071004_14218_BHdSMr
JIT - 20070820_1846ifx1_r8
GC - 200708_10)
JCL - 20071025
```

### 13.2.3 Preparation for installation

Because of the several environment variables to be set properly for installing, configuring, and running Cognos BI Server, we created a dedicated user ID that owns the physical files of the Cognos BI Server installation and that is also used for Cognos administration tasks. We use YaST (the Linux configuration tool) to define a new user named *cognos* and a group named *cognosgrp* (Figure 13-3 on page 312).

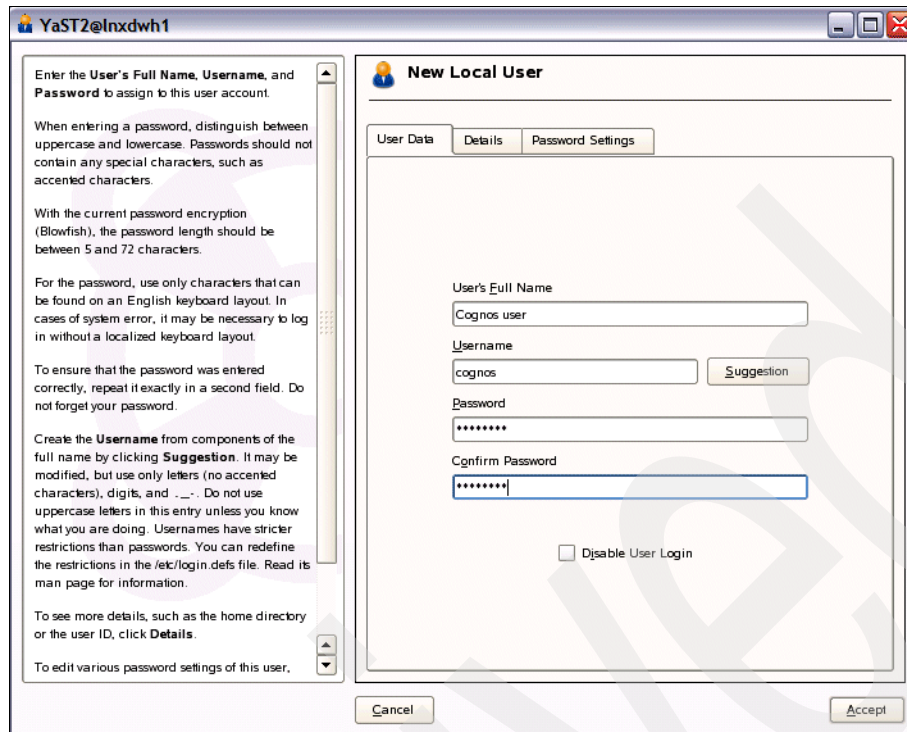


Figure 13-3 Creating a dedicated user ID for Cognos installation and administration

After the user is created in the system, you edit `.profile` (or the corresponding name for the shell you are using) and add the section shown in Example 13-6. Assuming that your DB2 installation created an instance and corresponding user `db2inst1` for it, running the **db2profile** script prepares the environment for the `cognos` user to access DB2 properly.

*Example 13-6 Additional .profile settings for the Cognos user*

```
# The following lines give access to the IBM DB2 instance
if [ -f /home/db2inst1/sqllib/db2profile ]; then
    . /home/db2inst1/sqllib/db2profile
fi
export LD_LIBRARY_PATH=/home/db2inst1/sqllib/lib32:$LD_LIBRARY_PATH
```

**Note:** The Cognos BI Server application code is compiled with 32-bit libraries and also needs access to the DB2 for Linux, UNIX, and Windows 32-bit libraries. Therefore, the `LD_LIBRARY_PATH` variable must include the DB2 32-bit libraries *before* the 64-bit libraries.

Invoking **db2profile** in Example 13-6 also sets the `CLASSPATH` environment variable for the `cognos` user to include the Java archive (JAR) files for the DB2 JDBC driver. If you do not want to run **db2profile**, you must set `CLASSPATH` explicitly as shown in Example 13-7.

*Example 13-7 Sample CLASSPATH setting for including DB2 JDBC driver JAR files*

```
export CLASSPATH =
/home/db2inst1/sqllib/java/db2java.zip:/home/db2inst1/sqllib/java/db2jcc.jar:/home/db2inst1
/sqllib/function:/home/db2inst1/sqllib/java/db2jcc_license_cu.jar:$CLASSPATH
```



## 13.2.4 Installing Cognos BI Server components

To install Cognos BI Server on Linux on System z:

1. Log in as user cognos and run the **issetup** program in your installation image. A GUI-based installation wizard opens and requires that you have an X Server running on your machine.
2. In the welcome panel (Figure 13-4), select **English** and click **Next**.

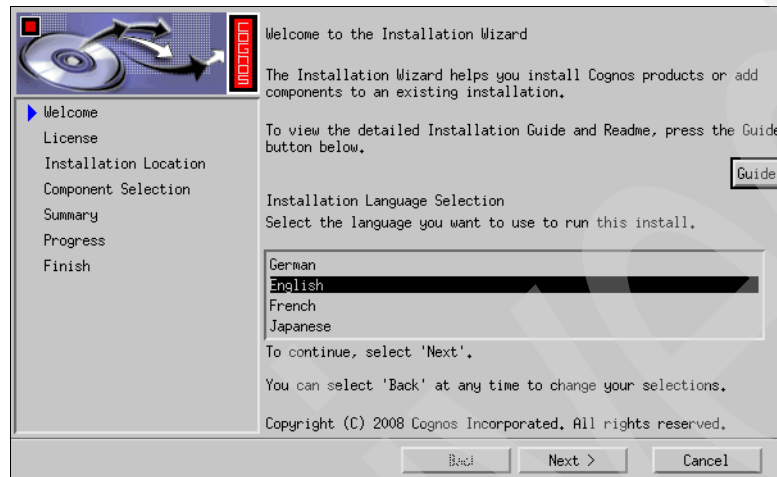


Figure 13-4 Welcome panel of the installation wizard for Cognos BI Server

3. In the License Agreement panel (Figure 13-5), click **I accept**. Since we used the beta version of the product, the example shows the corresponding license and conditions from that version. Click **Next**.

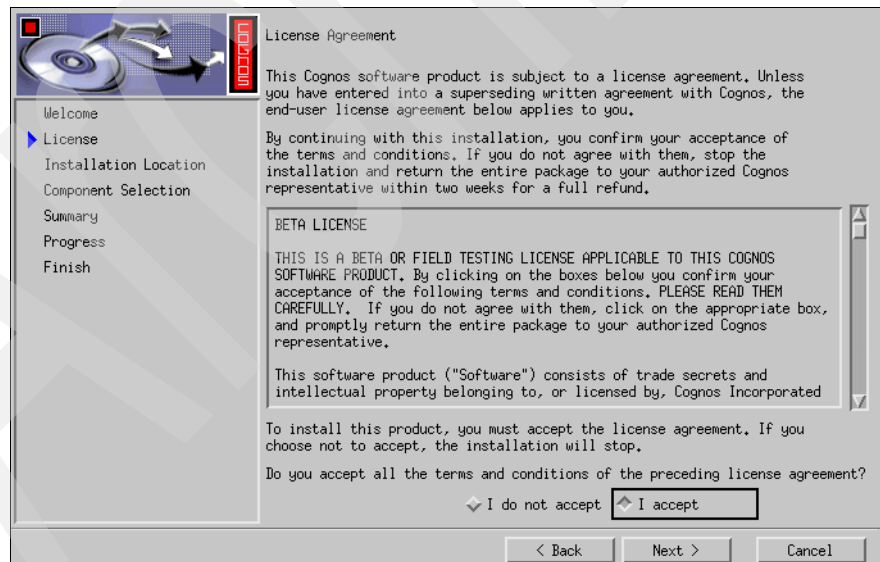


Figure 13-5 License agreement for Cognos BI Server

4. In the Installation Location panel (Figure 13-6), specify the file system location where the product files should be allocated. We select **/opt/IBM/cognos/c8**. The panel gives an overview of the available and required disk space. Keep in mind that the installation process also needs space in the temporary directory. Click **Next**.

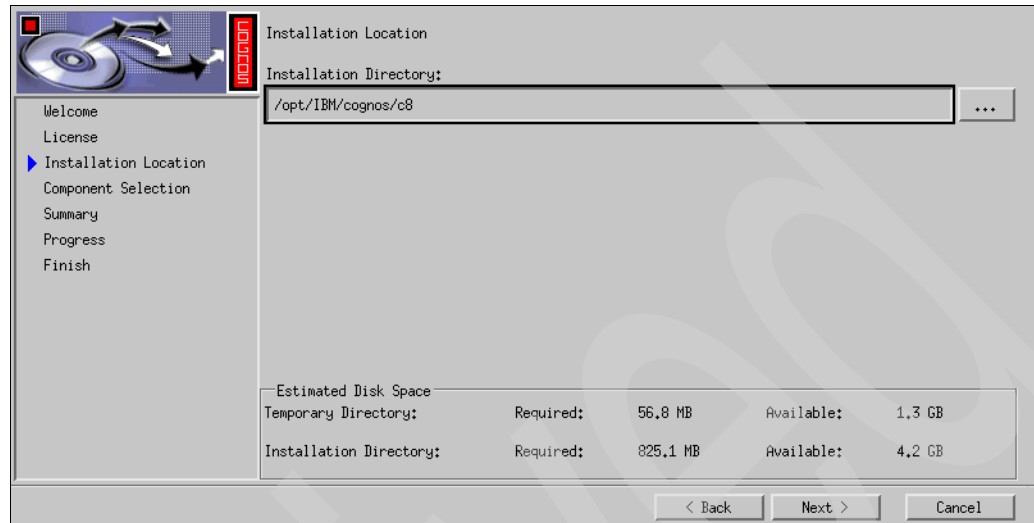


Figure 13-6 Installation location for Cognos BI Server

5. If a window opens with a message indicating the directory **/opt/IBM/cognos/c8** does not exist yet, click **Yes** to confirm that the directory should be created. You see this window if you did not install Cognos BI Server into the same location before.
6. In the Component Selection panel (Figure 13-7), specify the components that you want to install. We install all the components on our single Linux on System z. See the topology discussion in 13.2.1, “Topology overview” on page 308.

Because we plan to use DB2 for Linux, UNIX, and Windows for the content store, we clear the option **Cognos Content Database**. (Selecting this option results in the creation of an Apache Derby database for the content store.)

Click **Next**.

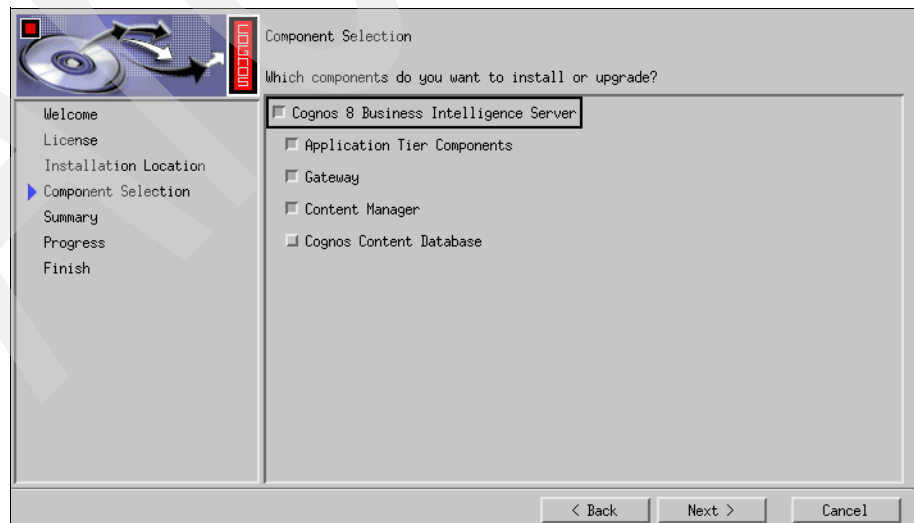


Figure 13-7 Component selection for Cognos BI Server

7. In the Installation Summary panel (Figure 13-8) review the selections for the installation and the location of the installation log file. Click **Next** to start the installation.

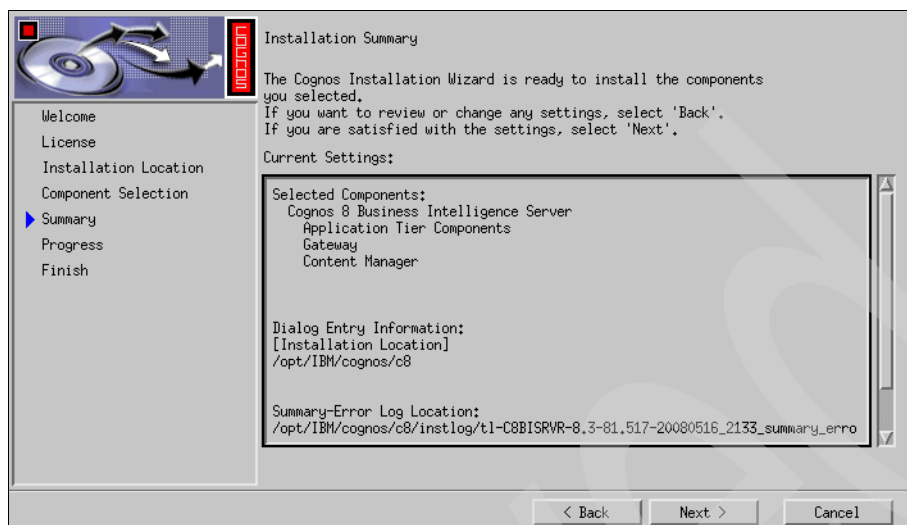


Figure 13-8 Summary before installation of Cognos BI Server

8. After the installation completed, in the Finish panel (Figure 13-9), notice that the message about the successful transfer of the files. You also see the location of the installation log files with more information. Clear the **Start Cognos Configuration** option and click **Finish**. Although we click Finish, we must perform some additional steps before we can run the configuration.

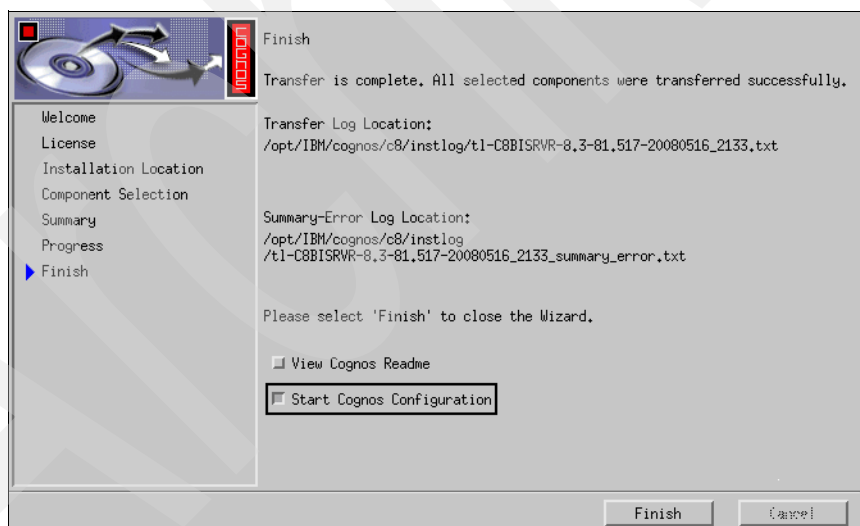


Figure 13-9 Completion notice after Cognos BI Server installation

We must revisit the environment settings for our cognos user ID again to extend the LD\_LIBRARY\_PATH. Edit ~cognos/.profile and adapt the export statement as shown in Example 13-8. We added both the cgi-bin and the bin directories, which are required to run the HTTP server and the Cognos configuration.

*Example 13-8 Adding Cognos directories to LD\_LIBRARY\_PATH*

---

```
export LD_LIBRARY_PATH=
/home/db2inst1/sqllib/lib32:$LD_LIBRARY_PATH:/opt/IBM/cognos/c8/cgi-bin:/opt/IBM/cognos/c8/
bin
```

---

Because we use the Cognos transformer to build PowerCubes later in this chapter, we install the required components from the respective installation image as well. The procedure is similar to the previous steps for the Cognos BI Server components, except that the component selection shows just the transformer component (Figure 13-10).

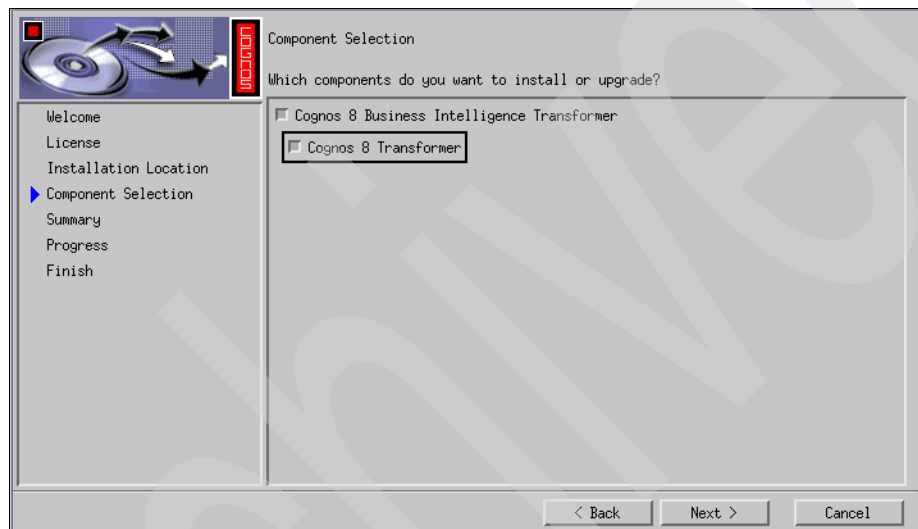


Figure 13-10 Installing Cognos 8 Transformer

## 13.2.5 Configuring the IBM HTTP Server

For a detailed description about how to configure the IBM HTTP Server, refer to the “Configure the Gateway for Cognos Apache Web Server Module” section in the *Cognos BI Server Installation Guide*.

In our environment, we install IBM HTTP Server 6.1 in /opt/IBM/HTTPServer. The HTTP server configuration file is in /opt/IBM/HTTPServer/httpd.conf. Edit the configuration file and make the changes as shown in Example 13-9.

*Example 13-9 Modifications in httpd.conf*

---

```
# For Section 1 (global configuration)

# 1. Change the listener port from 80 (default) to 9380
# (replace the port number 80 with 9380)
Listen 9380

# 2. Load the cognos module from the installation directory
# (add this line to the end of the LoadModule configuration lines)
LoadModule cognos_module /opt/IBM/cognos/c8/cgi-bin/mod2_cognos.so
```

```
# For Section 2 ('main' server configuration)

# 3. Change name of user and group for httpd
# (replace nobody/nobody with cognos/cognosgrp)
User cognos
Group cognosgrp

# 4. Cognos alias definition
# (add section after other alias and directory definitions)
ScriptAlias /cognos8/cgi-bin /opt/IBM/cognos/c8/cgi-bin
Alias /cognos8 /opt/IBM/cognos/c8/webcontent
<Directory "/opt/IBM/cognos/c8/webcontent">
    Options Indexes MultiViews
</Directory>

# 5. Cognos location definition
# (add after other location definitions in the file
# and make sure that this definition comes after the alias
# definition in 4.)
<Location /cognos8/cgi-bin/cognos_module>
    SetHandler cognos-handler
</Location>
<Location /cognos8/cgi-bin/diag_cognos_module>
    SetHandler cognos-handler
</Location>
<IfModule mod_cognos.c>
    CGIBinDir "/opt/IBM/cognos/c8/cgi-bin"
</IfModule>
```

---

You are now ready to start the HTTP daemon by entering the following command:

```
/opt/IBM/HTTPServer/bin/apachectl -k start
```

This command starts the configured number of httpd daemons. You can use **ps** to check if all of the daemons came up successfully. See Example 13-10.

*Example 13-10 HTTP daemons for the Cognos BI Web server gateway*

---

```
lnxdwh1:/opt/IBM/HTTPServer # ps -ef | grep httpd
root      7623      1  0 09:06 ?        00:00:00 bin/httpd -d /opt/IBM/HTTPServer
cognos    7624    7623  0 09:06 ?        00:00:00 bin/httpd -d /opt/IBM/HTTPServer
cognos    7625    7623  0 09:06 ?        00:00:00 bin/httpd -d /opt/IBM/HTTPServer
cognos    7628    7623  0 09:06 ?        00:00:00 bin/httpd -d /opt/IBM/HTTPServer
```

---

To stop the HTTP daemon(s) again, enter the following command:

```
/opt/IBM/HTTPServer/bin/apachectl -k stop
```

## 13.2.6 Configuring and starting the Cognos BI Server

Configuring the Cognos BI Server includes setting for the content store, running the configuration tool, adapting the configuration settings, and starting the server.

## Preparing DB2 for Linux, UNIX, and Windows for the content store

**Note:** In this section, we assume that DB2 for Linux, UNIX, and Windows is already installed and configured properly.

The Cognos installation documentation offers several ways to set up the content store database. We choose to modify and run the script for DB2 for Linux, UNIX, and Windows, which is provided in `/opt/IBM/cognos/c8/C8SE/C8DB2.sh`. Edit the user ID and password settings in the file and run it from the shell. By default, a database named C8CM is created with all the required settings (Example 13-11).

Refer to the *Cognos 8 BI Installation and Configuration Guide*, which you can find at the following Web address, for the recommended process to create your content store:

<http://support.cognos.com/index.html>

### Example 13-11 Running the C8DB2.sh script to create a content store database

```
cognos@lnxdwh1:/opt/IBM/cognos/c8/C8SE> ./C8DB2.sh
SQL1026N The database manager is already active.
DB20000I The CREATE DATABASE command completed successfully.
DB20000I The CHANGE DATABASE COMMENT command completed successfully.

Database Connection Information

Database server          = DB2/LINUXZ64 9.5.1
SQL authorization ID    = COGNOS
Local database alias    = C8CM

DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
DB20000I The SQL command completed successfully.
... (lines skipped) ...
DB20000I The SQL command completed successfully.
```

## Configuration settings for JRE

To run the Cognos BI Server configuration tool (**cogconfig**), you need a suitable JRE. We installed the IBM 32-bit 1.5 JRE in `/opt/IBM/ibm-java2-s390-50`. Refer to “Checking the Java version” on page 311 for a description of and checks for supported JREs. Running an unsupported 64-bit JRE with **cogconfig** can result in an error message such as the example in Figure 13-11.

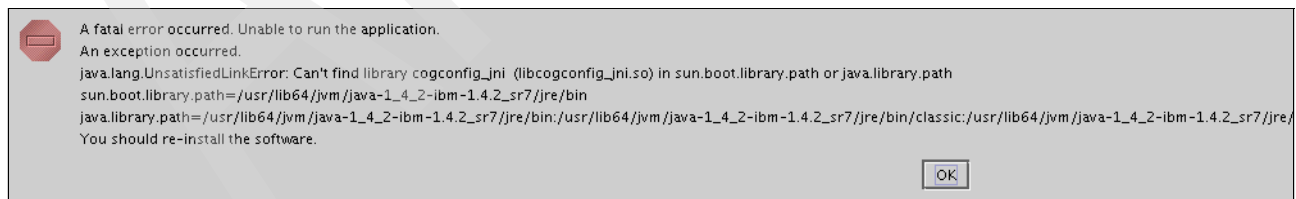


Figure 13-11 Error message with incompatible JRE

You probably have other Java environments on your system installed already. To use the right JRE for Cognos configuration, modify `~cognos/.profile` and add the section shown in Example 13-12.

*Example 13-12 Setting the JRE in .profile*

---

```
export JAVA_HOME=/opt/IBM/ibm-java2-s390-50/jre
export PATH=/opt/IBM/ibm-java2-s390-50/jre/bin:$PATH
```

---

As described in the Cognos BI Server documentation, copy the `bcprov-jdknn-nnn.jar` file from the `c8_location/bin/jre/version/lib/ext` directory to the `Java_location/jre/lib/ext` directory. In our environment, we run the following command:

```
cp /opt/IBM/cognos/c8/bin/jre/1.5.0/lib/ext/bcprov-jdk14-134.jar
/opt/IBM/ibm-java2-s390-50/jre/lib/ext
```

### **Configuration settings for DB2 for Linux, UNIX, and Windows**

Cognos BI Server needs access to the DB2 JDBC driver in two places: in the context of the application tier components and when running the configuration tool (**cogconfig**). Our DB2 installation is in `/opt/IBM/db2/V9.5`, and we use the JDBC driver that ships as `db2java.zip`.

To make the JDBC driver available for the application tier component running Tomcat as the servlet container, copy and rename the driver with the following command:

```
cp /opt/IBM/db2/V9.5/java/db2java.zip
/opt/IBM/cognos/c8/webapps/p2pd/WEB-INF/lib/db2java.jar
```

The Cognos configuration tool runs in its own shell and resets the `LD_LIBRARY_PATH` environment variable. Therefore, the JDBC driver is not accessible by default. You can either change the script that starts **cogconfig** or add the JDBC driver to the bin directory of the Cognos installation. We to create a soft link in the bin directory as follows:

```
ln -s /opt/IBM/db2/V9.5/java/db2java.zip /opt/IBM/cognos/c8/bin/db2java.jar
```

## Running the Cognos configuration tool (cogconfig)

Launch `/opt/IBM/cognos/c8/bin/cogconfig.sh` as user `cognos` from the Linux shell. This opens a GUI-based configuration panel and requires again X Windows server running on your machine. To complete the configuration:

1. In the Explorer pane of the Cognos Configuration window (Figure 13-12), select **Local Configuration** → **Data Access** → **Content Manager**. Right-click the existing configuration named **Content Store** and select **Delete** to remove the configuration for an (non-existing) SQL Server® database.

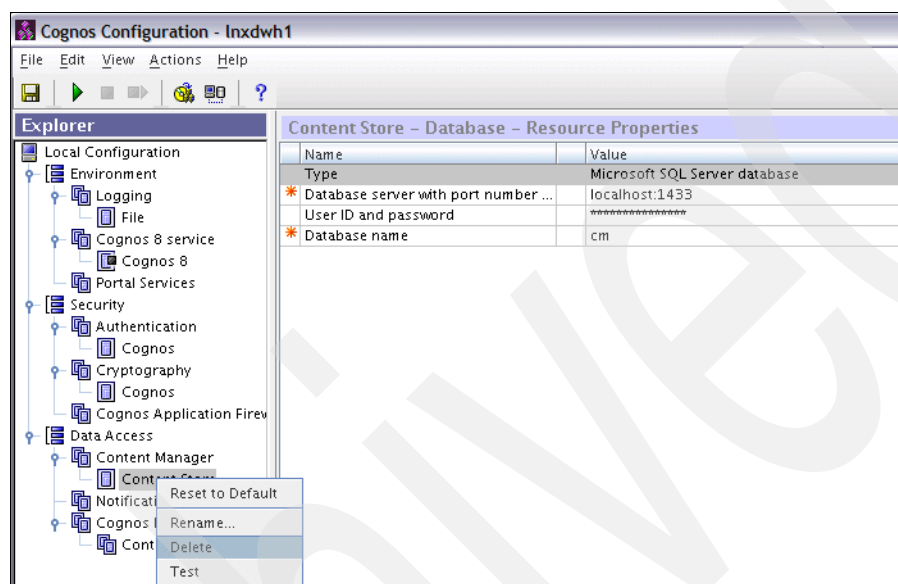


Figure 13-12 Deleting an existing setting for the content store

2. Right-click **Content Store** again and select **New resource** → **Database** (Figure 13-13).

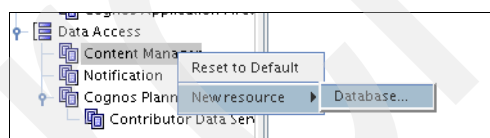


Figure 13-13 Creating a new database for the content store

3. In the window that opens, type **C8CM** as the new resource name and select **DB2 Database** as the database type. Click **OK**.



4. Add the database name (C8CM) and user ID (cognos) and password for access to the DB2 database. Then right-click the new entry for the content store and select **Test** (Figure 13-14).

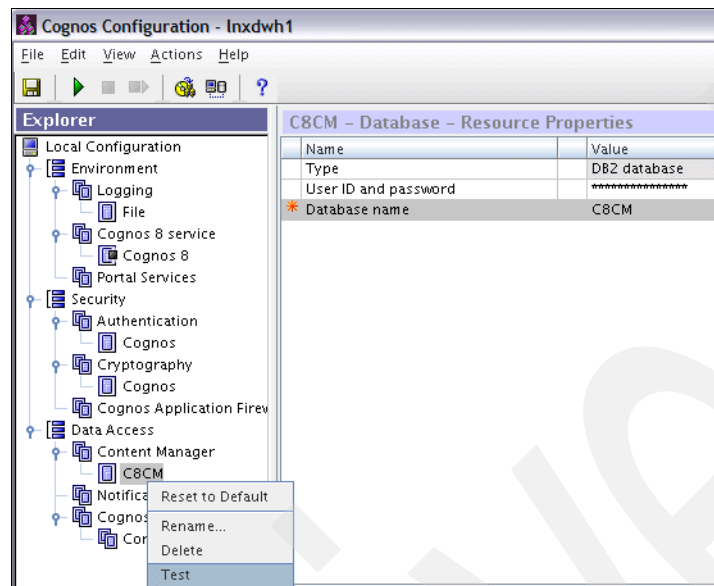


Figure 13-14 Configuring and testing the DB2 database for the Cognos content store

With the first connection to a new content store, Cognos creates the required tables and also generates cryptographic information, stored in the content store.

5. In the message window (Figure 13-15), click **Close** to continue.

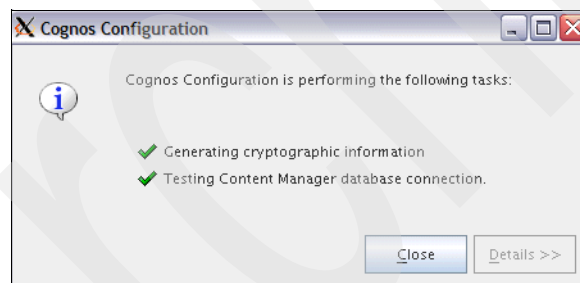


Figure 13-15 Operations on first touch for the Cognos content store

6. Select **Local Configuration** → **Environment** in the configuration explorer window (Figure 13-16) to modify the URI settings for the Web gateway. Change the port number in the definition for the Gateway URI and for the Controller URI for Gateway to 9380 (as configured in 13.2.5, “Configuring the IBM HTTP Server” on page 316).

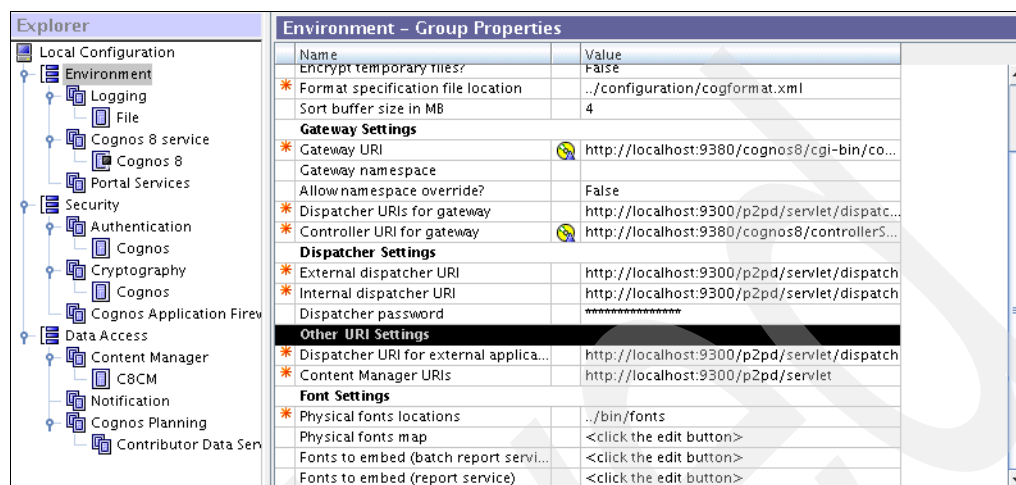


Figure 13-16 Setting port numbers for the Web server gateway

7. Start the Cognos BI Server. Click the green triangle in the configuration toolbar to start the required services.
8. In the window (Figure 13-17) that opens, which inform you about the progress, confirm the information to continue a successful startup of the server.

We did not use and configure a mail server connection. In this case, you might see a warning message.

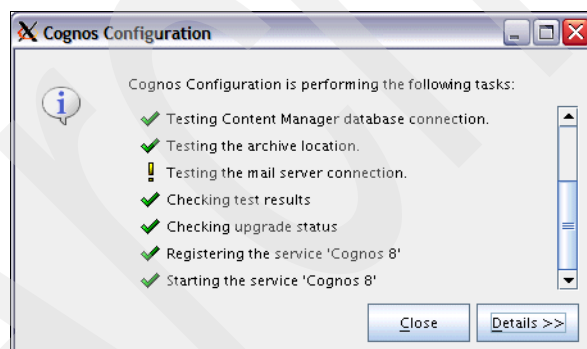


Figure 13-17 Starting the Cognos BI Server

The Cognos BI Server is now up and running. In this environment, you can direct your Web browser to the following URL to access the initial page of the Web portal:

<http://1nxdwh1.boeb1ingen.de.ibm.com:9380/cognos8/>

### 13.2.7 Defining DB2 for z/OS data sources

For access to our data warehouse data, the data sources to our DB2 for z/OS subsystems DHWD911 and DWHD912 must be configured.

We catalogued the remote DB2 for z/OS databases with the DB2 commands shown in Example 13-13. We use the HiperSocket connection to the z/OS system. If you already catalogued the DB2 for z/OS databases in the context of the Information Server installation, you do not have to perform this step again. In our environment, Cognos BI Server runs on a different Linux on System z.

*Example 13-13 Cataloguing remote DB2 for z/OS databases*

```
catalog tcpip node dwhd911 remote 10.10.10.2 server 5911 ostype mvs;  
catalog database dwhd911 as dwhd911 at node dwhd911 authentication dcs;  
catalog dcs database dwhd911 as dwhd911 parms ',,INTERRUPT_ENABLED';  
  
catalog tcpip node dwhd912 remote 10.10.10.2 server 5912 ostype mvs;  
catalog database dwhd912 as dwhd912 at node dwhd912 authentication dcs;  
catalog dcs database dwhd912 as dwhd912 parms ',,INTERRUPT_ENABLED';
```

Open the Cognos Administration panel in the Cognos Web portal and follow these steps:

1. Select **Configuration** → **Data Source Connections** and click the icon at the top of the list to create a new data source definition (Figure 13-18).

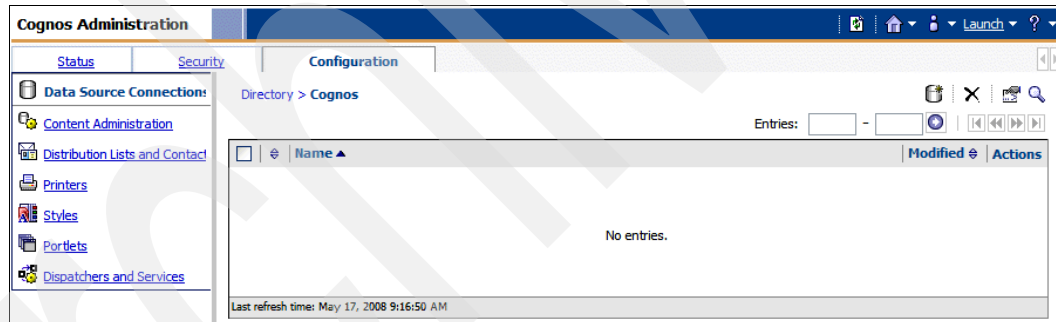


Figure 13-18 Data source connections in Cognos Administration

2. In the Specify a name and description panel (Figure 13-19), name the new data source DWHD912 and click **Next**.



Figure 13-19 Naming a new data source

3. In the Specify the connection panel (Figure 13-20), for the database type, select **DB2**. This is the place where OLAP data sources, such as a Cognos PowerCube, can also be selected. Click **Next**.

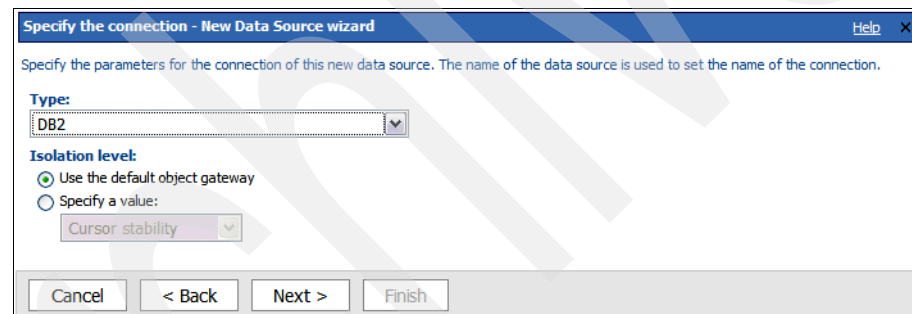


Figure 13-20 Specifying the type for the new data source

4. In the next panel (Figure 13-21 on page 325), for DB2 database name, type DWHD912, which is the name of the previously catalogued database in the local DB2 installation. We also provide a signon with user ID and password to connect to the DB2 database. These credentials are then always used for access. You do not have to enter a user ID and password for each request.

**Attention:** Depending on the following settings, by creating a signon user ID and password, you may expose access to data in DB2 for z/OS in an uncontrolled way. For our exercise, this approach might be appropriate, but in a production environment, you may want to implement more restrictive access control.

Click **Test the connection** to verify the parameters.

**DB2 database name:**  
DWHD912

**DB2 connect string:**

**Collation sequence:**

☐ Open asynchronously

---

**Timeouts**  
Specify the time in seconds, in which you want the database to connect or wait for your reply before timing out.

**Connect time:**  
0

**Reply time:**  
0

---

**Signon**  
Select whether a user ID and password is required in the connection string and, if so, whether to create a signon.

☒ User ID

☒ Password

☒ Create a signon that the Everyone group can use:

**User ID:**  
FNEUMAN

**Password:**  
\*\*\*\*\*

**Confirm password:**  
\*\*\*\*\*

---

**Testing**  
[Test the connection...](#)

Cancel < Back Next > Finish

Figure 13-21 Database name and signon parameters for the data source

- In the View the results panel (Figure 13-22), which indicates that the connection to DB2 for z/OS can be established, click **Close**. Finish the definition of the new data source.

View the results - Test the connection Help X

Name	Status	Message
...> http://lnxdwh1:9300/p2pd	Succeeded	

Close

Figure 13-22 Testing the connection to the new data source

- Repeat the definition for DWHD911 by repeating step 1 on page 323 through step 5. Figure 13-23 shows the data source list.

Directory > Cognos

Entries: 1 - 2

Name	Modified	Actions
DWHD911	May 17, 2008 9:21:50 AM	More...
DWHD912	May 17, 2008 9:20:57 AM	More...

Last refresh time: May 17, 2008 9:21:53 AM

Figure 13-23 Data source list with DWHD911 and DWHD912

## 13.2.8 Configuring Cognos with WebSphere Application Server

In the previous sections, we describe the installation and configuration of the Cognos BI Server components using Tomcat as the servlet container for the application tier (“dispatcher”). In this section, we show how we change the existing installation to use WebSphere Application Server instead.

For this book, we use WebSphere Application Server 6.1.0.17 and assume that the application server is installed and configured properly. Because we run multiple WebSphere Application Server versions on our Linux on System z, the one we use for Cognos is located in a non-default path /opt/IBM/WebSphere32. It runs under the cognos user ID that we defined earlier in this chapter.

For a description of how to check versions of an installed WebSphere Application Server and JREs, refer to “Checking the WebSphere Application Server version” on page 310.

### Change settings for the JDK

In the installation with Tomcat as the dispatcher component, we use a stand-alone IBM 31-bit JDK and set the environment for the cognos user accordingly. For the installation with WebSphere Application Server, we now use the JDK that ships with the application server and modify the settings in the cognos user .profile according to Example 13-14.

*Example 13-14 Profile settings for user cognos*

---

```
export JAVA_HOME=/opt/IBM/WebSphere32/AppServer/java
export PATH=/opt/IBM/WebSphere32/AppServer/java/jre/bin:$PATH
```

---

The Java version is reported as in shown Example 13-15. Again make sure that you are using the 31- or 32-bit JDK and WebSphere Application Server version. The highlighted number 31 in Example 13-15 indicates the correct version. Also double check that you run at least SR7 of the JDK.

*Example 13-15 Java version of the JDK that ships with WebSphere Application Server*

---

```
cognos@lnxdwh1:~> java -version
java version "1.5.0"
Java(TM) 2 Runtime Environment, Standard Edition (build pxz31dev-20080315 (SR7))
IBM J9 VM (build 2.3, J2RE 1.5.0 IBM J9 2.3 Linux s390-31 j9vmxz3123-20080315 (JIT enabled)
J9VM - 20080314_17962_bHdSMr
JIT - 20080130_0718ifx2_r8
GC - 200802_08)
JCL - 20080314
```

---

As with the stand-alone JDK, the bcprov-jdk14-134.jar file must be copied to the JRE location:

```
cp /opt/IBM/cognos/c8/bin/jre/1.5.0/lib/ext/bcprov-jdk14-134.jar
/opt/IBM/WebSphere32/AppServer/java/jre//lib/ext
```

**Note:** According to the *Cognos 8 Business Intelligence - Installation and Configuration Guide*, you must save encryption keys and data in the content store before you move to a new JDK. However, the version that we used turned out to be the same one that we used before with the Tomcat-based version. Therefore, our environment worked without regenerating encryption keys. If you plan to use a (completely) different JDK, check the documentation for the required additional steps.

## Creating and configuring a WebSphere Application Server profile

We recommend that you run the Cognos BI Server applications in a dedicated WebSphere Application Server profile. Because we run our application server instance just for Cognos BI Server, we use the default profile AppSrv01 that was created during installation of the application server. However, if you decide to use a new profile, you can use a script that ships with Cognos. Example 13-16 shows how it works. Or you can use the **manageprofile** tool from WebSphere Application Server directly to create a new profile.

---

### *Example 13-16 Running the create\_profile script to create a new WebSphere Application Server profile*

---

```
lnxdwh1:/opt/IBM/cognos/c8/C8SE # ./create_profile.sh cognos8
```

Where is IBM WebSphere installed to?

Input a directory, or hit enter to accept /opt/IBM/WebSphere:

/opt/IBM/WebSphere32

Using /opt/IBM/WebSphere32 as IBM WebSphere location.

Generating new portdef.props file.

Input 'y' to review or edit the generated portdef.props file,  
or anything else to proceed:

Executing the following command:

```
/opt/IBM/WebSphere32/AppServer/bin/manageprofiles.sh -create -profileName cognos8  
-profilePath /opt/IBM/WebSphere32/AppServer/profiles/cognos8 -templatePath  
/opt/IBM/WebSphere32/AppServer/profileTemplates/default/ -portsFile  
/tmp/create_profile.sh.26580/portdef.props
```

manageprofiles.sh completed successfully.

To start the cognos8 server profile, run the following command:

```
/opt/IBM/WebSphere32/AppServer/profiles/cognos8/bin/startServer.sh server1
```

When the server has started, it can be tested via the URL:

<http://lnxdwh1:7753/admin>

---

If you create a new profile with the **create\_profile** script, ports for access to the administrative console and for enterprise application virtual host are randomly assigned and stored in a properties files. You can use the link displayed at the end of the profile creation script (<http://lnxdwh1:7753/admin>) to access the administrative console and determine which port is assigned for enterprise applications. We need this information later to set the dispatcher settings in the Cognos configuration panel. For our setup with the default profile, we have (default) host port 9080 for applications.

Now, start the WebSphere Application Server as shown in Example 13-17. If you do not use the default profile, go to the profile's bin directory and start the server from there.

---

### *Example 13-17 Starting the server instance*

---

```
cognos@lnxdwh1:/opt/IBM/WebSphere32/AppServer/bin> ./startServer.sh server1
```

ADMU0116I: Tool information is being logged in file

/opt/IBM/WebSphere32/AppServer/profiles/AppSrv01/logs/server1/startServer.log

ADMU0128I: Starting tool with the AppSrv01 profile

ADMU3100I: Reading configuration for server: server1

ADMU3200I: Server launched. Waiting for initialization status.

ADMU3000I: Server server1 open for e-business; process id is 14024

---

If the user ID that is running the application server's JVM does not have a setting for LD\_LIBRARY\_PATH pointing to the Cognos libraries, you must perform the following steps. In any case, make sure that the user ID that is running the application server can access the files in the Cognos installation directory.

1. Open the WebSphere Application Server administration console in a Web browser. In our case, we type the following URL:  
`http://1nxdwh1:9060/ibm/console/`
2. Select any name for the user ID. Because the application server profile is not configured by using security, no user credential checks take place and the name is just used to separate concurrent changes to the configuration repository.
3. Expand **Servers** → **Application Servers**. In the right panel, select **server 1**.
4. Select **Java and Process Management**.
5. In the next pane, select **Process Definition**.
6. In the next pane, click **Environment Entries**.
7. In the Application servers panel (Figure 13-24), click **New**.

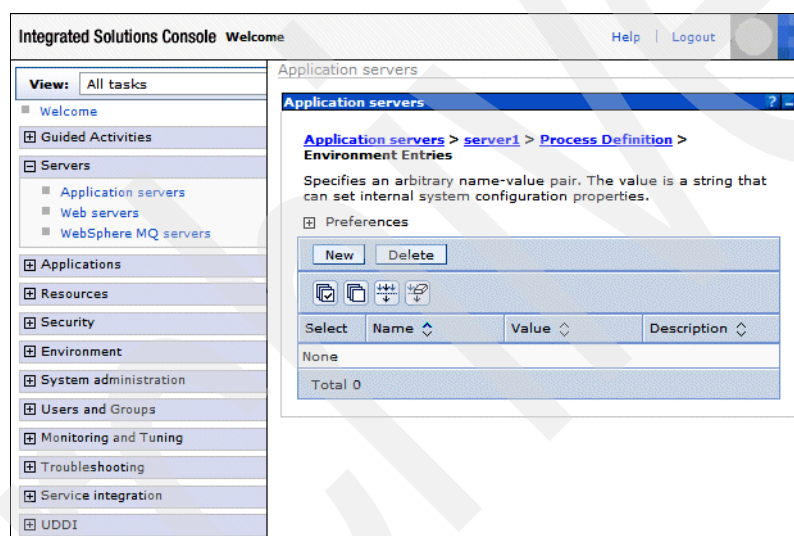


Figure 13-24 Navigating to JVM settings for the application server



- In the next panel (Figure 13-25), add a new entry with the name `LD_LIBRARY_PATH` and the value of `/opt/IBM/cognos/c8/bin`. Click **OK**.

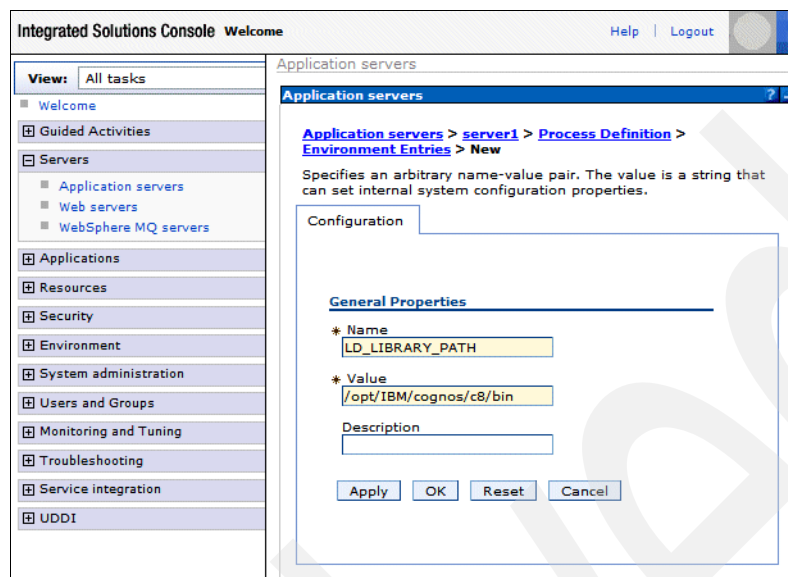


Figure 13-25 Adding `LD_LIBRARY_PATH` environment settings

- Click **Save directly to the master configuration**.

You might want to revisit other JVM settings at this point as well. Particularly the maximum JVM heap size in the Java Virtual Machine panel, under Process Definition, can be adapted, depending on the expected workload.

## Building the application files

The code for the Cognos dispatcher is installed as an enterprise application in WebSphere Application Server. Perform the following steps to create the corresponding EAR (enterprise application archive) file.

- Start the Cognos configuration with `c8location/bin/cogconfig.sh`.
- Select **Actions** → **Build Application Files** from the menu (Figure 13-26).

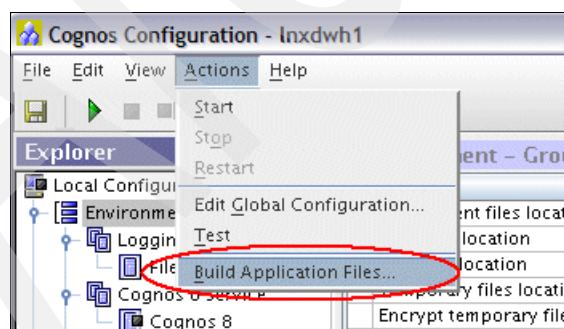


Figure 13-26 Starting to build application files for deployment in `cogconfig`

3. In the Select the application to build panel (Figure 13-27), select **Cognos 8** and **Include static files from the Web content folder** in the EAR file as well. Because we continue using the IBM HTTP Server for the gateway, we do not need the servlet gateway and therefore do not select the check boxes. For Application Server Type, select **Other**. Click **Next**.

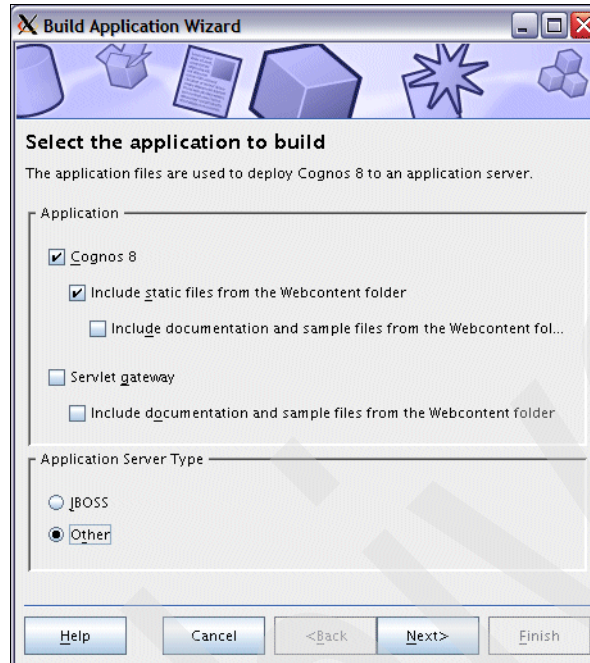


Figure 13-27 Selecting the applications to build

4. In the Specify Cognos 8 application files panel (Figure 13-28), leave the default name for the resulting EAR file as `p2pd.ear` and click **Next**.

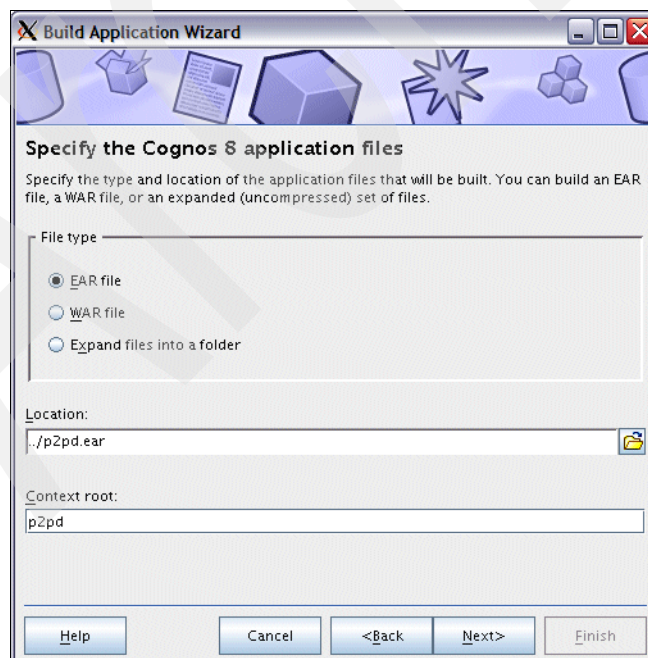


Figure 13-28 Settings for the `p2pd` EAR file

- When you see the panel that shows the successful creation of the EAR file (Figure 13-29), write down the location of the file. In this case, the location is `/opt/IBM/cognos/c8/p2pd.ear`. Then click **Finish**.

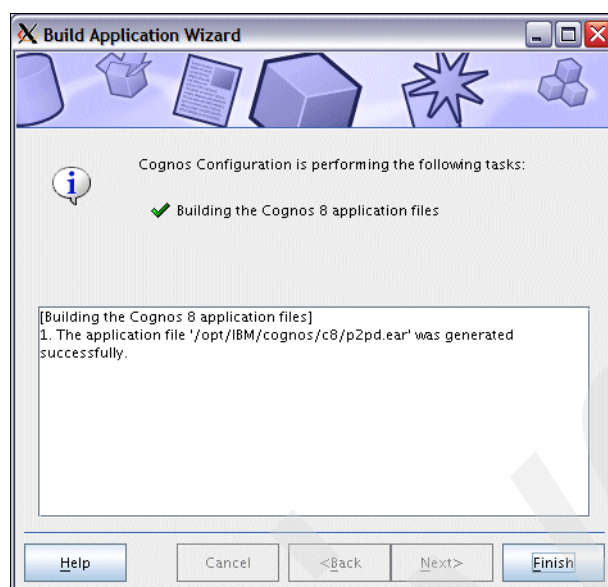


Figure 13-29 Report about successful creation of the p2pd EAR file

## Changing dispatcher URI settings

In the Cognos configuration tool, make sure that an existing service is stopped (if you ran with Tomcat before) and modify the dispatcher URIs to point to the WebSphere Application Server ports. In our example, we change the port number to 9080 (Figure 13-30), leave the rest unchanged, and save the changes.

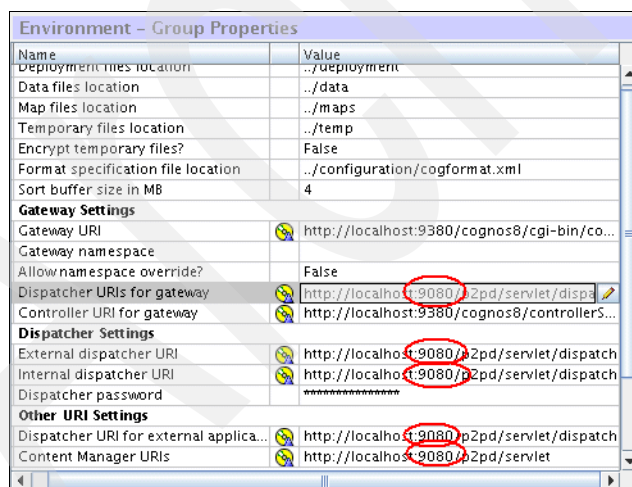


Figure 13-30 Changing the dispatcher port number

## Installing the application files

To install the application files:

1. Open the WebSphere Application Server administration console in a Web browser. We type the following URL in our example:  
`http://lncdwh1:9060/admin`
2. Navigate to **Applications** → **Install New Applications**.
3. In the Preparing for the application installation pane (Figure 13-31), select **Remote File system** and type `/opt/IBM/cognos/c8/p2pd.ear` for Full path. Click **Next**.

The screenshot shows the 'Integrated Solutions Console' with a 'Welcome' message and a 'Help | Logout' link. The left sidebar contains a tree view with categories like 'Guided Activities', 'Servers', 'Applications', 'Resources', 'Security', 'Environment', 'System administration', 'Users and Groups', 'Monitoring and Tuning', 'Troubleshooting', 'Service integration', and 'UDDI'. The 'Applications' category is expanded, showing 'Enterprise Applications' and 'Install New Application'. The main pane is titled 'Enterprise Applications' and 'Preparing for the application installation'. It contains the following fields and options:

- Specify the EAR, WAR, JAR, or SAR module to upload and install.**
- Path to the new application**
  - ☐ Local file system
    - Full path: [ ] Browse...
  - ☒ Remote file system
    - Full path: `/opt/IBM/cognos/c8/p2pd.ear` Browse...
- Context root** [ ] Used only for standalone Web modules (.war files) and SIP modules (.sar files)
- How do you want to install the application?**
  - ☒ Prompt me only when additional information is required.
  - ☐ Show me all installation options and parameters.
- Next** **Cancel**

Figure 13-31 Specifying the remote file system

- In the Select installation options pane (Figure 13-32), accept all default install options. Note that the application is named Cognos 8. Click **Next**.

**Install New Application**

Specify options for installing enterprise applications and modules.

**Step 1: Select installation options**

**Select installation options**

Specify the various options that are available to prepare and install your application.

☐ Precompile JavaServer Pages files

Directory to install application

☒ Distribute application

☐ Use Binary Configuration

☐ Deploy enterprise beans

Application name

Cognos 8

☒ Create MBeans for resources

☐ Enable class reloading

Reload interval in seconds

☐ Deploy Web services

Validate Input off/warn/fail

warn

☐ Process embedded configuration

**File Permission**

Allow all files to be read but not written to

Allow executables to execute

Allow HTML and image files to be read by everyone

Set file permissions

.\*\,dll=755#.\*\,so=755#.\*\,a=755#.\*\,sl=755

Application Build ID

Unknown

☐ Allow dispatching includes to remote resources

☐ Allow servicing includes from remote resources

Next Cancel

Figure 13-32 Installation options for the Cognos 8 application

- In the Map modules to servers pane, accept the default to deploy the application on the single server. Click **Next**.
- In the Map virtual hosts for Web modules pane, accept the defaults and click **Next**.
- Review the summary pane and click **Finish** to start installation of the EAR file. Example 13-18 shows the resulting output for our example.

*Example 13-18 Installation report of p2pd.ear*

ADMA5016I: Installation of Cognos 8 started.

ADMA5067I: Resource validation for application Cognos 8 completed successfully.

ADMA5058I: Application and module versions are validated with versions of deployment targets.

ADMA5005I: The application Cognos 8 is configured in the WebSphere Application Server repository.

ADMA5053I: The library references for the installed optional package are created.

ADMA5005I: The application Cognos 8 is configured in the WebSphere Application Server repository.

ADMA5001I: The application binaries are saved in  
/opt/IBM/WebSphere/AppServer/profiles/cognos8/wstemp/0/workspace/cells/lnxdwh1Node03Cell  
/applications/Cognos 8.ear/Cognos 8.ear

ADMA5005I: The application Cognos 8 is configured in the WebSphere Application Server repository.

SECJ0400I: Successfully updated the application Cognos 8 with the appContextIDForSecurity information.

ADMA5011I: The cleanup of the temp directory for application Cognos 8 is complete.

ADMA5013I: Application Cognos 8 installed successfully.  
Application Cognos 8 installed successfully.

---

## 8. Select **Save directly to the master configuration**.

This completes the installation of the Cognos 8 application in WebSphere Application Server. The application is still in the state of *stopped*, but to make the earlier JVM settings effective, we must restart the application server anyway. After the restart, the Cognos 8 application is started automatically.

Restart the application server with the following two commands. Again, use the scripts in the profile's directory if you do not use the default profile.

```
/opt/IBM/WebSphere32/AppServer/bin/stopServer.sh server1  
/opt/IBM/WebSphere32/AppServer/bin/startServer.sh server1
```

After the restart is completed, check the SystemOut.log file in the server's log directory and search for an entry like the one in Example 13-19. Note that it might take a few minutes after the server starts until the dispatcher reports that it is ready for requests.

*Example 13-19 Output in SystemOut.log indicating that dispatcher is ready to process requests*

---

```
[6/3/08 15:13:49:034 CEST] 0000000a WsServerImpl A WSVR0001I: Server server1 open for  
e-business  
^B[6/3/08 15:15:05:802 CEST] 00000034 SystemOut 0 JMXConnectorServer started at:  
service:jmx:hessian://lnxdwh1:9080/p2pd/hessian  
[6/3/08 15:15:06:150 CEST] 00000034 SystemOut 0 The dispatcher is ready to process  
requests.
```

---

If you encounter error messages such as the ones in Example 13-20, there is either a problem with the JDK (64-bit versus 31-bit) or the library path to the Cognos binaries is not properly set up.

*Example 13-20 SystemOut.log messages during servlet startup*

---

```
[6/2/08 15:13:28:274 CEST] 00000025 WebApp E [Servlet Error]-[dispatcher]:  
java.lang.NoClassDefFoundError: com.cognos.accman.jcam.crypto.jni.JNISystemProperties  
(initialization failure)  
    at java.lang.J9VMInternals.initialize(J9VMInternals.java:132)  
    at  
com.cognos.accman.jcam.crypto.SystemProtectionSession.<init>(SystemProtectionSession.java:3  
9)  
    at com.cognos.accman.jcam.crypto.misc.Configuration.<init>(Configuration.java:49)
```

```
at
com.cognos.accman.jcam.crypto.misc.Configuration.getInstanceWithDefaultConfig(Configuration
.java:96)
at com.cognos.accman.jcam.crypto.CAMFactory.initialize(CAMFactory.java:131)
at com.cognos.caf.CAFFactoryImpl.configure(CAFFactoryImpl.java:89)
...
```

---

### Testing the new setup

After directing your Web browser to the gateway address, which is the following address in our example, you can access the Cognos BI Server functions as usual:

<http://lnxdwh1.boeblingen.de.ibm.com:9380/cognos8>

You can also try to access the dispatcher directly by typing the following URL:

<http://lnxdwh1.boeblingen.de.ibm.com:9080/p2pd/servlet/dispatch/ext>

Or you can type the following URL to validate access to the content manager:

<http://lnxdwh1.boeblingen.de.ibm.com:9080/p2pd/servlet/>

## 13.3 Building Cognos data models and packages

In this section, we give an overview the Cognos 8 client modeling tools used to implement our scenario. We also discuss the development and deployment of the packages that we used to create queries and reports from the portal Cognos Connection.

The following client modeling tools are installed in our scenario:

- ▶ Cognos 8 Framework Manager for building packages that contain business reporting views of the operational data store (ODS) and dimensional data store (DDS)
- ▶ Cognos 8 BI Transformer for building the model that will be used to generate an OLAP PowerCube

This cube is used to demonstrate multidimensional online analytical processing (MOLAP) analysis

Transformer was installed both on a Windows client machine for developing the model and on the server for the processing the model and deployment of the cube. You can find complete installation instructions for these components in the *Cognos 8 BI Installation and Configuration Guide*.

For guidelines and best practices on modeling metadata in Framework Manager, refer to the *Cognos 8 BI Guidelines for Modeling Metadata* manual and the *Cognos 8 BI Framework Manager User Guide*. For guidelines about modeling metadata in Transformer, refer to the *Transformer User Guide*.

After you install the client modeling tools, you have to update the connection details of the Cognos configuration on the client machine, as shown in Figure 13-33, to ensure a successful connection with the Cognos server.

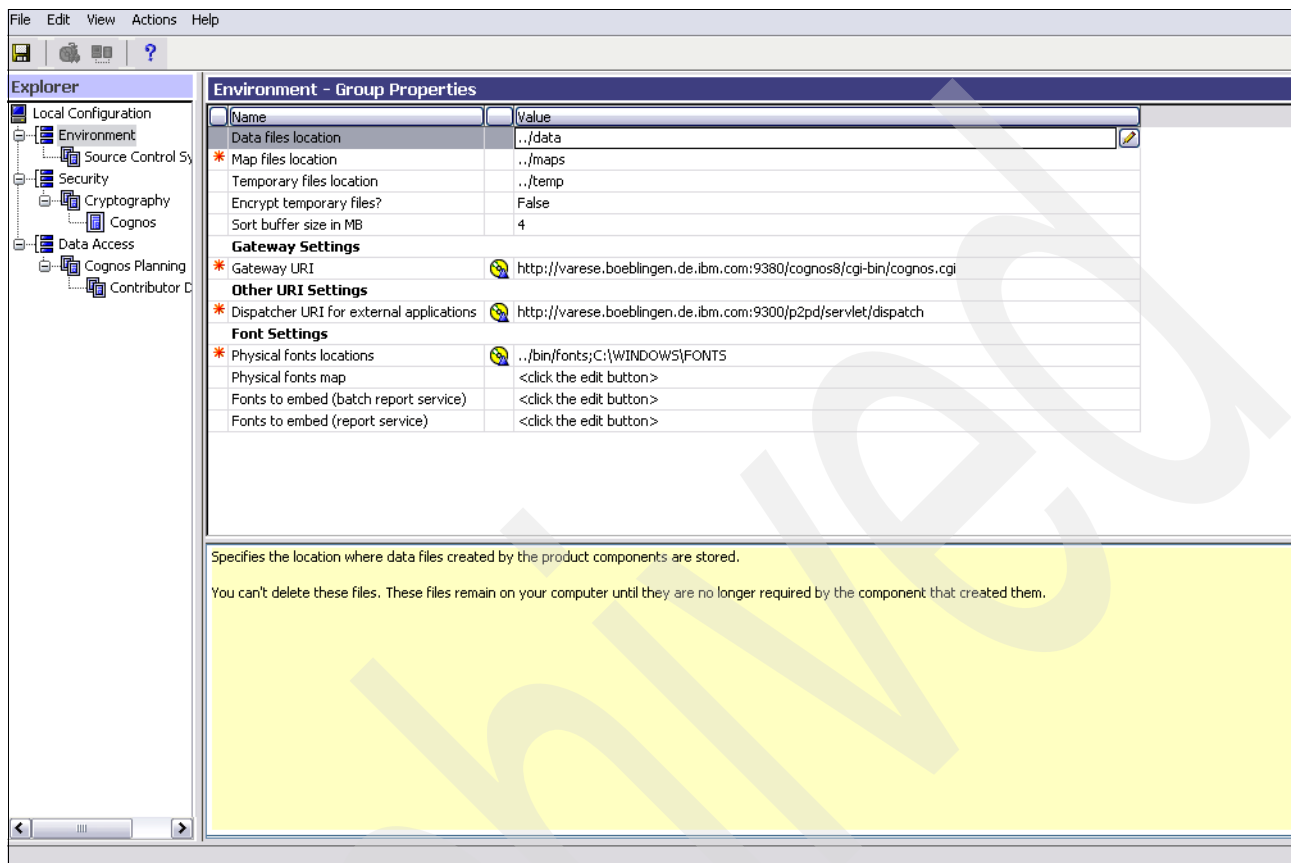


Figure 13-33 Cognos configuration



You might also have to check the cryptography security settings in the left pane (**Security** → **Cryptography**) and then click the **Test** button to generate cryptographic information (Figure 13-34). Otherwise you might see an error message that indicates that the credentials (user ID and password) cannot be decrypted when you later try to build a data source.

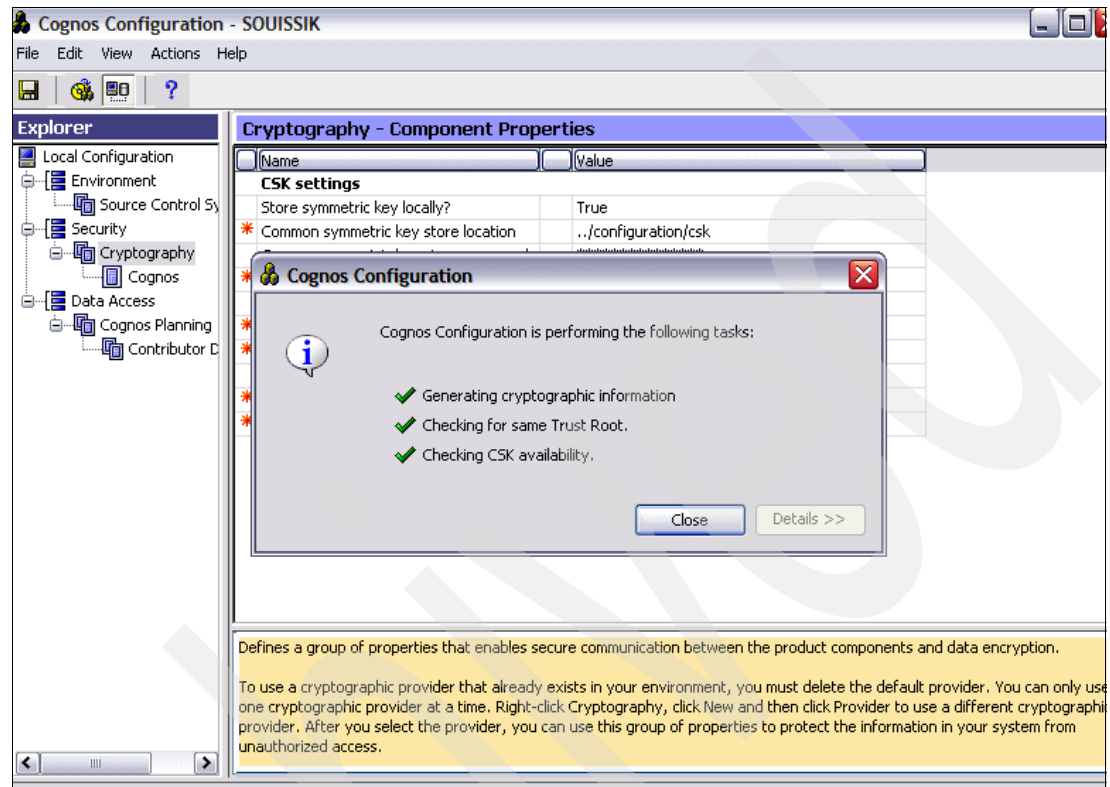


Figure 13-34 Cognos configuration test

### 13.3.1 Framework Manager packages

In this section, we provide an overview of the development and deployment of the Framework Manager packages that we used to implement our scenario in Cognos 8 Business Intelligence. For detailed information and guidance on modeling within Framework Manager, refer to the Cognos publication *Framework Manager Guidelines for Modeling Metadata*.

Framework Manager is a metadata modeling tool. A *model* is a business presentation of the information from one or more data sources that is published and made available for multiple groups of users to serve their query and reporting requirements.

Table 13-2 shows the models that we developed along with the resulting packages that are published to Cognos Connection. We include the Cognos 8 Transformer model that we build later in 13.3.3, “Creating and publishing OLAP PowerCubes” on page 363, in this table to show a complete list.

Table 13-2 Models and packages

Model	Description	Published package
dwhz_RedParts_ODS	Framework Manager model containing information from the RedParts organization ODS. The model includes relational business views, filters, and calculated columns for performing reporting and querying within Cognos 8 BI.	RedParts Distribution ODS (Query)
dwhz_RedParts_DDS	Framework Manager model containing information from the RedParts organization DDS. The model includes relational business views, filters, and calculated columns for performing reporting and querying within Cognos 8 BI.  <b>Note:</b> Typically this model might also include dimensional data views for performing relational aware analysis against the source data. These views can then be published as another package for analysis tasks.	RedParts Distribution DDS (Query)
RedParts Order Analysis.pyj	Transformer model showing the relationship between defined dimensions and measures for the analysis of order transaction information.	RedPart Order (Analysis)

**Note:** The model design strategy that we used was to create separate projects and packages for the ODS and the DDS environment. We did this because it typically aids in maintenance, providing flexibility, and there is no overlap in the metadata we wanted to make available to users in different reporting, querying, or analysis scenarios. If your reporting requirements show that you might want to combine your ODS and dimensional data, consider defining one Framework Manager project for both. You might also need to consider the size of the model and the number of query subjects in regard to the manageability of the project.

In the following sections, we explain the key steps and concepts that we followed in implementing the models and publishing the packages. We do not show the step-by-step process but only highlight key areas of the modeling that were undertaken.

## Creating the model project files and importing metadata

To create the model project files and import the metadata:

1. Start Cognos 8 Framework Manager and click **Create a new project** (Figure 13-35).

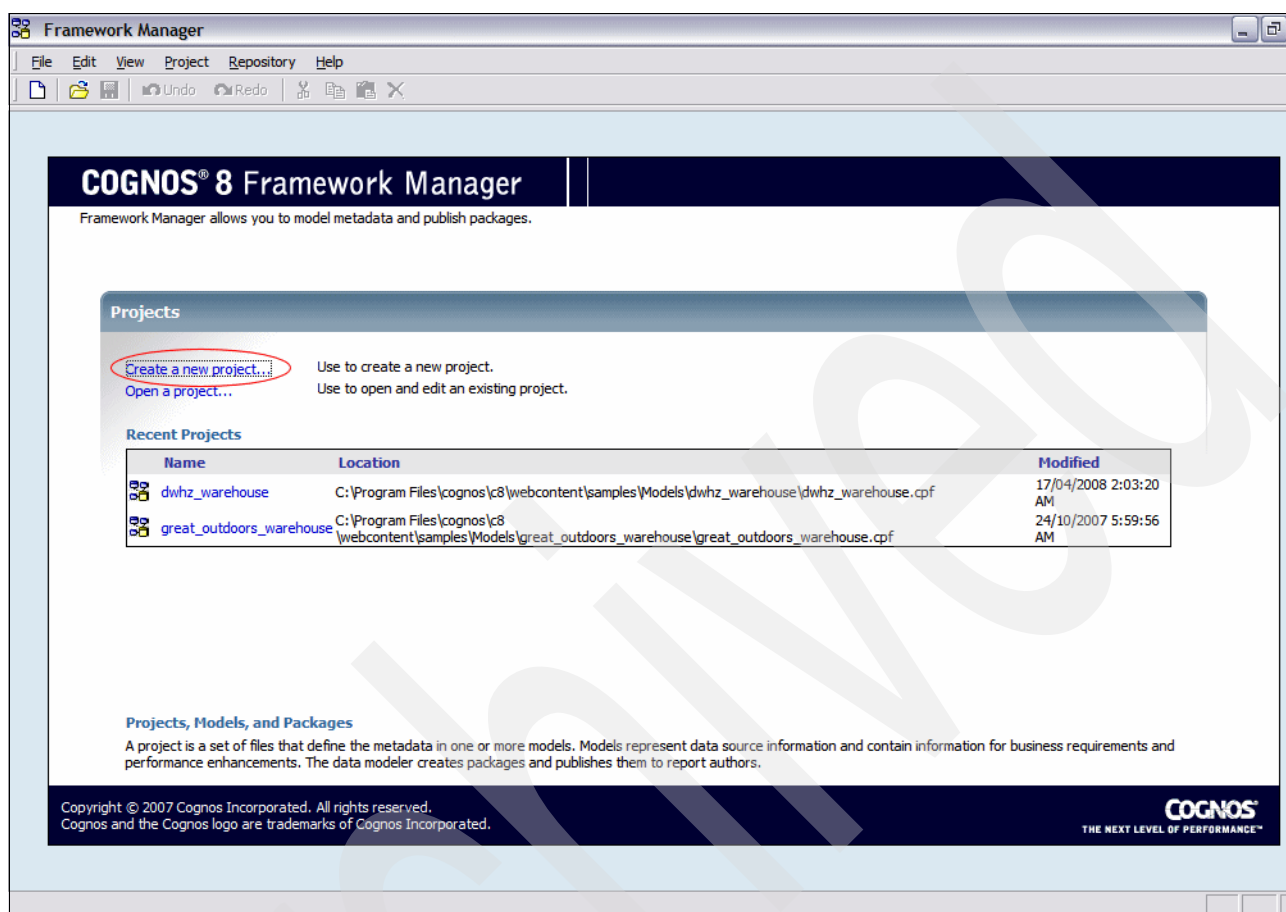


Figure 13-35 Creating a new Framework Manager project

2. Specify a name for the project and update the location to store the project files if required. A project contains a project file (.cpf) and other related files in the one location. Click the **OK** button. The wizard displays a message indicating that it is creating the project and initializing the components.
3. Choose the language for the project, which in our case is **English**. Click **OK**.

**Tip:** At this point, you can continue within the wizard to define the data source and import metadata for the project or you can click **Cancel** to create an empty project and manually perform the import. Figure 13-36 shows where you can right-click and select Run Metadata Wizard to run the metadata import manually.

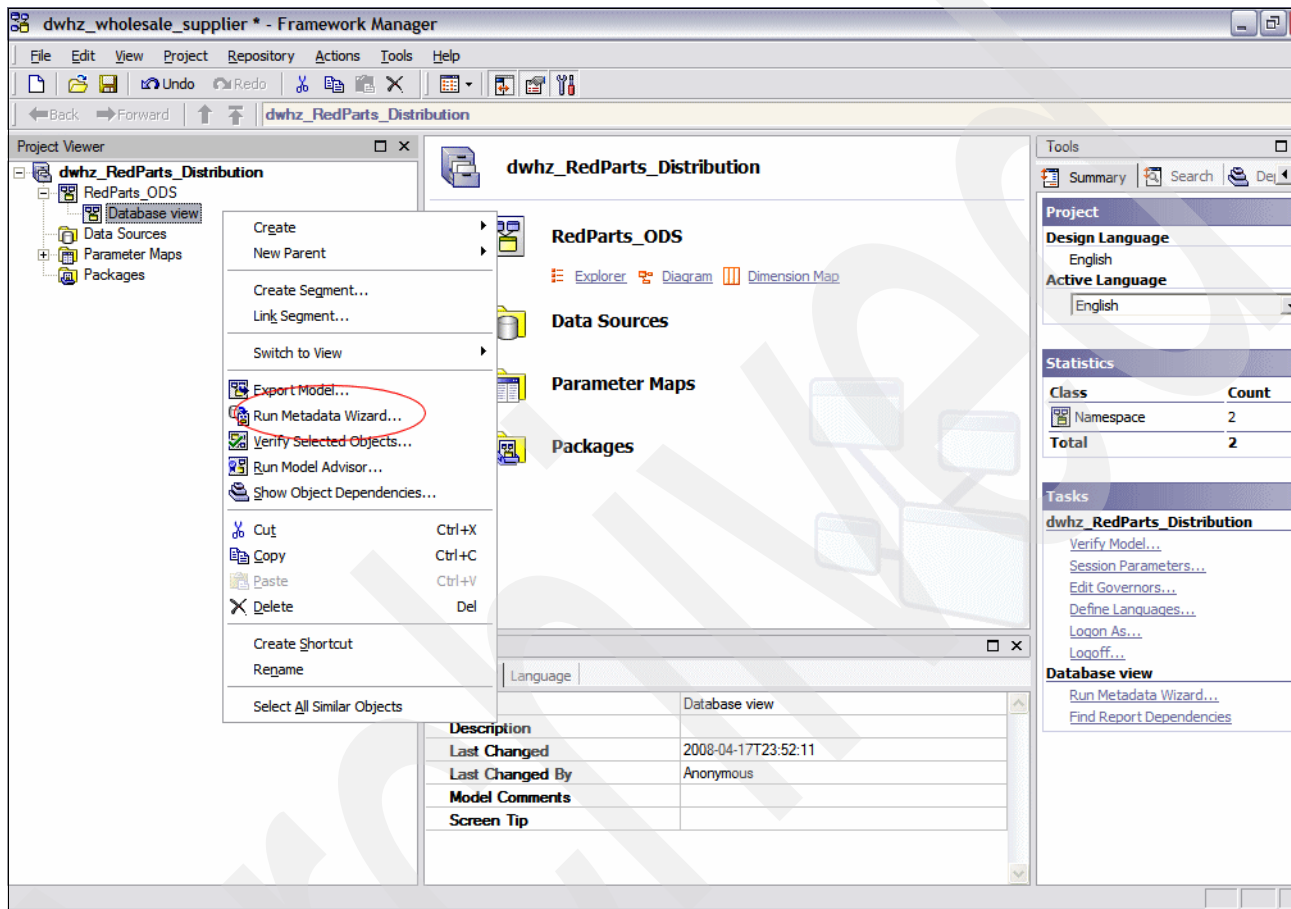


Figure 13-36 Manually running the Run Metadata Wizard

4. Select the metadata source as **Data Sources** and click **Next**.
5. In the Select Data Source window (Figure 13-37 on page 341), which shows the list of data sources that were previously defined in Cognos 8, click the scenario data source, which is **DWHD912** in our scenario. The process that we used to define our data source is explained in 13.2.7, “Defining DB2 for z/OS data sources” on page 323.

**Third-party metadata sources:** There are several alternatives to select from when choosing a metadata source. Particular selections can help in ensuring that you define metadata in a single environment and quickly bring additional metadata into your cognos environment.

The following examples are of metadata sources:

- ▶ IBM WebSphere DataStage, with which you can import the existing metadata that has been defined in XML or DSX files. XML files are recommended
- ▶ CA AllFusion Erwin for importing XML data model information such as physical and logical models and definition comments defined in Erwin

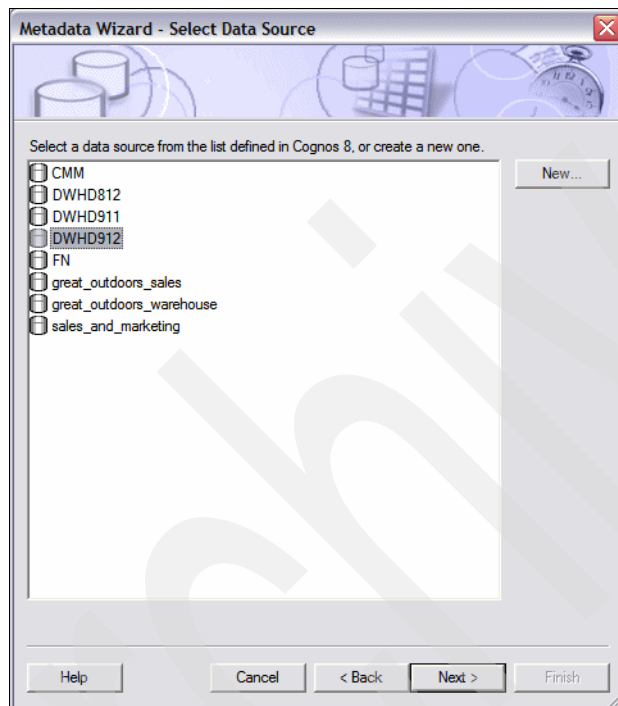


Figure 13-37 Cognos 8 defined data source list

**Note:** The list of data sources that is displayed in Figure 13-37 on page 341 is retrieved from the server definition where the Cognos BI Server component is installed and running. Framework Manager is a traditional client-server application. While it retrieves database connection information, such as authentication data, from the content manager, which might be on another machine, Framework Manager performs its own data access locally. Therefore, you must install the client and configuration tools for the data source on the machine where Framework Manager resides. For our scenario, you must have a suitable DB2 client installed and configured on your Windows machine, and you must catalog the DB2 for z/OS databases with the same name (DWHD911 and DWHD912 in our case). This ensures that you do not get an error message indicating that configured credentials (user ID and password) cannot be used to connect to the data source.

Click **Next**.

6. In the Select objects window (Figure 13-38), select the scenario database to import all metadata. We use the default option **Import and create a unique name** to create a unique name for objects when importing query subjects (tables) if the name is not already unique. Click **Next**.

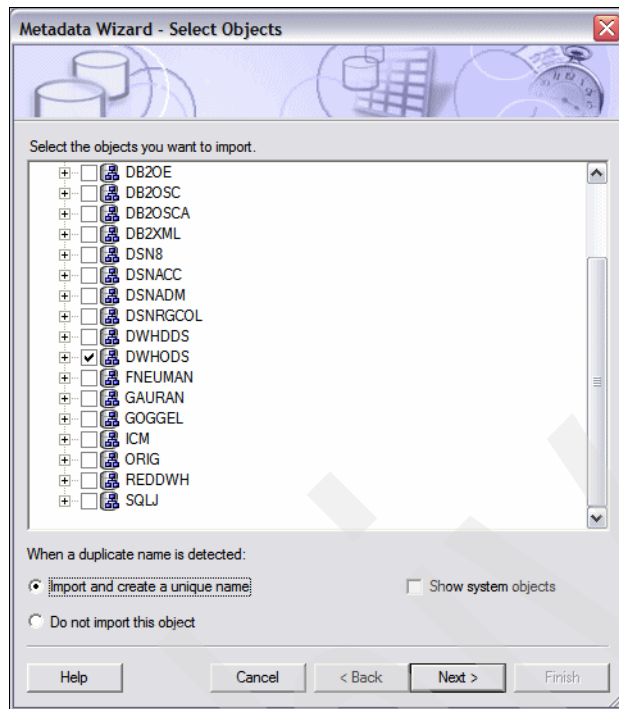


Figure 13-38 Metadata wizard - Selecting objects

7. In the Generate Relationships window, for our scenario, keep the default options and click **Import** for the following actions to occur:
  - a. Generate relationships and use primary and foreign keys to determine the relationship
  - b. Enable fact data detection for measures such as counts and currency
8. In the Metadata Wizard Finish window, click the **Finish** button to view the model.

The relational query subjects are now imported into the project under a namespace that has been given a default name. For our scenario, we renamed this namespace to reflect that it was based on the ODS or DDS physical models. By using namespaces, you can have more than one occurrence of an object with the same name to be in a model.

**Verifying metadata after import:** It is important to verify the import results from a requirements perspective after importing metadata. The items to verify include attribute usage properties, aggregation types (both used to determine aggregation rules), relationships, and determinants (similar to key fields).

In the Framework Manager project in Figure 13-39 on page 344, we highlight that the query item (field) L\_PARTKEY has been identified as usage type *fact*. This is noted by the icon in front of the attribute name and by viewing the Usage property in the properties pane. Framework Manager has determined this as a fact because the data type for the attribute is *integer*. In this case, we want this attribute to act as an identifier or key for records within the query subject. Figure 13-39 shows an example of changing the Usage property from *fact* to *identifier*.

## Data source query processing

Set the default data source query processing mode for the Framework Manager projects to *Limited Local*. Do this for both the ODS and DDS projects created in our scenario. With this setting, the database server can do as much processing as possible, while allowing local processing for some reports where required. If you do not set this mode, an exception error message is generated when an entire operation cannot be sent to the database engine. Follow these steps:

1. In Framework Manager, expand the **Data Sources** folder and click a data source from the list, for example, **DWH912**.
2. In the Properties window, locate the Query Processing property. Ensure the **Limited Local** is set as the property value by using the drop-down list.

## Project structure

Within our projects, we created further child namespaces to organize the project, provide context for the relationships between query subjects, and ensure that object names are unique. Our model structure is setup with namespaces to represent the physical *Database view* and the logical *Business view*. Depending upon requirements, in some cases, you can include a *Dimension view* to represent hierarchies and levels in your data and a *Presentation view*, which contains shortcuts to model objects, to provide a semantic layer to minimize the flow on effect of changes in your model. For more information about namespaces and project structure techniques, refer to the Cognos recommended practices in the *Cognos 8 Framework Manager - User Guide* and in the best practice guides.

Figure 13-39 shows the relational query subjects held within the Database view namespace after import. In the Project Viewer pane on the left side of the window, toward the end of the list, you also see the Business view namespace, which is where business views of the physical definitions are held.

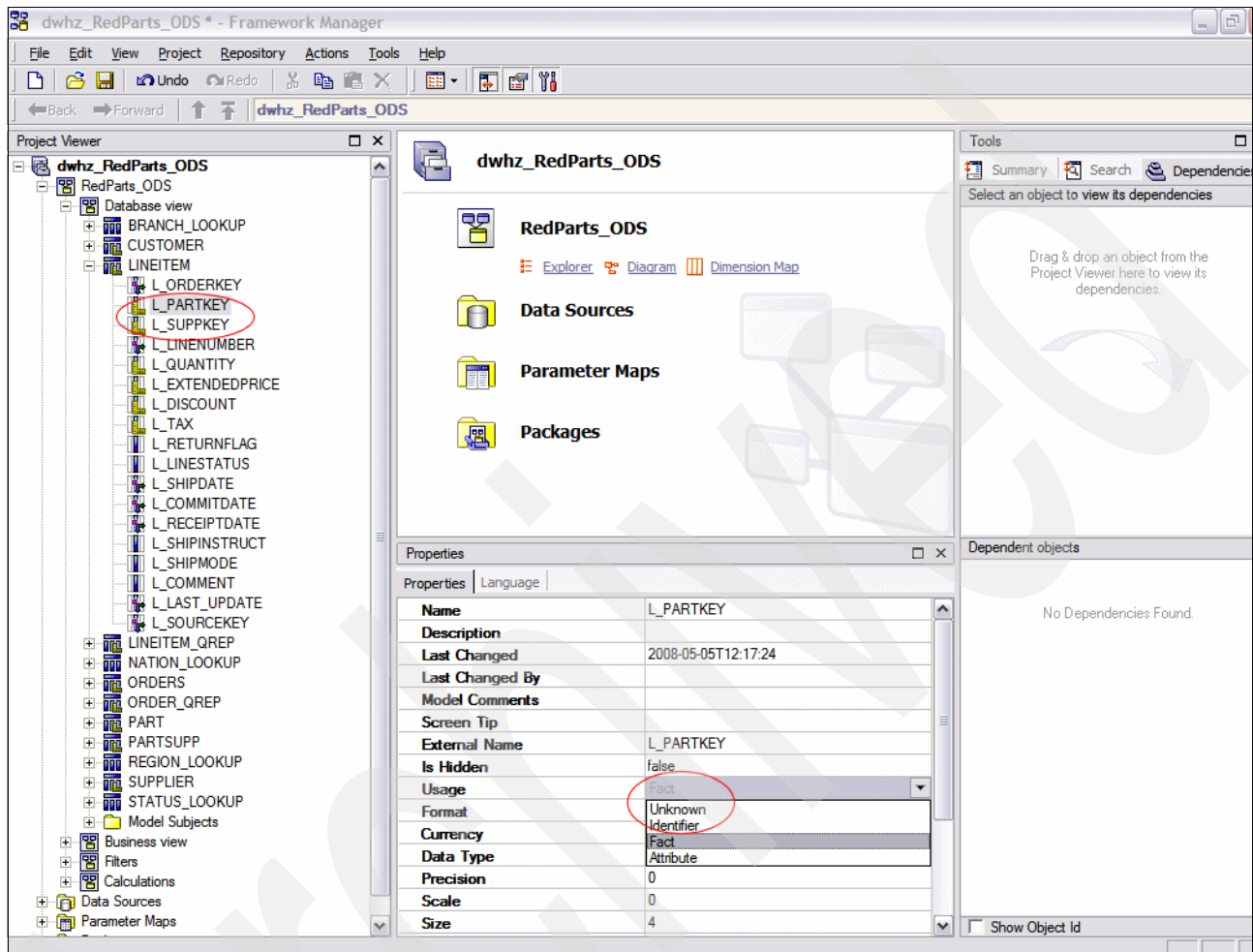


Figure 13-39 Database View - Imported metadata



Figure 13-40 demonstrates the definition of the business-oriented model query subject Order Details - Returned Products, stored within the Business view namespace. This model query subject is defined in order that, when selected by a user, the result set only contains the combined order and line-item information where the part has been returned. The original order and line-item query subjects are held in the Database view namespace.

As much as possible, it is best to leave these original objects untouched and have any extensions or modifications to them be made in the business view or logical namespace. This provides the opportunity to reduce or sometimes eliminate additional API calls to the source database for metadata at run time. By using this approach, model query subjects within the business view have their attributes renamed to be more meaningful to users when used in one of the Cognos studios. This model query subject also has a filter defined on the Filters page. This filter is used to restrict the data results to parts that have been returned by customers.

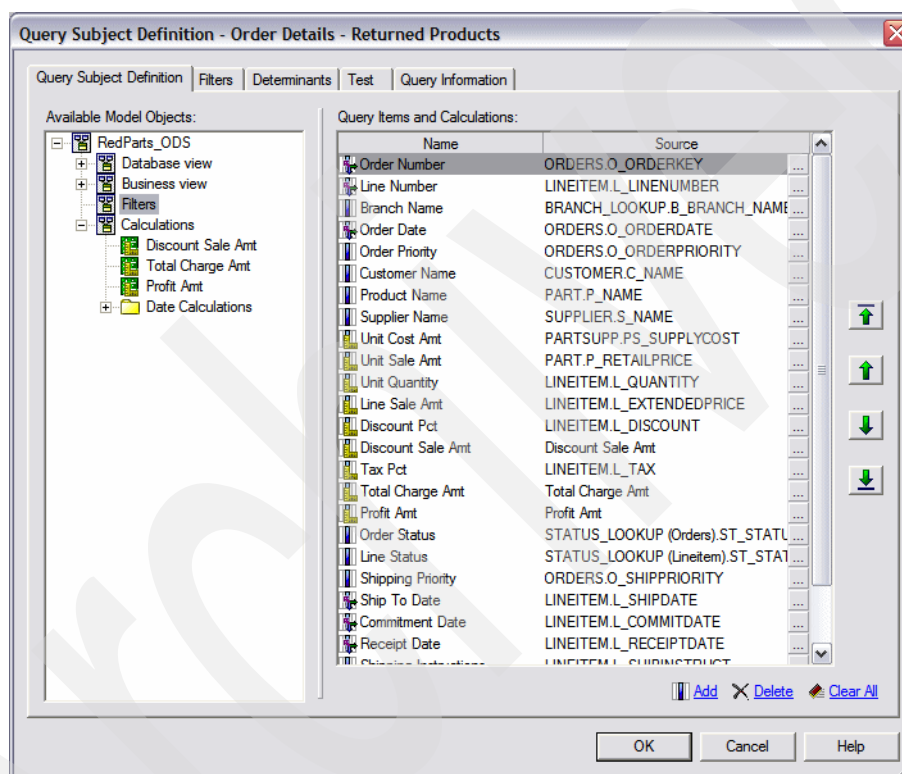


Figure 13-40 Example query object within the Business view namespace

**Note:** With Cognos 8, you can define model objects, such as the query subject, or use shortcuts. By using model objects, you can define the object as you choose to by including or excluding items. Shortcuts that point to a target object are less flexible, but are automatically updated when the target object changes. If there is no need to change an object and maintenance is a concern, shortcuts are useful. Shortcuts are also useful for designing a Presentation view in your model that points back to your Business view. This is helpful when changes occur in your model and you want to minimize the impact. Cognos 8 provides *regular* reference shortcuts and *alias* shortcuts.

Refer to the Cognos 8 BI product documentation for more information about shortcuts and their use.

Figure 13-41 demonstrates the ability to define dimensionally modeled relational (DMR) metadata into hierarchies and levels to represent dimensional data, by using the relational query subjects as a source. Dimensions can be published within a separate analysis package to perform relational aware analysis within Cognos 8 BI Analysis Studio and can also be used for drill-down functionality in reports. Such a package can also be used as a source to define an OLAP PowerCube within Cognos 8 Transformer.

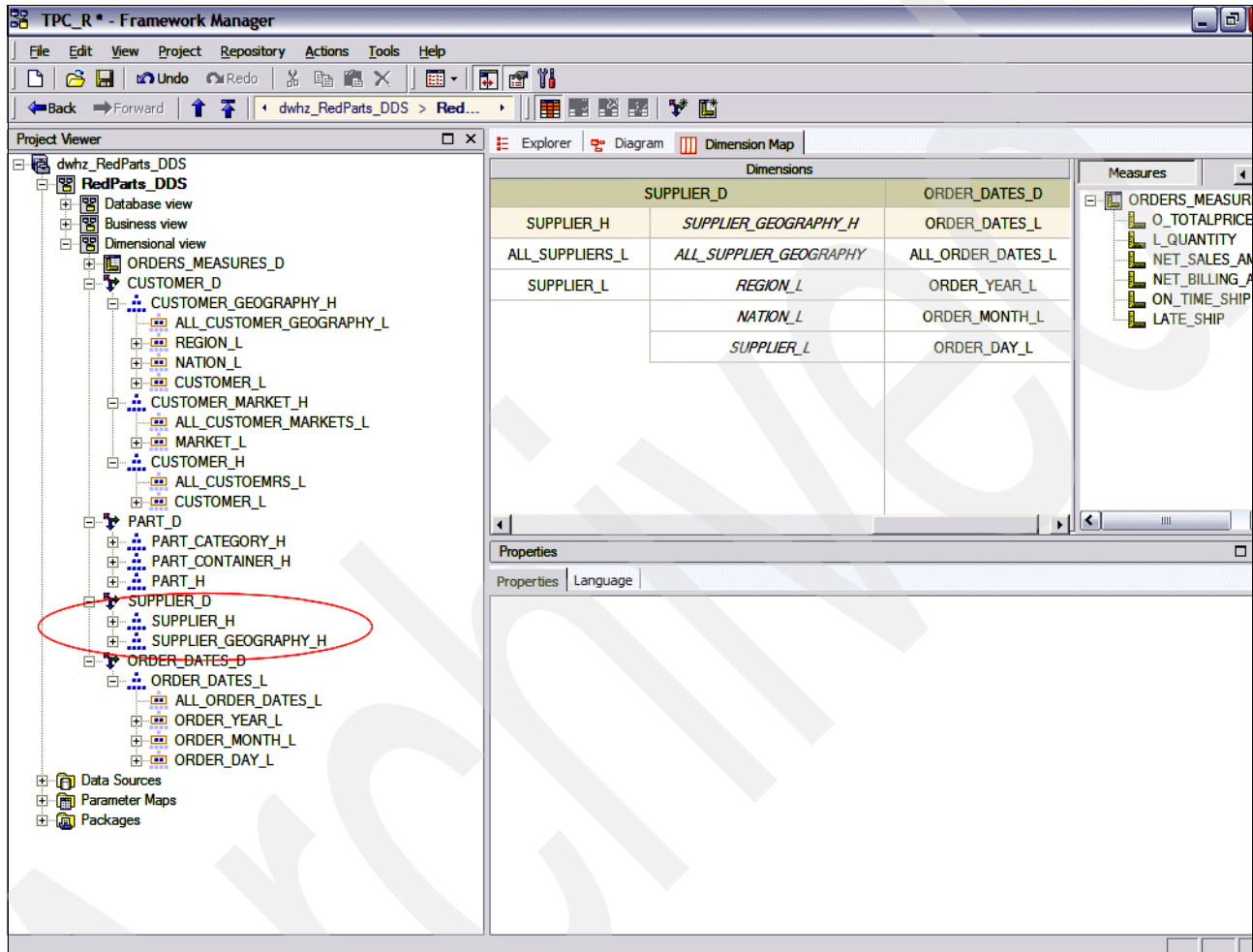


Figure 13-41 Example of a Dimensional view

**Data analysis using PowerCubes:** To conduct analysis of data, depending on your requirements, you can use either OLAP PowerCubes. If pre-cubing is not a requirement, you might model your relational data by using dimensional concepts. PowerCubes can often out perform relational equivalents and can be constructed from sources that might be required but that do not reside in your warehouse environment. PowerCubes can be combined with relational data to allow drill-through capability from analysis to detail. In addition, PowerCubes can be used for tabular and multidimensional queries.

## Creating or validating relationships

Within our scenario models, we validate existing relationships that were created during the metadata import wizard and create new ones. Figure 13-42 shows an example of a relationship being defined between the Customer and Order tables.

You can access the relationship definition window by using the following methods. Other shortcuts are not documented here.

- ▶ Click the query subject and select **Action** → **Create** → **Relationship**.
- ▶ Right-click a query subject and select **Create** → **Relationship**.
- ▶ In the Explorer tab, navigate to create or edit an existing relationship object. The Explorer is in the top right pane of Framework Manager. By using the Explorer, you can drill up and down within folders and namespaces. Relationships are identified by their icon showing a join between two query subjects.

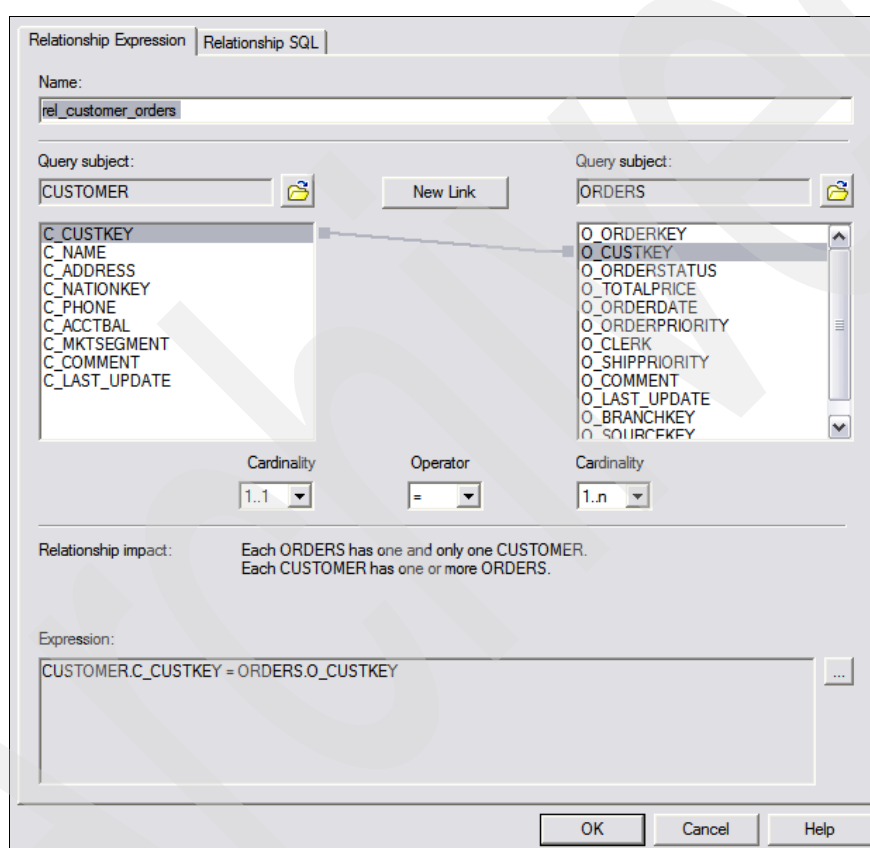


Figure 13-42 Defining relationships between query subjects

## Creating or validating determinants

Determinants describe functional dependencies between query items and reflect granularity by representing subsets or groups of data such as are required for a denormalized dimension table. For example, country code and region code can be used to identify the attribute region name in a dimension. Identifying this dependency by using determinants can be useful in some cases to ensure accurate results when querying data. Determinants are also used to ensure correct aggregation of repeated data within these subsets.

Determinants are most closely related to the notion of a key in a third normal form (3NF) database design. The collection of columns uniquely identifies one or more functionally

dependent columns. A best guess of required determinants can be auto populated by using RDBMS keys, constraints, and indexes that are read during the metadata import process.

You might want to create or modify determinants based on the business reporting requirements that you are implementing in your model. By doing this, you will override, within your model, the index and key information in your data source. The usage of index metadata from the source is only an indication to probable dependencies in your BI layer. When imported, modelers should review whether they need determinants, and if so, determine if they have to be modified accordingly.

Within our scenario models, we validate and modify determinants to ensure the correct aggregation within repeated data groups. Figure 13-43 shows the Branch key of the Branch lookup table that is being uniquely defined as a determinant along with its remaining attributes.

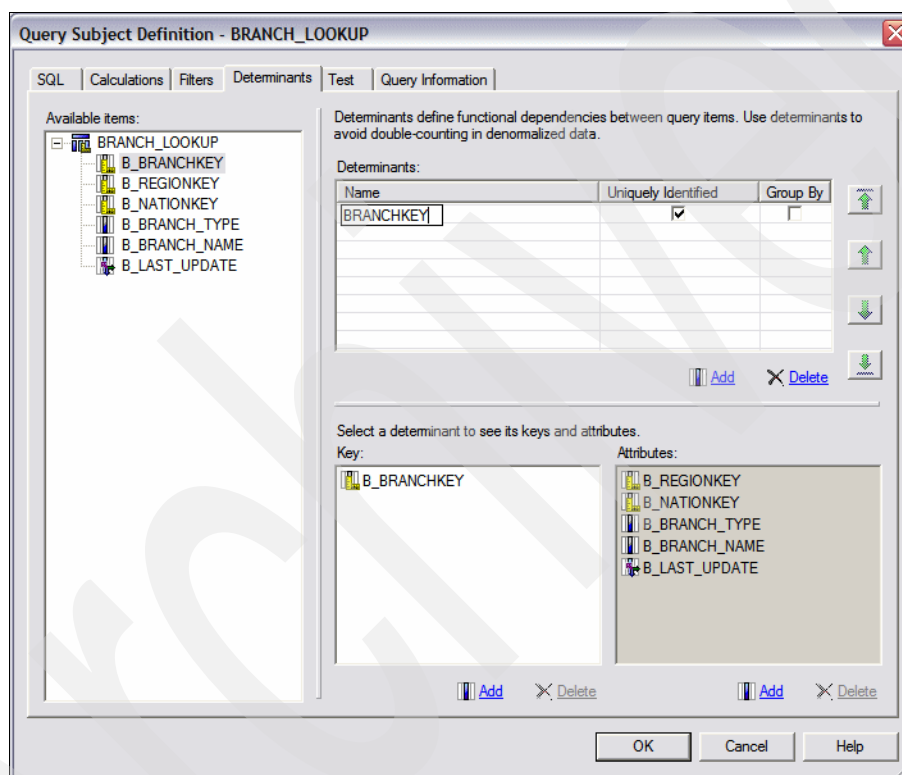


Figure 13-43 Single determinant definition

Figure 13-44 shows multiple determinants that are being created on the Branch dimension query subject within the scenario DDS database. Here we highlight that the extra non-unique determinants are created by selection of the Group By check box. This option is used to group and aggregate the data correctly. It is also used to apply aggregate functions and grouping to avoid double counting when a user reports at the Nation, Region, and Type level.

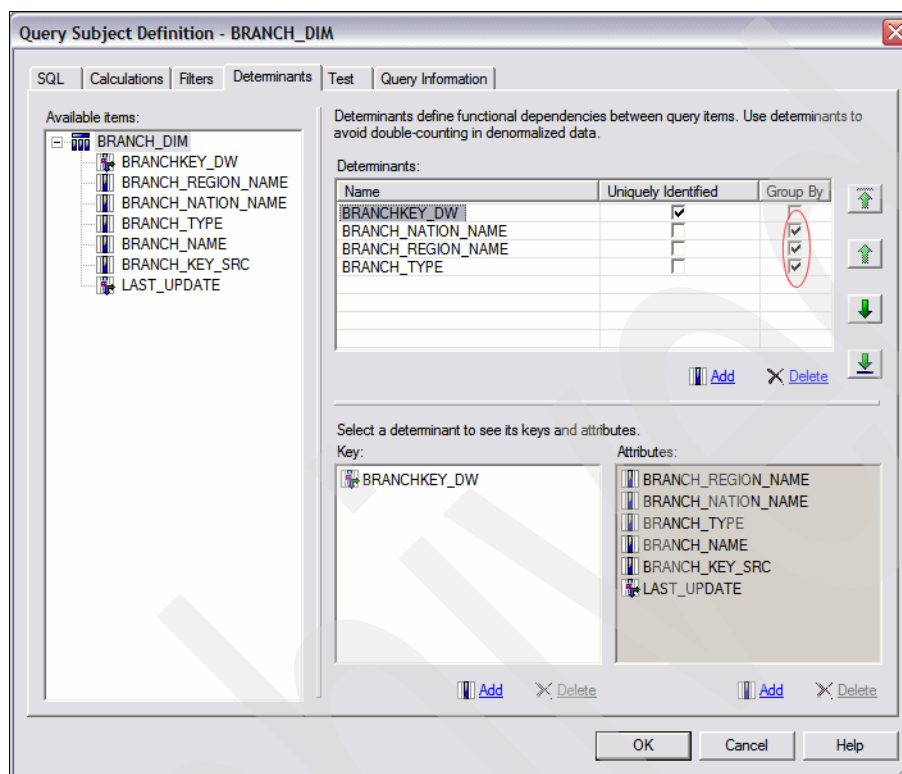


Figure 13-44 Multiple determinant definitions

## Creating filters and calculated columns

With Framework Manager, you can create filters and calculated columns within your models to support business rule implementation at the BI semantic layer. Investing in these objects has the advantage of providing consistency by defining the same definitions when used by multiple end-user query and report objects. Filters and calculated columns can also aid users in Web studios such as query studio and report studio. Users can drag several objects from the insertable objects pane and reduce the number of executions of requests on the database. Additionally, the filter and calculated column objects can contain logic that might be unable to be expressed in the given Web studio.

Filters and calculated columns can be defined as either stand-alone or embedded. Stand-alone has the benefit of being defined once and used across multiple model objects or published to be selected for use by users in a package. Embedded filters or calculated columns can be defined when the filter or calculation is only used by one model object, such as being included in the definition of a business view query item.

**Stand-alone calculations in Analysis Studio:** You cannot reference stand-alone calculations in Analysis Studio. You must use an embedded calculation or create the calculation within your analysis.

Depending upon whether you use stand-alone or embedded filters and calculations in your model, you may want to consider the timing of when the filter or calculation may be applied and the difference in possible generated SQL. For example, consider the situation where a user is querying a customer query subject that has no embedded filters and then drags a stand-alone filter on to their report. In this case, the generated SQL applies the filter differently than if the query subject had the filter embedded. You can find more discussion about this topic in *Cognos 8 Framework Manager - Guidelines for Modeling Metadata*.

A number of filters are created in our models that can be used later in implementing the scenario business requirements. Figure 13-45 shows the creation of a filter that, when selected within Cognos 8, restricts the result set to those records where the return flag is set to a value of 1. In our scenario, a value of 1 signifies that the parts were returned with a return reason of *Unsatisfactory*.

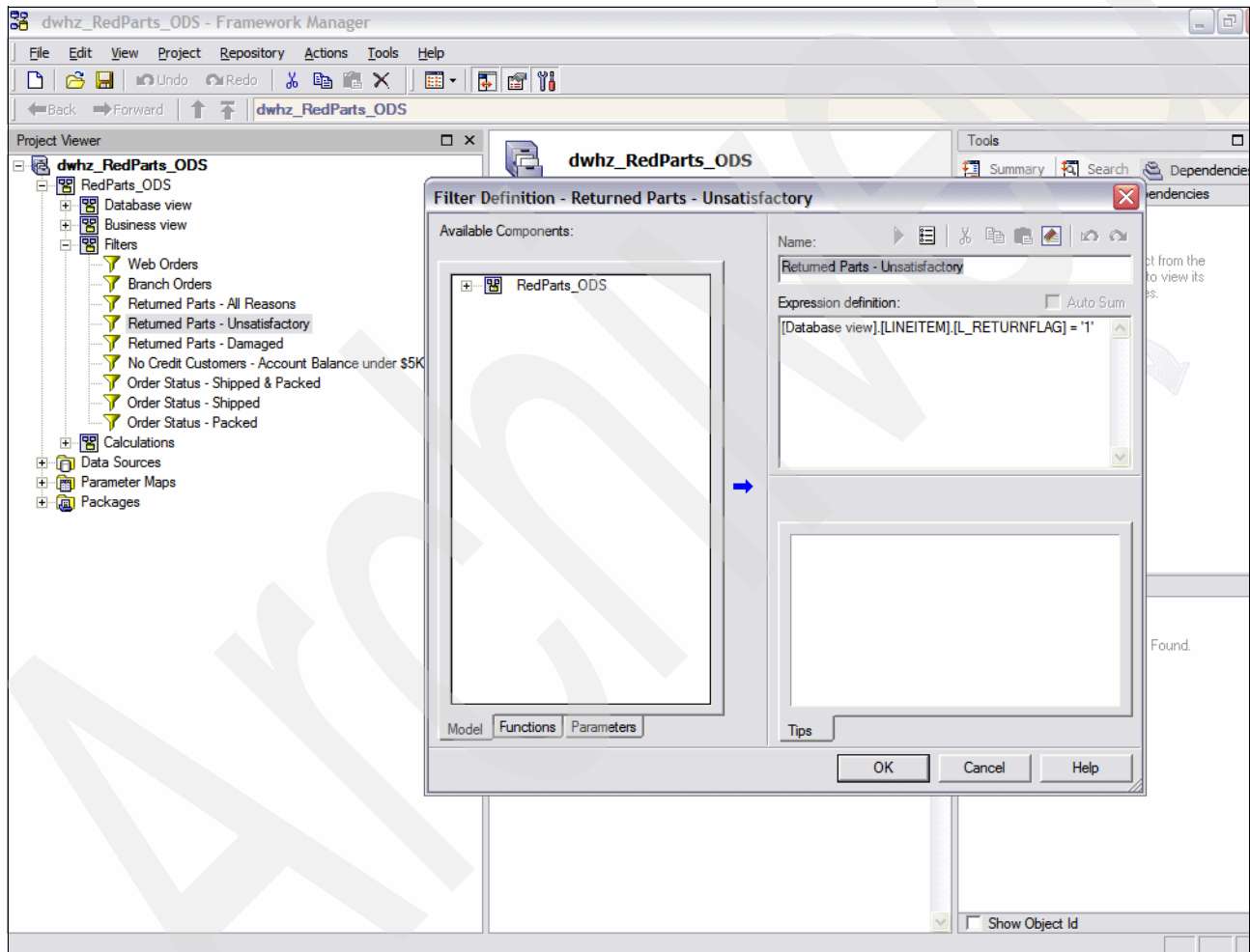


Figure 13-45 Defining a business filter

Calculated columns are also created to fulfill scenario requirements and to demonstrate this functionality. Figure 13-46 shows the calculation for Discount Sale Amount being created.

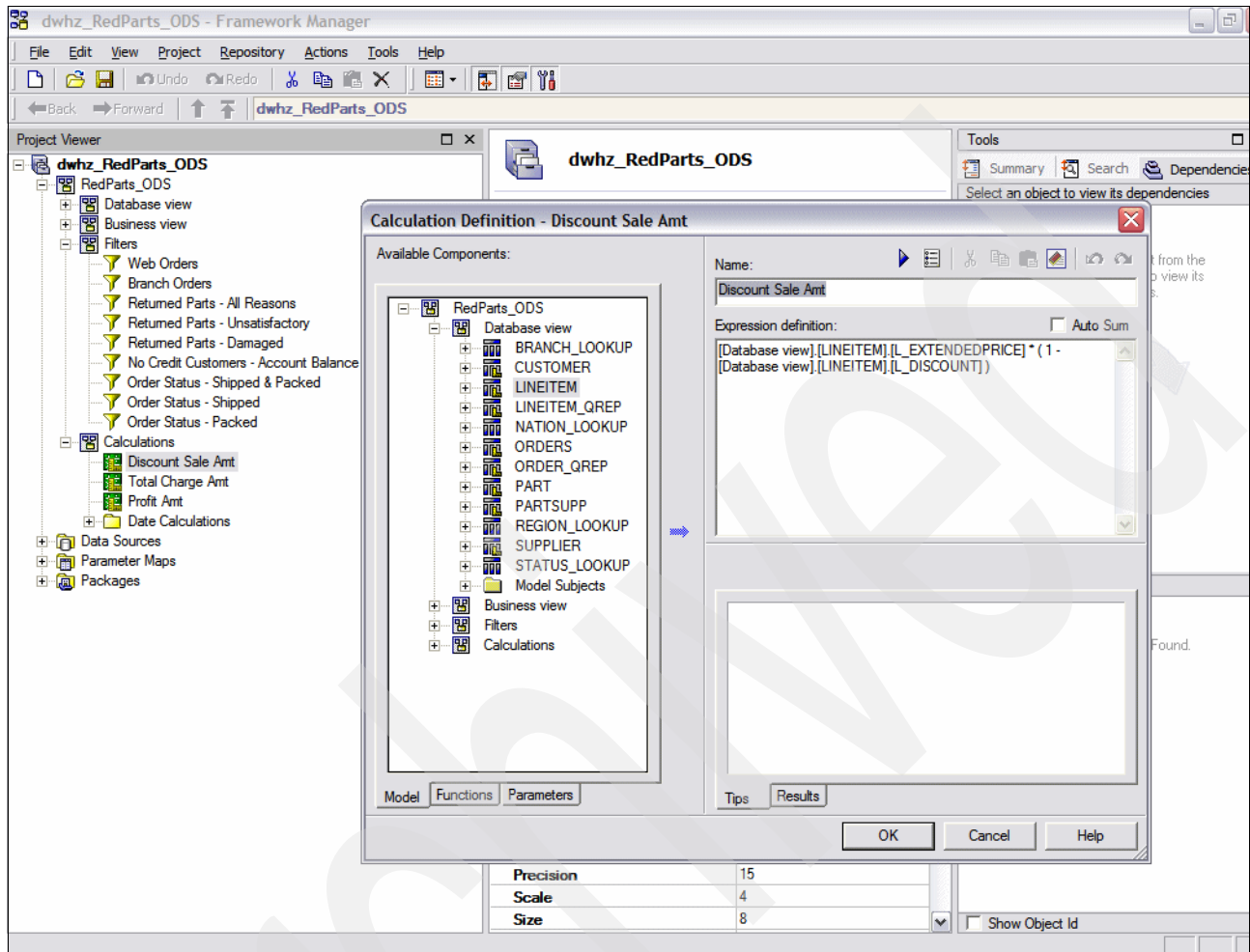


Figure 13-46 Defining a calculated column

## Publishing packages to the Cognos content store

Packages are a subset of your Framework Manager project and contain the metadata that you make available to users. The best practice is to define multiple packages for different usage types such as queries and analysis. Table 13-2 on page 338 documents the packages that we created as part of the RedParts scenario.



Figure 13-47 highlights the definition of the query-oriented package called *RedParts Distribution ODS (Query)*. This figure shows that the Database view namespace is not published but rather referenced as part of the final package definition. This is shown by the smaller green checkmark icon.

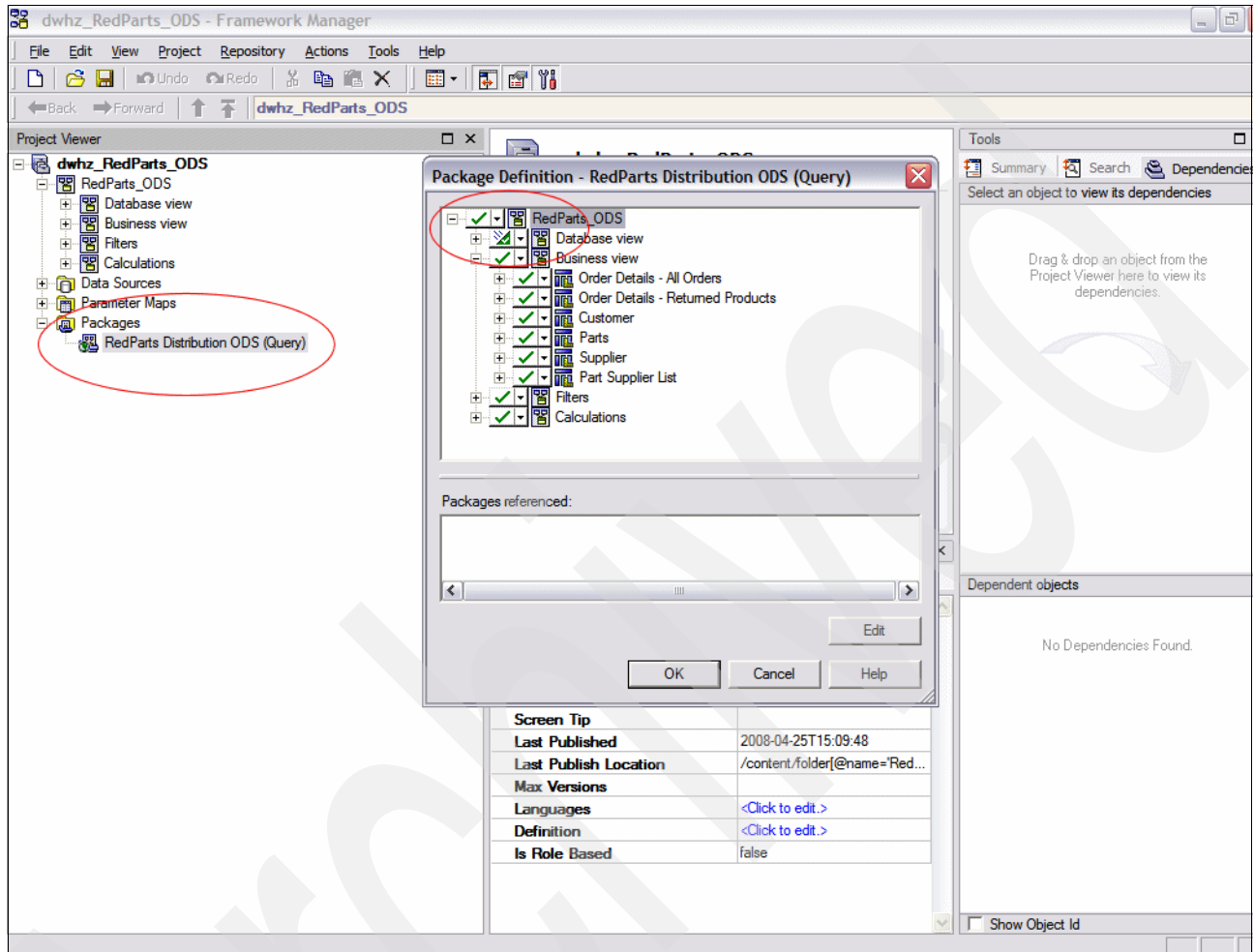


Figure 13-47 Defining packages to publish

### 13.3.2 Defining Transformer models for PowerCubes

For our scenario, we require an OLAP cube to perform product return rate analysis. The return rate is a calculated percentage that does not exist in the source data. The analysis shows, for a given financial quarter, the highest return rates by product, manufacturer, and customer nation.

In this section, we demonstrate the development of the Transformer model for building the OLAP cube. The information shows the steps to create the model data source and the selected techniques that we used to design and implement the model. Step-by-step instructions for building the model are not provided.

In 13.3.3, “Creating and publishing OLAP PowerCubes” on page 363, we explain the process that is used to deploy the OLAP PowerCube to Cognos 8. For our scenario, the model is designed in Transformer on the Windows platform. The model file is then transferred to the server where a script is used to direct the server Transformer installation to build the cube.



After the cube is built, we create the package for the PowerCube from Cognos Connection. See “Creating a data source for the PowerCube” on page 365. Alternatively we might have used Transformer’s new functionality to publish the cube directly to Cognos Connection.

**Combining PowerCube and relational data as a single published source:** If you want to combine a PowerCube and relational data together as a single published source for users, then you must use Framework Manager to publish the cube. Otherwise you can create the package for the PowerCube from Cognos Connection instead of using Framework Manager or publish the cube directly from Transformer.

## Defining the model data source

The data source for our model is the RedParts Distribution DDS (Query) package, which was published from Framework Manager. This package contains relational query subjects that we use to build a dimensional model.

**Building dimensions and measures in Transformer:** An alternative to building the dimensions and measures in Transformer is for us to define dimensions by using relational query subjects within the Framework Manager model and publish another Analysis package that includes these dimensions. Figure 13-41 on page 346 shows an example of this. Transformer can then use these dimensions to define the cube model file.

To set up the name of our model and set the published source package as the data source, we used the New Model wizard in Transformer:

1. Launch the New Model wizard. Either select **File** → **New** from the menu bar or right-click in the Data Source pane and select **Insert Data Source**.
2. In the initial New Model wizard panel, click **Next**.
3. Enter the model name as RedParts Order Analysis.pyj and click **Next**.
4. In the New Model panel (Figure 13-48), for Data source name, type RedParts Distribution DDS (Query) or leave the default variable. For Data source type, select **Cognos Package**. Click **Next**.

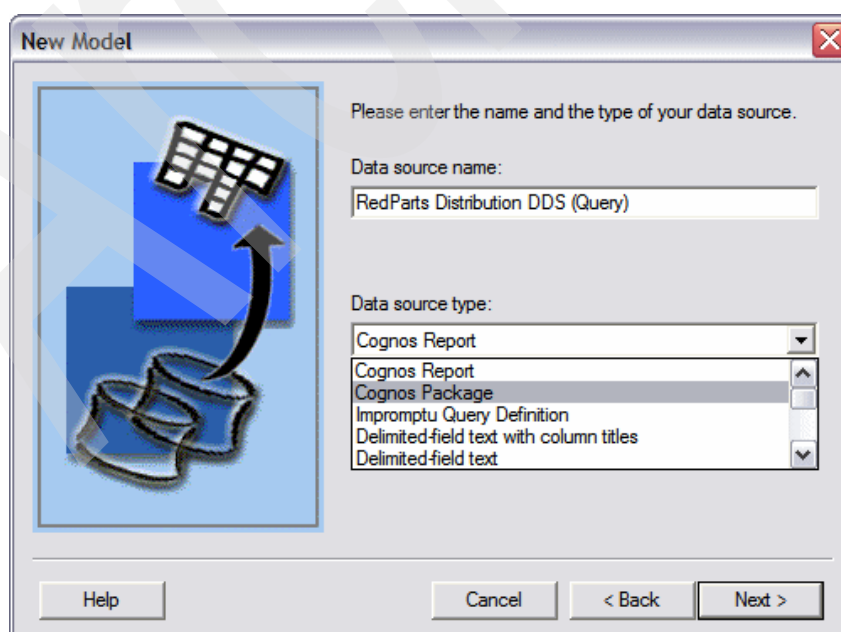


Figure 13-48 Defining Transformer model data source

5. In the Package Data Source panel, click **Browse**. Navigate to and select the **RedParts Distribution DDS (Query)** package. Click **OK** to return to the data source panel and click **Next**.
6. Define the attributes that you want to include as shown in Figure 13-49. To build the time dimension in Transformer, we use Transformer's relative time functionality with the Order Date field. We do this instead of using the rest of the date part fields that are defined in the source Order Date dimension table. Therefore, in Figure 13-49, we highlight that we do not include all fields from the Time Dimension that were defined in our source Framework Manager package. Click **OK** to continue.

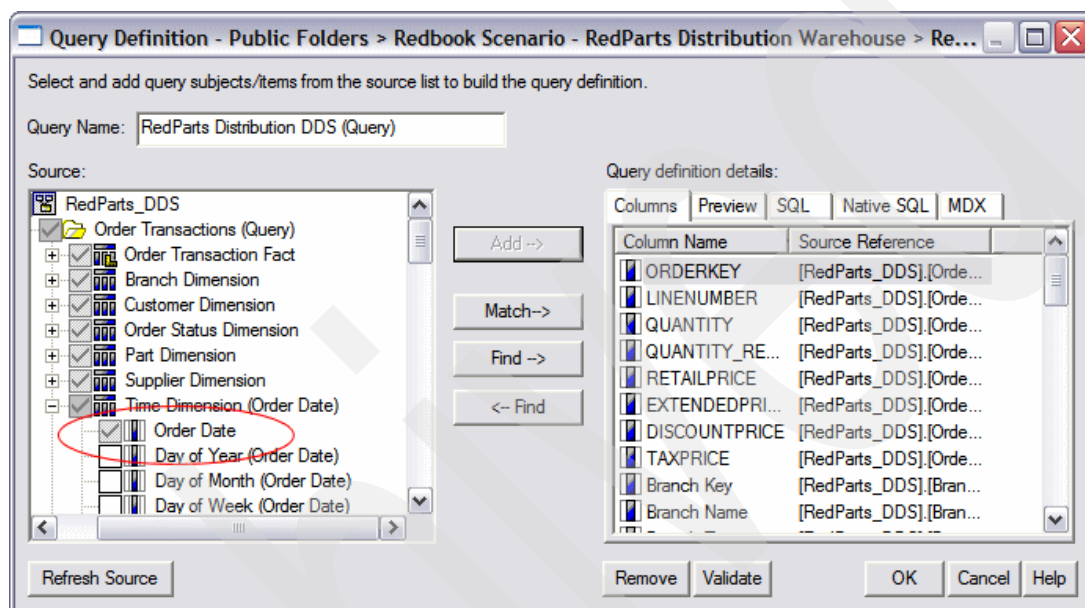


Figure 13-49 Query definition in Transformer

7. In the Finish panel, for our scenario, deselect the **Run AutoDesign** feature to build the model from scratch. Click **Finish**. A blank model is displayed with the query items that you selected from the source package shown in the Data Sources pane.

**Transformer warning message:** You receive the following Transformer warning message:

“The rollup type ‘automatic’ for ‘Total Charge Amt’ is not supported. The default will be used.”

The query item *Total Charge Amt* is defined in Framework Manager as a calculation with the Regular Aggregate property set to Automatic. Transformer does not support a regular rollup of Automatic. Therefore, a warning message is displayed to indicate that the default regular rollup will be used. In our case, the default Transformer regular rollup is set to Sum, which is correct for this field. That is, the Total Charge Amt for a given member in a level is the sum of the underlying amounts. Click **OK** in the warning message window to continue.

8. Double-click each source attribute to see the Data class definition property shown in Figure 13-50. If you are building the model without the AutoDesign feature, the data class can remain as *unspecified* because the source package contains the data type for each source field. If you are using the AutoDesign feature, you might want to verify that the data class is appropriate for how you want the design feature to use the field. Transformer attempts to set the correct data class by using information that is held in query item properties within the Framework Manager package.

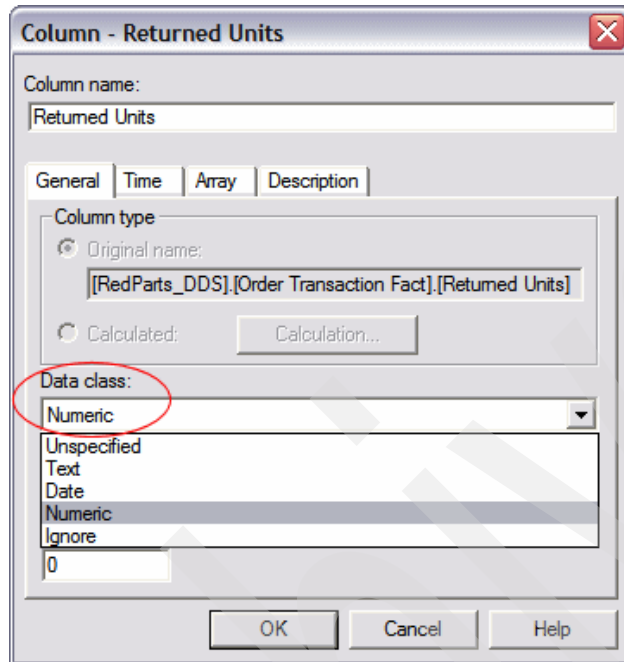


Figure 13-50 Data class definition for source fields

For our scenario, we build a Cognos relative time dimension by using the Order Date attribute. We confirm that the data class for the order date is set to *Date*. We also click the **Time** tab in the Column - Returned Units window (Figure 13-50), and for the degree of detail for the time field, we select **Day**. This ensures that Transformer automatically builds levels within the dimension to the day level. The date input format can be left as unspecified or set to match the input of YMD.

In the following sections, we show specific areas of interest that are required to build and complete our Transformer model.

## Insert dimensions

All dimensions in our model are created by right-clicking in the Dimension map and selecting Insert Dimension. Our time dimension is also created manually by using the Insert Dimension option and the Transformer's relative time functionality. An alternative is to use the Date Wizard or to drag the Order Date field to the top of the Dimension map to have Cognos create it automatically. Figure 13-51 highlights the Date Wizard and Insert Dimension options. Figure 13-58 on page 363 shows the Transformer model with all dimensions completed.

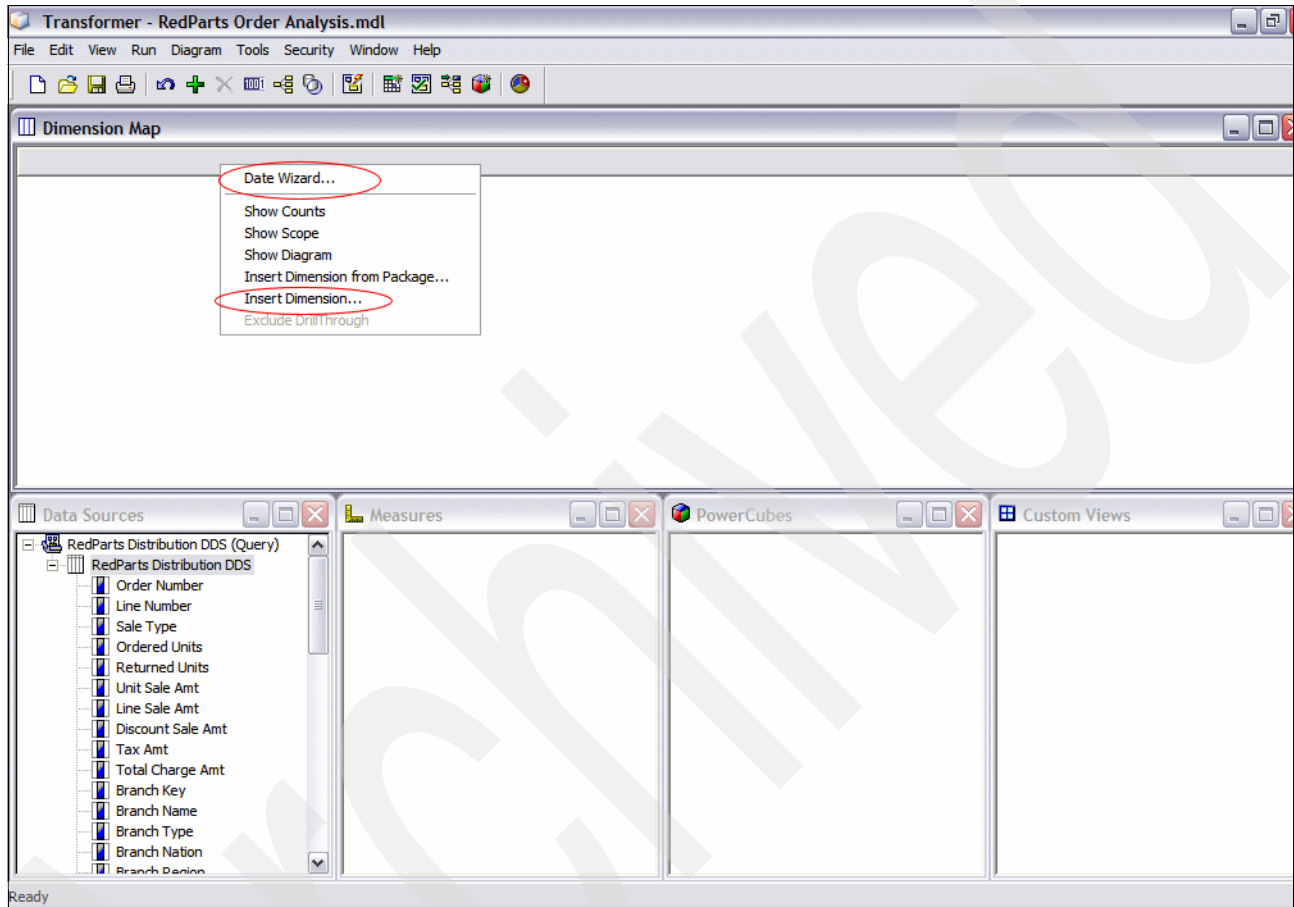


Figure 13-51 Insert dimensions

### Setting the dimension types

All non-time dimensions are set as Regular type dimensions. Figure 13-52 shows the dimension type for the time dimension All Order Dates that is set to a dimension type of Time. The time dimension also requires additional information to be set that is available under the Calculation and Time tabs. If you are creating this dimension manually and not dropping the Order Date field into the Dimension map, you must use the Calculation tab to set the source field to Order Date.

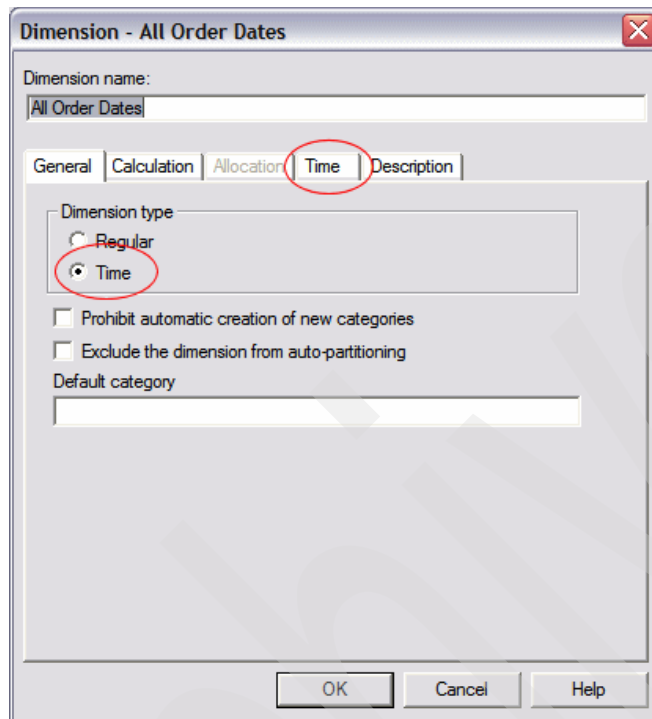


Figure 13-52 Defining the general properties for the time dimension

### Time dimension settings

Figure 13-53 shows the additional information that is set on the Time page. In our scenario, we want Transformer to set the current time period automatically. This will be used for current time and YTD related members in the cube and will be available as a selection in the time dimension by using Analysis Studio. The current time is set to the latest order date read by the data source definition at the time the cube is processed.

The screenshot shows a dialog box titled "Dimension - All Order Dates" with a close button (X) in the top right corner. The dialog has five tabs: "General", "Calculation", "Allocation", "Time", and "Description". The "Time" tab is selected. Inside the "Time" tab, there are two main sections. The left section is labeled "Regular time" and contains two date pickers: "Earliest date:" with a dropdown showing "1/01/1901" and "Latest date:" with a dropdown showing "31/12/2100". The right section is labeled "Days in week" and contains a list of days from Sunday to Saturday, each with a checked checkbox. Below these sections is a section labeled "Current period" which contains a checked checkbox labeled "Automatically set the current time period" and an empty text field labeled "Current period:". The "Current period" section is circled in red. At the bottom of the dialog are three buttons: "OK", "Cancel", and "Help".

Figure 13-53 Defining the time properties for the time dimension

### Dimension levels and role associations

Levels within dimensions are created by dragging the source field under the relevant dimension headers that were previously created. Figure 13-56 on page 361 shows an example of two dimensions with their levels created.

Double-click a level within the Dimension map to view the source associations for that level, as shown in Figure 13-54. In the following example, the Branch level is created by using the Branch Key field. We also want to add a label for the branch by using the Branch Name field. Click the **Order By** tab and click the **Add** button to add the branch name as the description or label.

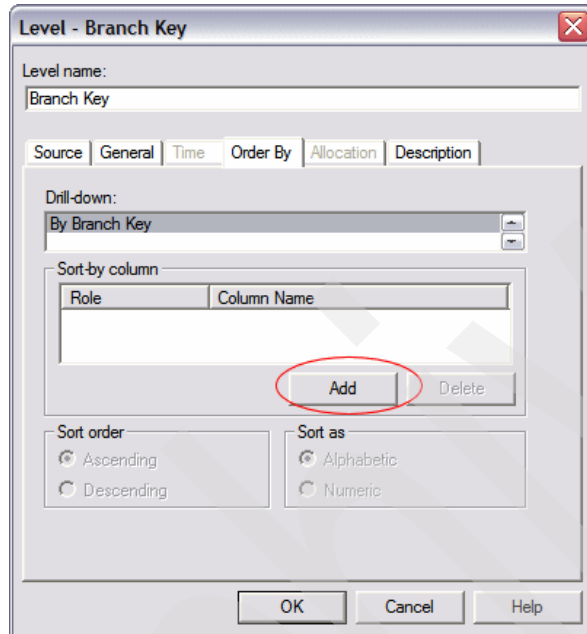


Figure 13-54 Adding the Branch Name for further role associations

Figure 13-55 shows the result of adding the field Branch Name as the label for this level. Highlighted also is that the refresh Label option is selected. Each time the PowerCube is processed, a check is performed to ensure that the latest labels are being used.

The screenshot shows a dialog box titled "Level - Branch Key". At the top, there is a text field labeled "Level name:" containing the text "Branch Key". Below this is a tabbed interface with tabs for "Source", "General", "Time", "Order By", "Allocation", and "Description". The "Allocation" tab is currently selected. Inside this tab, there is a section titled "Associations" which contains a table with two columns: "Role" and "Column Name". The table has two rows: "Source" with "Branch Key" and "Label" with "Branch Name". Below the table are "Add" and "Delete" buttons. Further down, there is a "Categories" section with a checked "Unique" checkbox and an unchecked "Move" checkbox. Below that is a "Refresh" section with a checked "Label" checkbox, and two unchecked checkboxes for "Description" and "Short Name". At the bottom of the dialog are "OK", "Cancel", and "Help" buttons. A red circle highlights the "Refresh" section, specifically the "Label" checkbox.

Role	Column Name
Source	Branch Key
Label	Branch Name

Figure 13-55 Completed role associations



### Alternate drill-down paths

Alternate drill-down paths in a dimension provide a different perspective of the data that it contains. Each alternate path connects to the primary path at a convergence level.

Figure 13-56 shows an example of one of the alternate drill-down paths that we create for our scenario. This figure highlights that the Branch Type field is dragged to the All Branches dimension. Drag the field to the right of the existing Region and Nation levels for Transformer to give you an indication of where the alternate path and convergence level will be created.

After you drop the field, confirm the action and click **No** to the following question from Transformer:

“You made level 'Branch Key' into a convergence level that connects two or more alternate drill-down paths. Categories in convergence levels must have unique source values. Can there be two categories in this level with the same source value?”

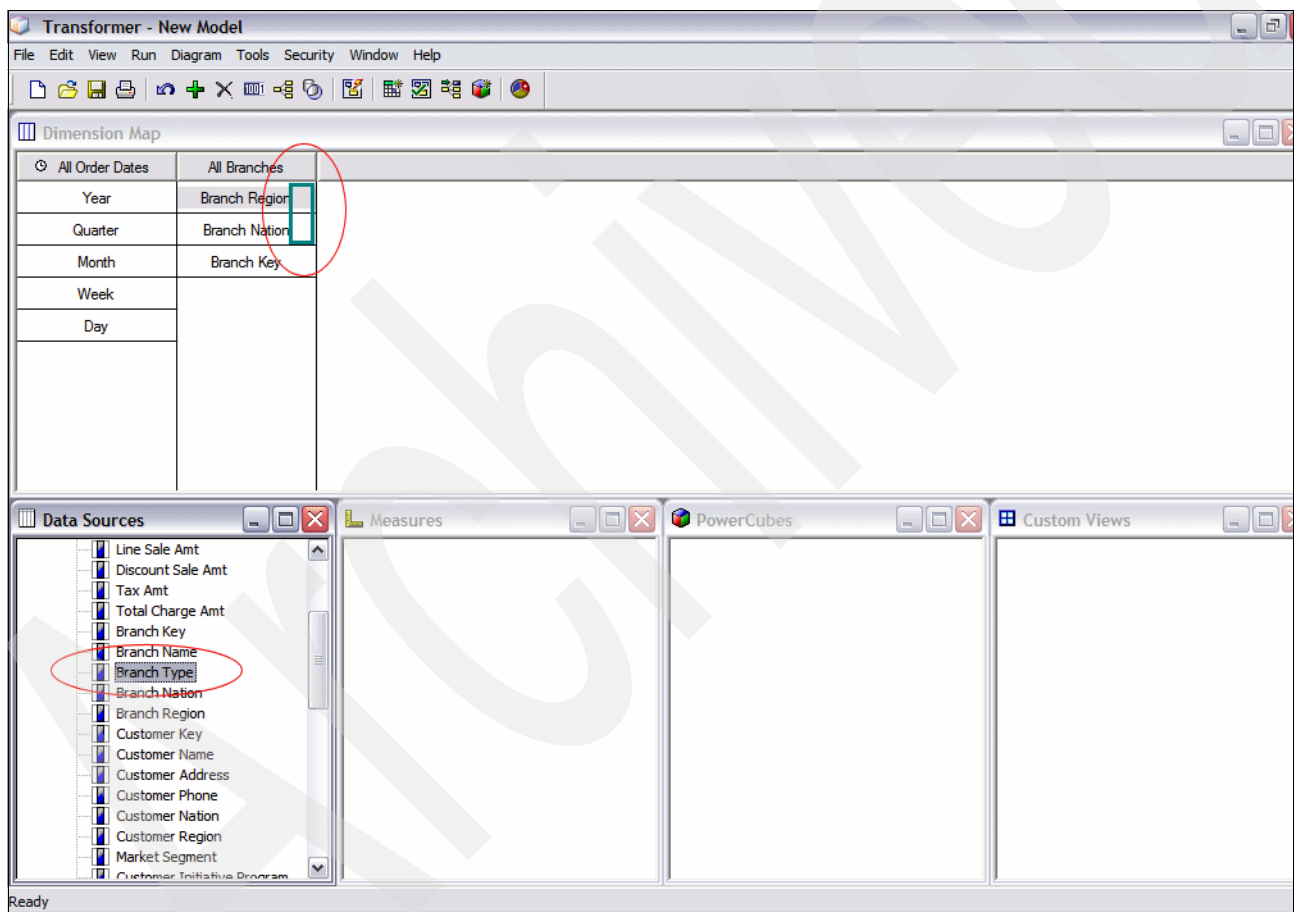


Figure 13-56 Defining an alternate drill-down path

## Insert measures

Drag measures from the Data Sources pane to the Measures pane. Figure 13-58 on page 363 shows the list of completed measures.

## Calculated measures

Figure 13-57 shows an example of defining the calculated measure Return Rate %. To access this window, right-click in the Measures pane and select **Insert Measure**. This window also shows that the output scale for the calculated column is set to the value of 2. Output scale is required to get the correct result after the percent function is applied later. Additional information to define the calculated measure is required on the Type, Rollup, and Format pages as follows:

- ▶ The Type page contains the calculation percent (Returned Units, Ordered Units).
- ▶ The Rollup page contains the setting for the Regular Timing property, for which you must select After Rollup. This setting ensures the calculation is performed at dimension levels after the measures Returned Units and Ordered Units are aggregated.
- ▶ The Format tab contains the settings to format the result with a percent sign and two decimal places.

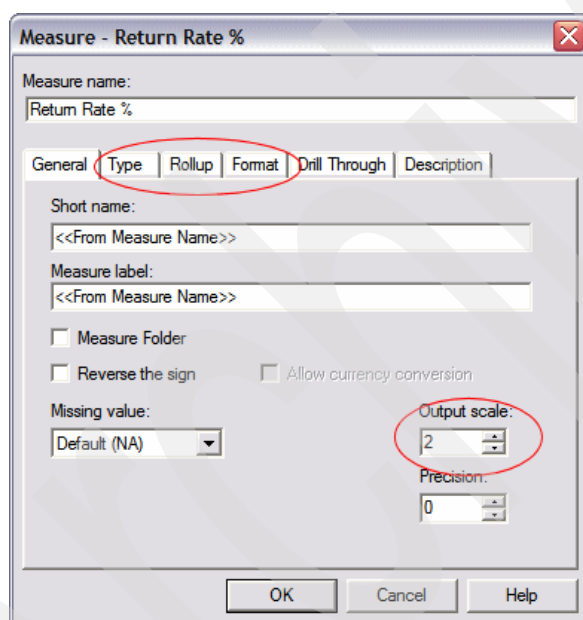


Figure 13-57 Defining a calculated measure

Figure 13-58 shows the final Transformer model for the RedParts scenario. When completed, we validate the model and generate categories by selecting **Run** → **Test Build**.

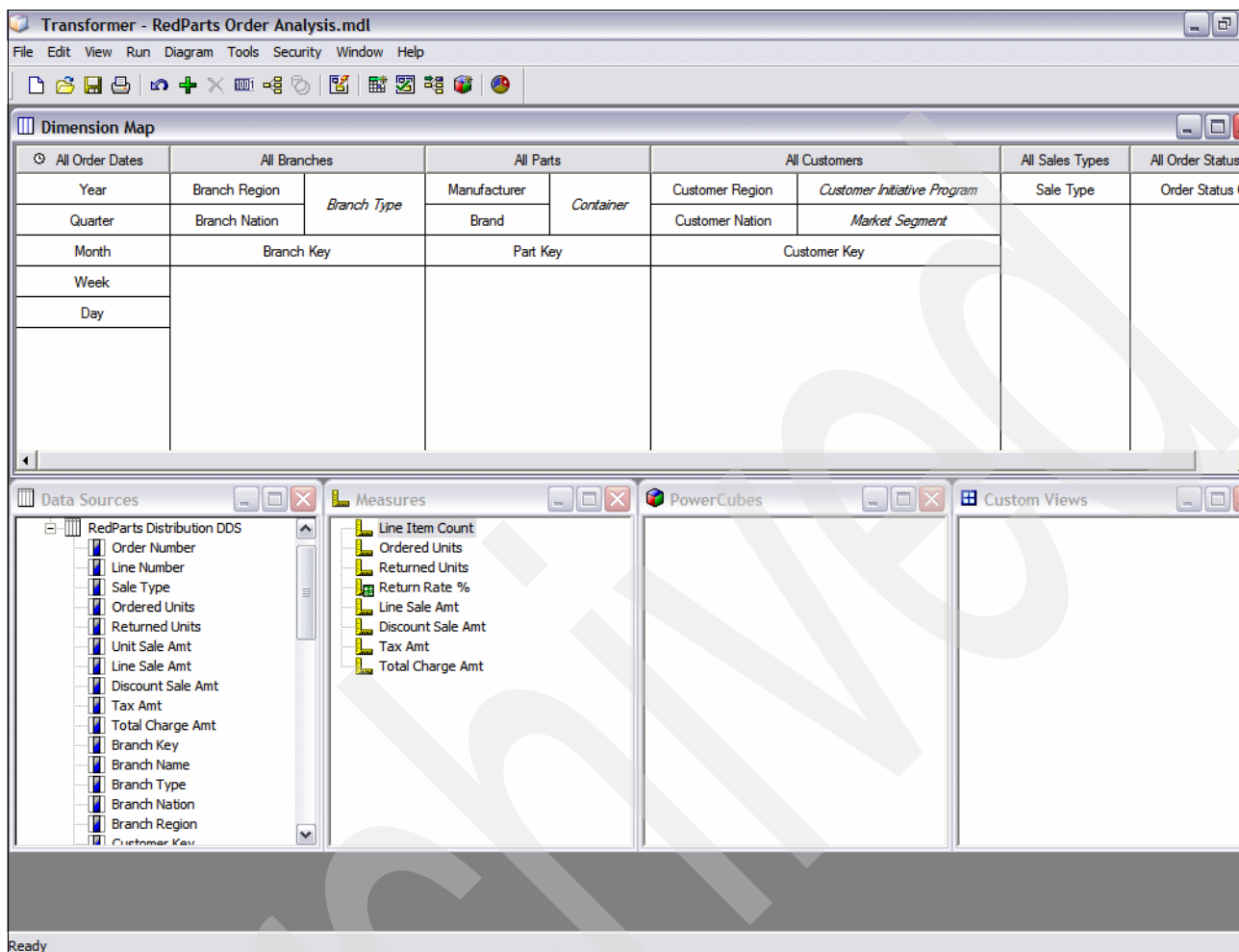


Figure 13-58 Completed Transformer model

### 13.3.3 Creating and publishing OLAP PowerCubes

In this section, we assume that a model file has been built in Transformer and sent from the Windows client on which it was developed to the server. Refer to 13.3.2, “Defining Transformer models for PowerCubes” on page 352, for information about the process to develop the model.

To start Cognos Transformer on the server with the model file RedParts Order Analysis.mdl and generate the PowerCube, enter the following command on the server:

```
cogtr -c -m"/opt/cognos/cube/RedParts Order Analysis.mdl"
```

The command creates a file (Example 13-21) that represents the OLAP cube, which is /opt/cognos/c8/temp/RedPart Order Analysis.mdc in our case, in the file system.

#### Example 13-21 Creating the PowerCube on the Server

---

Business Intelligence Transformer version .3.818.0  
Transformer(Transformer) Mon Apr 28 12:17:50 2008

```
LogFileDirectory=/opt/cognos/c8/logs/
ModelSaveDirectory=/opt/cognos/c8/temp/
DataSourceDirectory=/opt/cognos/c8/data/
CubeSaveDirectory=/opt/cognos/c8/temp/
DataWorkDirectory=/opt/cognos/c8/temp
ModelWorkDirectory=/opt/cognos/c8/temp/
MaxTransactionNum=500000

Mon 28 Apr 2008 12:17:50 Command Line: cogtr -c -m/opt/cognos/cube/RedParts Order Analysis.mdl [->OK]
Mon 28 Apr 2008 12:17:50 Processing MDL file /opt/cognos/cube/RedParts Order Analysis.mdl
Mon 28 Apr 2008 12:17:50 Creating model file /opt/cognos/c8/temp/ppd36928.qyj
Mon 28 Apr 2008 12:17:50 Completed processing of MDL file /opt/cognos/cube/RedParts Order Analysis.mdl
Mon 28 Apr 2008 12:17:50 Start cube update.
Mon 28 Apr 2008 12:17:50 No cube defined. Create default cube.
Mon 28 Apr 2008 12:17:50 Initializing categories.
Mon 28 Apr 2008 12:17:50 Timing, INITIALIZING CATEGORIES,00:00:00
Mon 28 Apr 2008 12:17:50 Start processing data source 'RedParts Distribution DDS'.
Mon 28 Apr 2008 12:17:50 Reading source data.
Mon 28 Apr 2008 12:17:56 Timing, OPEN DATA SOURCE,00:00:06
Mon 28 Apr 2008 12:17:58 End processing 18000 records from data source 'RedParts Distribution DDS'.
Mon 28 Apr 2008 12:17:58 Timing, READ DATA SOURCE,00:00:08
Mon 28 Apr 2008 12:17:58 Marking categories used.
Mon 28 Apr 2008 12:17:58 Timing, MARKING CATEGORIES USED,00:00:00
Mon 28 Apr 2008 12:17:58 Updating category status.
Mon 28 Apr 2008 12:17:58 Processing the work file.
Mon 28 Apr 2008 12:17:58 Processing cube 'RedPart Order Analysis' at location /opt/cognos/c8/temp/RedPart Order Analysis.mdc
Mon 28 Apr 2008 12:17:58 Timing, UPDATE CATEGORY AND PROCESS WORK FILE,00:00:00
Mon 28 Apr 2008 12:17:58 Start metadata update of cube 'RedPart Order Analysis'.
Mon 28 Apr 2008 12:17:58 Marking categories needed.
Mon 28 Apr 2008 12:17:58 Updating the PowerCube metadata.
Mon 28 Apr 2008 12:17:58 Updating the PowerCube with currency data.
Mon 28 Apr 2008 12:17:58 End metadata update of cube 'RedPart Order Analysis'. 1675 categories were added to the cube.
Mon 28 Apr 2008 12:17:58 Timing, METADATA,00:00:00
Mon 28 Apr 2008 12:17:58 Start update of cube 'RedPart Order Analysis'.
Mon 28 Apr 2008 12:17:58 --- Performing Pass 0 with 18000 rows and 1675 categories remaining.
Mon 28 Apr 2008 12:17:58 Start Write leaving 1675 categories remaining.
Mon 28 Apr 2008 12:17:58 Updating the PowerCube data.
Mon 28 Apr 2008 12:17:58 Updating the PowerCube data.
Mon 28 Apr 2008 12:17:59 Performing DataBase Commit at record number 18001.
Mon 28 Apr 2008 12:17:59 End Write leaving 1675 categories remaining..
Mon 28 Apr 2008 12:17:59 Timing, CUBE UPDATE,00:00:01
Mon 28 Apr 2008 12:17:59 Committing PowerCube(s).
Mon 28 Apr 2008 12:17:59 Timing, CUBE COMMIT,00:00:00
Mon 28 Apr 2008 12:17:59 Updating the PowerCube with optimized metadata.
Mon 28 Apr 2008 12:17:59 Start update of cube '/opt/cognos/c8/temp/RedPart Order Analysis.mdc' with optimized metadata.
Mon 28 Apr 2008 12:17:59 End update of cube '/opt/cognos/c8/temp/RedPart Order Analysis.mdc' with optimized metadata.
Mon 28 Apr 2008 12:17:59 End cube update.
Mon 28 Apr 2008 12:17:59 Timing, TOTAL TIME (CREATE CUBE),00:00:09
Mon 28 Apr 2008 12:17:59 Closing model file /opt/cognos/c8/temp/ppd36928.qyj
Mon 28 Apr 2008 12:17:59 Transformer exiting - OK
```

---

## Creating a data source for the PowerCube

After we create the mdc file, we copy it from the temporary directory to its target directory:

```
cp /opt/cognos/c8/temp/*.mdc /opt/cognos/cube
```

Then we create a Cognos data source and package for further use in the Cognos Connection portal. You can also perform this step by using the commanding tool (**cogtr**). In the following steps, we create the data source by using the Web Cognos Connection user interface:

1. In the Cognos Web interface, click **Cognos Administration** → **Data Source Connections** and click the icon to create a new data source definition (circled in Figure 13-59).

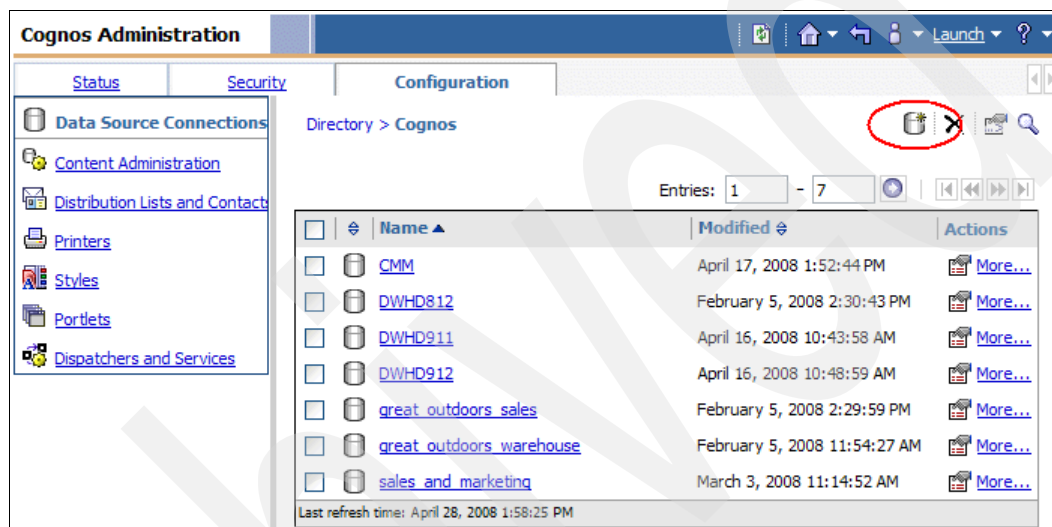


Figure 13-59 Starting to create a new data source for the Power Cube

2. In the New Data Source wizard, type the name RedPart Order Analysis and click **Next**.
3. In the Specify the connection panel (Figure 13-60), for Type, select **Cognos PowerCube** and click **Next**.

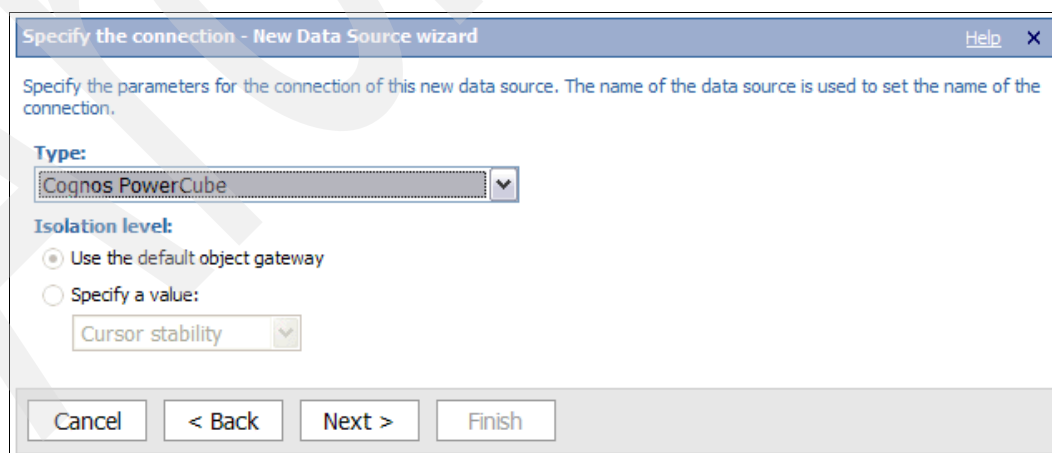


Figure 13-60 Selecting the data source type

4. In the Specify the Cognos PowerCube connection string panel (Figure 13-61), specify the file system location of the PowerCube file created previously (/opt/cognos/cube/RedPart Order Analysis.mdc).

**Note:** When we performed this step, we inserted a value in the Windows location field as well. By not doing this, we received the following error message:

The field “Windows location:” is mandatory. Please enter a value.

The Windows location is not used for processing, but is required to move past this step.

Then click **Finish**.

Specify the Cognos PowerCube connection string - New Data Source wizard

Edit the parameters to build a Cognos PowerCube connection string.

**Read cache size (MB):**

**Location**

Specify the location of the cube on a Windows operating system. If the cube also resides on a Unix or Linux operating system, enter its location.

**Windows location:**

d:\tmp

**Unix or Linux location:**

/opt/cognos/cube/RedPart Order Analysis.mdc

**Signon**

Select an authentication method.

☒ All applicable namespaces (including unsecured PowerCubes)

☐ Restrict PowerCube authentication to a single namespace

Select whether a cube password is needed.

☐ Cube password

☒ Create a signon that the Everyone group can use:

**Password:**

**Confirm password:**

**Testing**

[Test the connection...](#)

Cancel < Back Next > Finish

Figure 13-61 Specifying the path to the mdc file for the PowerCube definition

After the data source wizard creates the data source successfully, it offers to create a package that allows users to use the cube as a source for analysis.

5. Select the check box to **Create a package** since we need an additional package for analysis later and click **OK**.
6. In the Specify the name and description panel (Figure 13-62), enter the name RedPart Order (Analysis) and select a location (folder) for the new package. For our scenario, we create the new package in the folder **Public Folders > Redbook Scenario - RedParts Distribution Warehouse**. Click **Finish**.

Specify the name and description - New Package wizard

Specify a name and location for this entry. You can also specify a description and screen tip.

**Name:**  
RedPart Order (Analysis)

**Description:**  
[Empty text area]

**Screen tip:**  
[Empty text area]

**Location:**  
Public Folders > Redbook Scenario - RedParts Distribution Warehouse  
[Select another location...](#) [Select My Folders](#)

Buttons: Cancel, < Back, Next >, Finish

Figure 13-62 Creating a new package for analysis

This completes the creating and publishing of the new PowerCube for our analysis. We have a new data source named RedPart Order Analysis that points to our PowerCube mdc file in the file system and a new package that we can use within Analysis Studio. Our analysis example is in 13.6, “Multidimensional analysis with Analysis Studio” on page 385.

## 13.4 Reports with Report Studio

In this section, we show how to create a simple sample report based on the data in our ODS. We use a slightly modified native SQL query from the queries that TPC-H provides. See Example 13-22 on page 368. We base our report on a business requirement listed in Table 5-2 on page 73.

**Important:** In this example, we copy the native SQL shown directly into Report Studio. *This method is not a best practice nor do we recommend it.* Generate queries for reports interactively by selecting objects and functions from the published package using the drag-and-drop method. By doing this, Cognos can determine the most efficient query to generate selected results, provide more value to the user, and allow report authoring to be done by those who do not have SQL skills. A similar example of using the drag-and-drop method is shown in 13.5, “Ad hoc queries with Query Studio” on page 374. The purpose of this chapter is to demonstrate that a report can be created in Cognos 8 BI based on data stored in our ODS that is hosted by DB2 on z/OS.

*Example 13-22 Returned item query*

---

```
select
  c_custkey,
  c_name,
  sum(l_extendedprice * (1 - l_discount)) as revenue,
  c_acctbal,
  n_name,
  c_address,
  c_phone,
  c_comment
from
  dwhods.customer,
  dwhods.orders,
  dwhods.lineitem,
  dwhods.nation_lookup
where
  c_custkey = o_custkey
  and l_orderkey = o_orderkey
  and l_returnflag = '1'
  and c_nationkey = n_nationkey
group by
  c_custkey,
  c_name,
  c_acctbal,
  c_phone,
  n_name,
  c_address,
  c_comment
order by
  revenue desc
```

---

According to the description in the TPC-H benchmark,<sup>2</sup> this query returns the following information:

“... customers, in terms of their effect on lost revenue ... who have returned parts. The query lists the customer’s name, address, nation, phone number, account balance, comment information, and revenue lost. The customers are listed in descending order of lost revenue. Revenue lost is defined as  $\text{sum}(\text{l\_extendedprice} * (1 - \text{l\_discount}))$  for all qualifying line items.”

In the following steps, we assume that the RedParts Distribution ODS (Query) package with our ODS model information is already created and published to the Cognos BI Server. See 13.3.1, “Framework Manager packages” on page 337.

---

<sup>2</sup> Transaction Processing Performance Council (TPC): [http://www.tpc.org/tpch/results/tpch\\_perf\\_results.asp](http://www.tpc.org/tpch/results/tpch_perf_results.asp)



1. In the Cognos 8 BI Server portal, in the My Actions panel (Figure 13-63), choose **Create professional reports** to start the Report Studio.

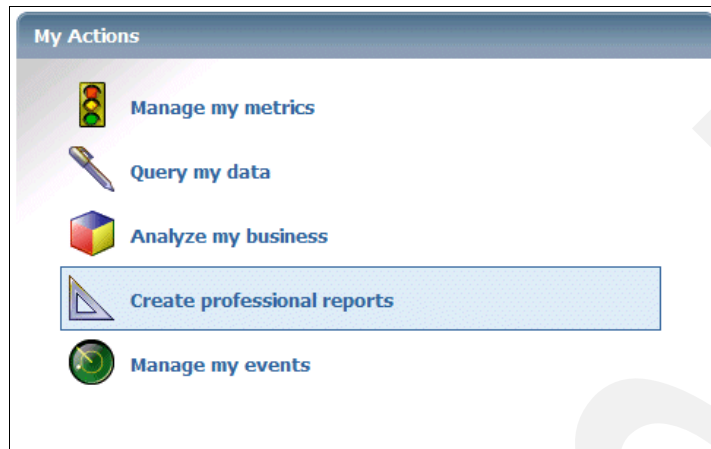


Figure 13-63 Starting the Report Studio from the My Actions panel

2. In the next window (Figure 13-64), select a package that contains the metadata on which the reports should be built. Select **RedParts Distribution ODS (Query)**.

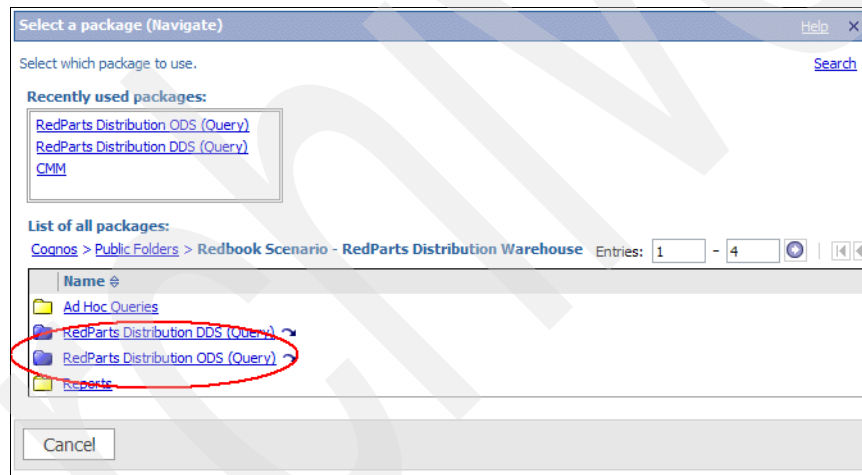


Figure 13-64 Package selection for the new report

3. In the welcome panel (Figure 13-65) of the full-screen, Report Studio browser window, select **Create a new report or template** to create a new report.



Figure 13-65 Report Studio welcome panel

**Note:** Report Studio (and other studios as well) require Internet Explorer® 6.0 or higher. If you try to run Report Studio with an unsupported browser, you receive an error message.

4. In the New panel (Figure 13-66), select the type of report to be created. For our simple report, select **Blank** and click **OK** to confirm the selection.

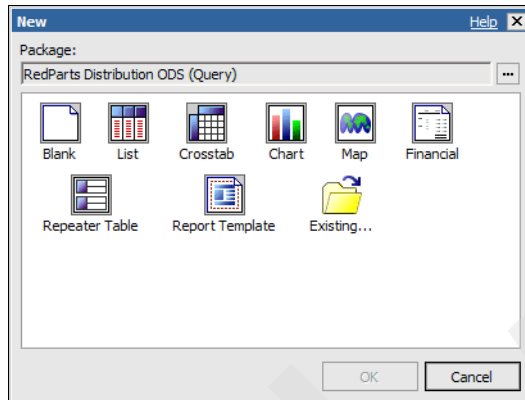


Figure 13-66 Report template selection in Report Studio

5. In the next panel, select **Query Explorer** and then **Queries** to display the available queries. Drag the **SQL** object from the Insertable Objects list to the explorer view (Figure 13-67). Both a new SQL icon and a new query icon, named Query 1, are created in the view. The vertical bar between the properties panels on the left side and the view panel on the right side offers explorers for pages, queries, and conditions.

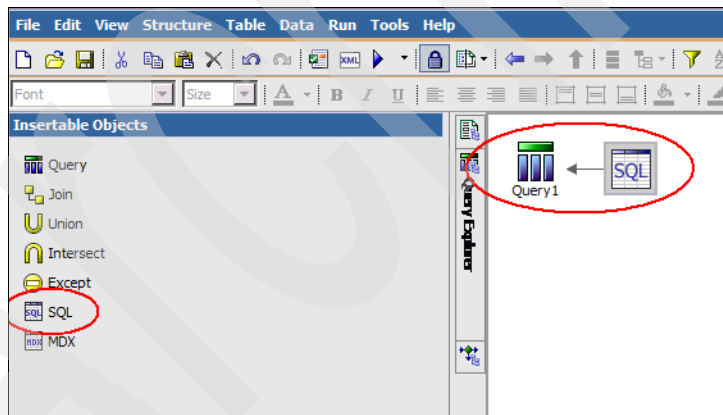


Figure 13-67 Defining a new SQL-based query object

6. Make sure that the SQL icon in the query explorer is selected and edit the properties panel (Figure 13-68):
  - a. Select the Data Source property for your SQL query (**DWDH912** in our case) from the list of data sources that are configured through Cognos Administration on the Cognos BI Server.
  - b. Click the ... (ellipses) button for the SQL property text to copy the SQL statement from Example 13-22 on page 368 into the SQL editor window.
  - c. In the Name property field, rename the SQL query to Q10.

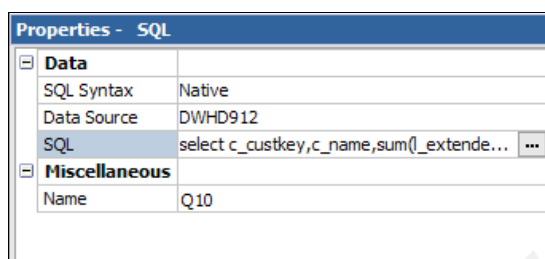


Figure 13-68 Properties for the SQL query in Report Studio

**Note:** When entering SQL in the SQL property, if the SQL you enter contains constructs that DB2 does not allow in a derived table, you must change the SQL Syntax property from Native to Pass-Through. This is not required in this particular example.

We recommend that you allow Cognos to generate the most optimized SQL by using drag-and-drop functionality and not entering SQL text.

7. Depending on the complexity of the query and the capabilities of the database engine, provide query hints for query processing. Ensure that the Processing property is set to **Limited Local**. This should have been set as one of the data source properties in the Framework Manager model by default. With the setting of Limited Local, the Cognos query engine can perform some local processing instead of trying to run the entire operation in the database engine.

Select the **Query 1** icon to the left of the SQL icon and confirm the properties for the query. If not already selected, choose **Limited Local** for the Processing property in the Query Hints section (Figure 13-69).

The processing option specifies whether the query engine picks up a minimal amount of processing.

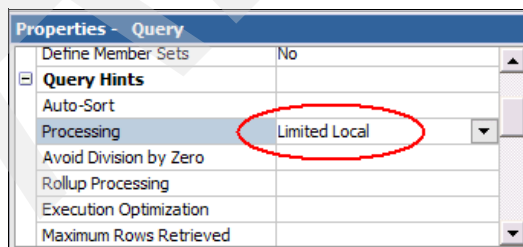


Figure 13-69 Setting query hints

8. Use the Page Explorer and select **Page 1** from the report pages. A blank window is displayed because we selected the Blank report template when creating the report.
  - a. In the second tab under the Insertable Objects panel, select the **Data Items** icon. The panel now looks as shown in Figure 13-70. Drag the **Query 1** object from the Insertable Objects pane to the pane on the right.

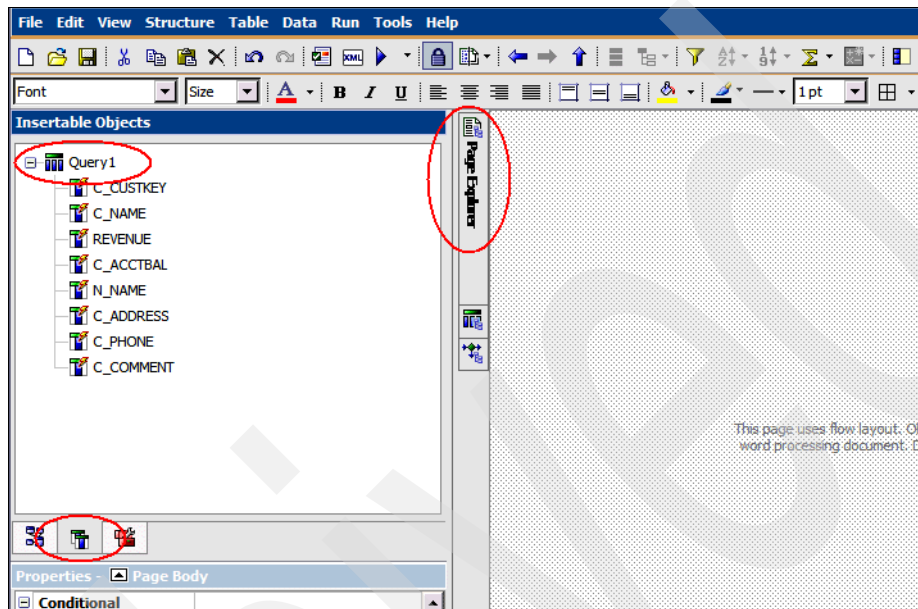


Figure 13-70 Selecting data items in Report Studio

- b. To add a heading for the report, at the bottom of the Insertable Objects pane on the left, click the third **Toolbox** tab and drag the **Text Item** object to just in front of the table with the query columns (Figure 13-71).

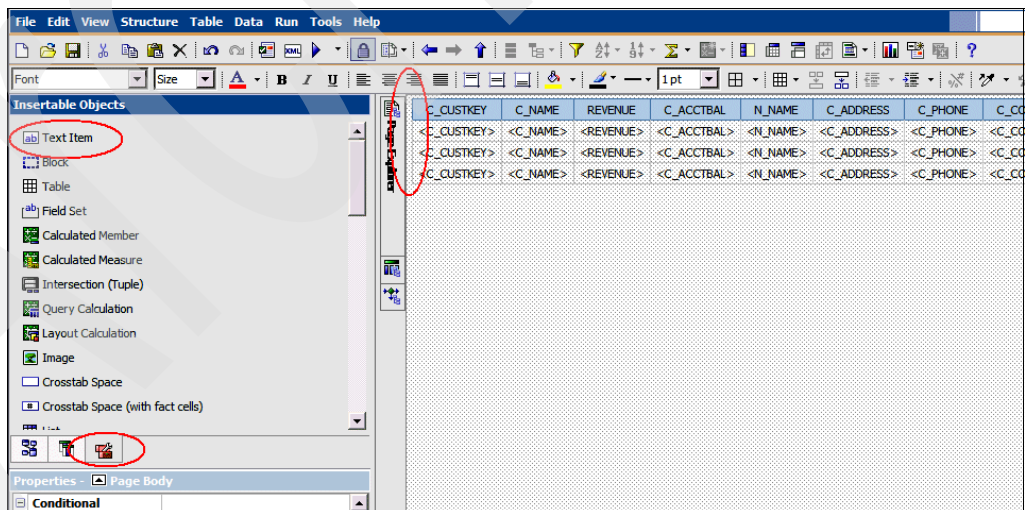


Figure 13-71 Adding a text item to the report

- c. In the new panel that opens, add Returned Item Reporting and click **OK**.

- d. Select the inserted table heading and use the toolbar tools to increase font size to 14 pt and select a bold font. Report Studio now looks like the example shown in Figure 13-72.

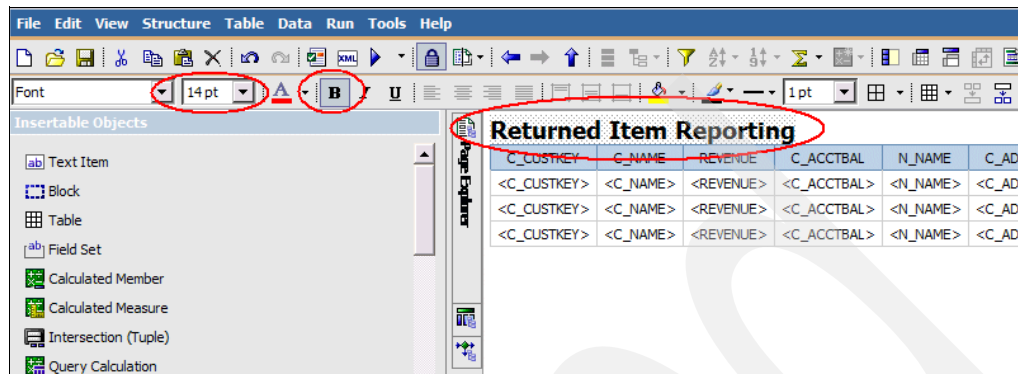


Figure 13-72 formatting the table heading

- e. Select **File** → **Save** from the menu and store the report definition (for example, with the name Returned Item Reporting) in the Cognos BI Server.

This completes the definition of the simple report. To test and run the report, click **Run Report - HTML** in the toolbar (Figure 13-73).

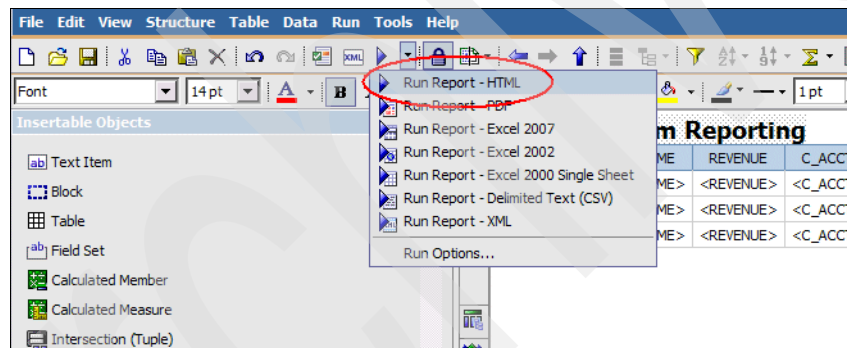


Figure 13-73 Running the defined report

Figure 13-74 shows the resulting report.

Cognos Viewer							
Returned Item Reporting							
C_CUSTKEY	C_NAME	REVENUE	C_ACCTBAL	N_NAME	C_ADDRESS	C_PHONE	C_COMMENT
213,343	Robert J R	30.5072	85,840	PERU	MM Park Lane68, 116836 C Town	555-0121-1091	
213,597	John I B	23.74	10,600	GERMANY	TG Highway7, 149914 V Town	555-0149-4024	
213,495	John D O	23.6118	52,110	UNITED STATES	FV Pkway56, 187587 Q Town	555-0188-3117	
212,697	Lothar Q Y	14.5068	78,610	KENYA	TX Road74, 115503 T Town	555-0166-6501	
213,075	Lothar A T	10.7082	10,510	INDIA	IL Ave4, 124988 Y Town	555-0119-9562	
213,301	Khadija S M	5.0699	96,990	MOROCCO	IP Crossing51, 162530 Q Town	555-0169-3068	
213,499	Sam X A	4.8977	78,940	CANADA	XZ Blvd13, 172805 F Town	555-0193-4092	
211,765	Frank F R	4.8922	65,940	JORDAN	RG Highway88, 184990 T Town	555-0121-2111	
213,065	Bob A Y	3.786	55,530	INDIA	DA Street43, 182165 Q Town	555-0131-4788	
212,711	Sreeni A W	0.6432	38,930	MOROCCO	PZ Street2, 186760 O Town	555-0112-8091	
213,005	Bob Z Q	0.3944	76,500	RUSSIA	AI Street10, 130027 C Town	555-0186-9173	
212,819	Cristian C M	-0.5224	62,460	ROMANIA	WS Blvd65, 187332 K Town	555-0183-3624	
211,937	Lothar R V	-1.5498	65,900	GERMANY	VE Pkway28, 182938 U Town	555-0172-7009	
212,243	Peter X E	-3.9183	88,510	JORDAN	YM Park Lane58, 129344 V Town	555-0186-8378	
212,049	Sreeni V Y	-4.1544	20,950	ROMANIA	VC Pkway39, 174288 K Town	555-0136-9037	
211,831	Thomas N Z	-4.4668	82,430	UNITED STATES	UV Way69, 166815 B Town	555-0114-3524	
212,907	Robert I J	-5.0238	65,630	FRANCE	AY Blvd30, 154743 B Town	555-0198-8820	
211,955	Kirk Y W	-7.4284	65,490	FRANCE	XM Place3, 168206 M Town	555-0180-1674	
212,075	Paolo G C	-8.712	32,220	PERU	MY Road83, 125931 K Town	555-0147-4756	
213,419	Sam U P	-11.3332	39,970	RUSSIA	FR Crossing11, 151957 Z Town	555-0162-6248	

[Top](#)
[Page up](#)
[Page down](#)
[Bottom](#)

Figure 13-74 Result of the Returned Item report

## 13.5 Ad hoc queries with Query Studio

In this section, we show how to create a simple ad hoc query based on the data in our ODS by using Query Studio within Cognos 8 BI. We base our query on a business requirement listed in Table 5-2 on page 73.

The business requirement is to produce the top volume customer query. This query retrieves a list of customers based on them having placed a large quantity order. For this sample, large quantity orders are defined as those orders where total quantity is greater than or equal to 18. The query lists the customer name, nation, order key, order date, total price, and the quantity for the order. Total price is sorted in descending order.

In the following steps, we assume that the package RedParts Distribution ODS (Query) with our ODS model information is already created and published to the Cognos BI Server. See 13.3.1, “Framework Manager packages” on page 337.

1. In the Cognos 8 portal, select **Launch** → **Query Studio**, which is highlighted in Figure 13-75.

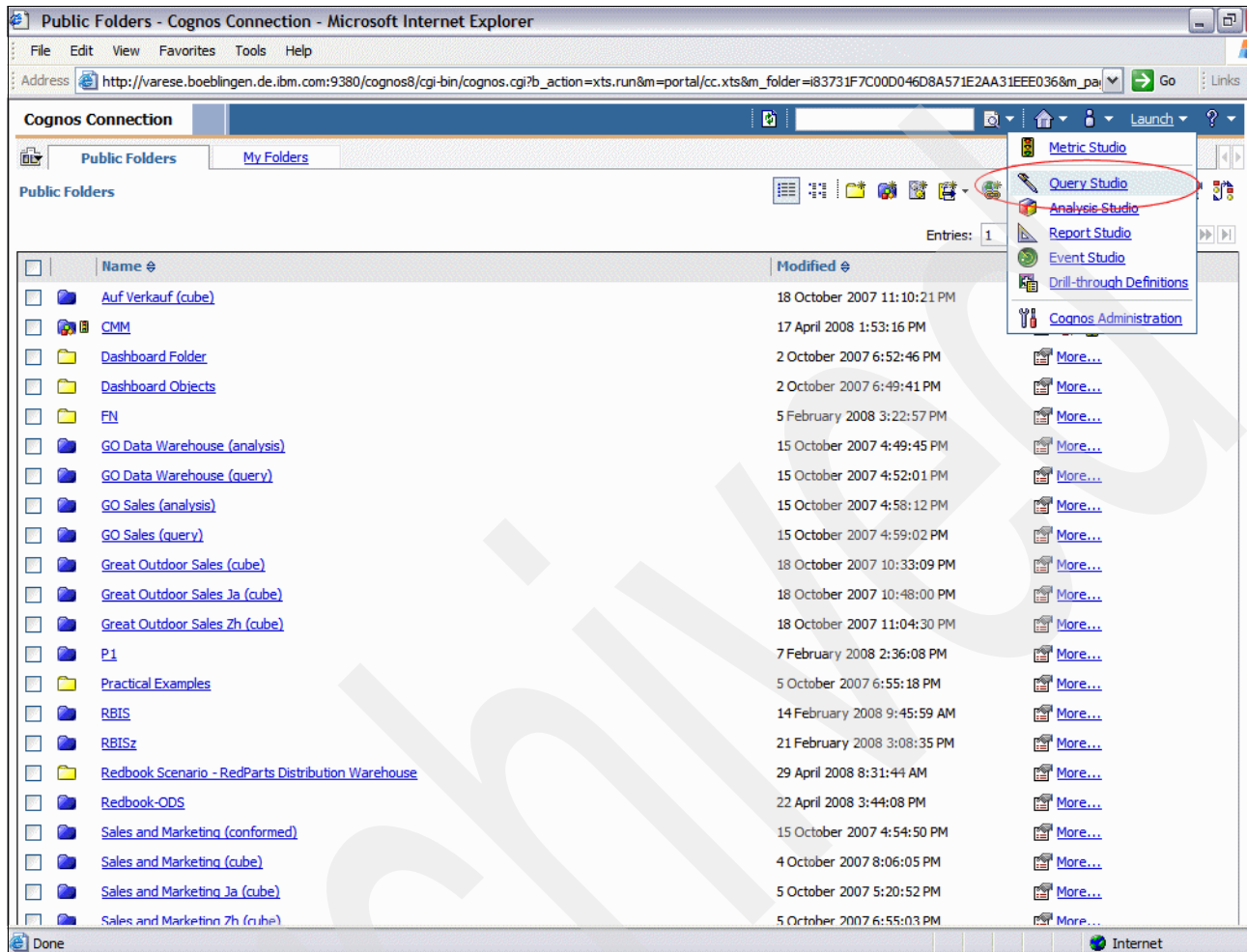


Figure 13-75 Navigating to Query Studio



2. In the Select a package window (Figure 13-76), select the package that contains the business metadata for users to select query items from. Click the **RedParts Distribution ODS (Query)** package to open a blank Query Studio canvas.

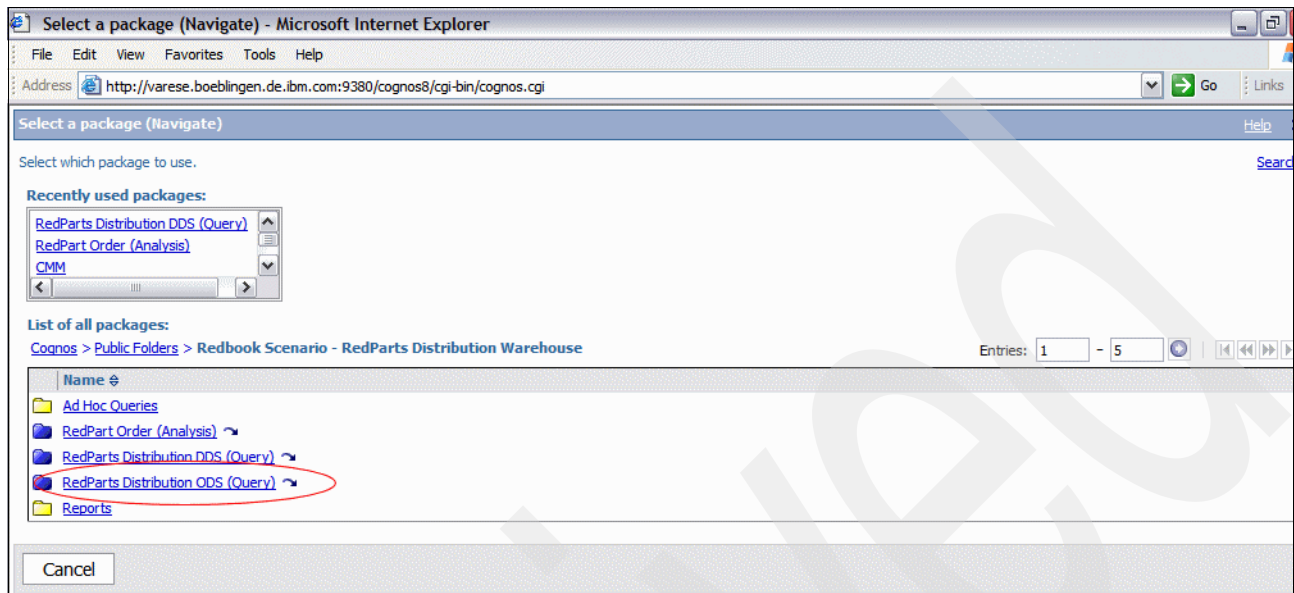


Figure 13-76 Query - Selecting the source package



3. In the Business view (Figure 13-77), expand the query subject **Order Details - All Orders**. Press Ctrl+click to select the query items **Order Number**, **Order Date**, **Customer Name**, **Unit Quantity**, **Total Charge Amt** and then drag them to the blank canvas. The selected query items are displayed in the query along with the result set.

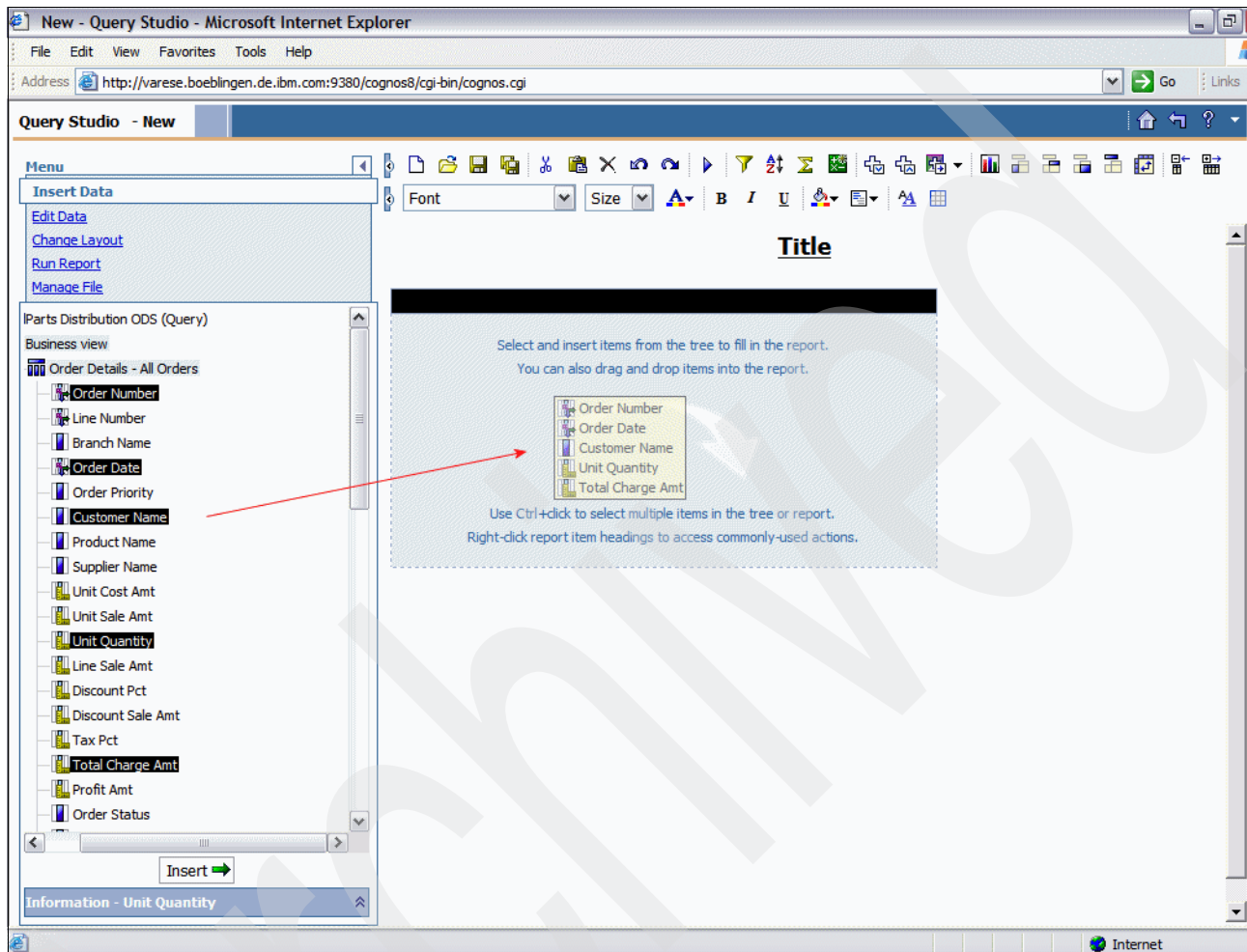


Figure 13-77 Query - Dragging the order query items

- Expand the query subject **Customer**, click the query item **Nation**, and drag the query item to the canvas to the left side of the existing Customer Name query item (Figure 13-78). Nation is displayed in the result set.

The screenshot shows the Query Studio interface in Microsoft Internet Explorer. The left pane displays the 'Customer' query item expanded, showing sub-items: Customer Name, Customer Address, Nation, Region, Contact Phone, Market Segment, and Current Account Balance. A red arrow indicates the 'Nation' item being dragged to the canvas. The main canvas displays a table titled 'Title' with the following data:

Customer Name	Order Number	Order Date	Total Charge Amt	Unit Quantity
Andrew A S	29866	23/10/2007	\$13,309.56	15
Andrew A S	30842	24/08/2007	\$9,224.77	11
Andrew A S	35590	11/07/2007	\$5,272.53	6
Andrew A S	35926	15/11/2007	\$2,262.00	3
Andrew A S	36108	19/02/2007	\$1,895.35	2
Andrew A S	4140747	18/11/2007	\$15,163.91	17
Andrew A S	4140973	16/08/2007	\$9,100.16	10
Andrew A S	4141339	24/09/2007	\$6,212.93	7
Andrew A S	4141363	04/10/2007	\$8,102.38	10
Andrew A S	4144279	27/04/2007	\$14,736.24	16
Andrew A S	4144653	17/10/2007	\$1,457.71	2
Andrew A S	4151653	24/02/2007	\$5,424.07	6
Andrew A V	30034	14/11/2007	\$5,150.77	6
Andrew A V	31634	10/10/2007	\$8,913.75	10
Andrew A V	40318	13/04/2007	\$5,848.75	7
Andrew A V	4141537	11/12/2007	\$5,395.40	7
Andrew A V	4145309	19/11/2007	\$897.97	1
Andrew A V	4146523	15/10/2007	\$4,931.82	6
Andrew A V	4147085	10/02/2007	\$6,671.71	9
Andrew A V	4149549	04/10/2007	\$8,402.55	10

Figure 13-78 Query - Dragging the customer nation

- For the top volume customers query, group the results by Nation. Click the **Nation** header as shown in Figure 13-79 and then click the **Group** toolbar button to perform the grouping.

The screenshot shows the Cognos Query Studio interface. On the left, the 'Menu' pane includes options like 'Insert Data', 'Edit Data', 'Change Layout', 'Run Report', and 'Manage File'. Below this is a tree view of the query structure, showing 'Business view' with sub-items like 'Order Details - All Orders', 'Order Details - Returned Products', 'Customer', 'Parts', and 'Supplier'. The 'Customer' item is expanded, showing 'Customer Name', 'Customer Address', 'Nation', 'Region', 'Contact Phone', and 'Market Segment'. The 'Parts' item is also expanded, showing 'Part Number', 'Part Name', 'Manufacturer', 'Brand', 'Part Type', 'Part Size', 'Type of Container', and 'Unit Retail Amt'. The 'Supplier' item is also expanded, showing 'Supplier'. The 'Information - Total Charge Amt' pane is at the bottom left. The main area displays a table with the following data:

Nation	Customer Name	Order Number	Order Date	Total Charge Amt	Unit Quantity
ALGERIA	Andrew Q C	32112	19/05/2007	\$8,008.78	10
ALGERIA	Andrew Q C	36632	06/05/2007	\$5,524.46	7
ALGERIA	Andrew Q C	40582	08/07/2007	\$4,948.58	6
ALGERIA	Andrew Q C	4147739	28/01/2007	\$6,416.78	8
ALGERIA	Andrew Q C	4147747	16/04/2007	\$7,433.42	9
ALGERIA	Andrew Q C	4152087	21/03/2007	\$9,908.63	11
ALGERIA	Andrew Q C	4152475	15/03/2007	\$12,059.24	15
ALGERIA	Bob K M	32372	18/11/2007	\$9,950.99	11
ALGERIA	Bob K M	34748	22/12/2007	\$3,100.70	4
ALGERIA	Bob K M	35462	11/02/2007	\$7,415.36	9
ALGERIA	Bob K M	35972	08/02/2007	\$4,056.60	5
ALGERIA	Bob K M	4142359	06/04/2007	\$11,250.09	12
ALGERIA	Bob K M	4145297	08/04/2007	\$3,418.26	4
ALGERIA	Bob K M	4147441	04/01/2007	\$2,280.48	3
ALGERIA	Bob K M	4147681	07/06/2007	\$8,927.36	9
ALGERIA	Bob K M	4147889	17/07/2007	\$6,249.40	9
ALGERIA	Bob K M	4152023	27/11/2007	\$3,409.99	4
ALGERIA	Bob W S	29028	20/03/2007	\$11,144.85	13
ALGERIA	Bob W S	30076	18/07/2007	\$12,255.84	15
ALGERIA	Bob W S	30308	18/11/2007	\$9,241.86	12

The 'Nation' column header is highlighted in yellow. The 'Group' button in the toolbar is circled in red. The date '07/05/2008' is displayed at the bottom left, and the page number '- 1 -' is at the bottom right.

Figure 13-79 Query - Grouping by nation

- To sort the Total Charge Amt in descending order, right-click the **Total Charge Amt** header and select **Sort**. The sort options pane is displayed at the bottom of the browser window. Figure 13-80 shows that the sort options of Descending and Sort on report details are selected.

The screenshot shows the Query Studio interface in a Microsoft Internet Explorer browser window. The main area displays a table with the following data:

Nation	Customer Name	Order Number	Order Date	Total Charge Amt	Unit Quantity
ALGERIA	Andrew Q C	32112	19/05/2007	\$8,008.78	10
	Andrew Q C	36632	06/05/2007	\$5,524.46	7
	Andrew Q C	40582	08/07/2007	\$4,948.58	6
	Andrew Q C	4147739	28/01/2007	\$6,416.78	8
	Andrew Q C	4147747	16/04/2007	\$7,433.42	9
	Andrew Q C	4152087	21/03/2007	\$9,908.63	11
	Andrew Q C	4152475	15/03/2007	\$12,059.24	15
	Bob K M	32372	18/11/2007	\$9,950.99	11
	Bob K M	34748	22/12/2007	\$3,100.70	4
	Bob K M	35462	11/02/2007	\$7,415.36	9

Below the table, a 'Sort' dialog box is open. It contains the following options:

- Sort on:** Total Charge Amt
- Sort order:**
  - ☒ Descending (9 to 1)
  - ☐ Ascending (1 to 9)
  - ☐ Don't sort
- Apply the sort to:**
  - ☒ Based on report details
  - ☐ Based on group summaries:

The 'Sort' dialog box also includes an 'OK' button and a 'Cancel' button.

Figure 13-80 Query - Sorting on descending order

- For this query, return orders that have a status of shipped and packed. In the Insert Data tree structure (Figure 13-81), expand the **Filters** folder to view the available filters. Click the **Order Status - Shipped & Packed** filter and drag it to the query canvas.

The screenshot shows the Cognos Query Studio interface in a Microsoft Internet Explorer browser window. The title bar reads "Large Volume Customers - Query Studio - Microsoft Internet Explorer". The address bar shows "http://varese.boeblingen.de.ibm.com:9380/cognos8/cgi-bin/cognos.cgi". The main window has a menu bar (File, Edit, View, Favorites, Tools, Help) and a toolbar. Below the menu bar is a "Query Studio - Large Volume Customers" header. The left pane contains a "Menu" section with links like "Insert Data", "Edit Data", "Change Layout", "Run Report", and "Manage File". Below the menu is a tree structure for "Current Account Balance" with sub-items like "Part Supplier List", "Part Number", "Part Name", "Manufacturer", "Branch", "Supplier Number", "Supplier Name", "Unit Supply Amt", and "Supplier Acct Bal". A "Filters" section is expanded, showing a list of filters: "Web Orders", "Branch Orders", "Returned Parts - All Reasons", "Returned Parts - Unsatisfactory", "Returned Parts - Damaged", "No Credit Customers - Account Balance under \$5K", "Order Status - Shipped & Packed" (highlighted with a red box and a red arrow pointing to the query canvas), "Order Status - Shipped", and "Order Status - Packed". Below the filters is a "Calculations" section with an "Insert" button. The right pane shows a query canvas with a "Title" field and a "Total Charge Amt: Descending order" filter. A table of data is displayed, with columns: "Nation", "Customer Name", "Order Number", "Order Date", "Total Charge Amt", and "Unit Quantity". The data is filtered by "Nation" = "ALGERIA". The bottom status bar shows the date "07/05/2008", page navigation links ("Top", "Page up", "Page down", "Bottom"), and the time "12:08:47 PM".

Nation	Customer Name	Order Number	Order Date	Total Charge Amt	Unit Quantity
ALGERIA	Paul H J	4152503	10/01/2007	\$16,180.88	18
ALGERIA	Gaurav R V	4144605	02/10/2007	\$15,672.11	18
ALGERIA	Frank B Z	4151579	20/12/2007	\$15,097.17	16
ALGERIA	Sreeni C D	28646	03/03/2007	\$14,906.29	16
ALGERIA	Sreeni X J	4143907	25/04/2007	\$14,718.62	16
ALGERIA	Khadija M T	4142289	07/12/2007	\$14,640.58	16
ALGERIA	Michael G B	4140857	17/01/2007	\$14,631.08	17
ALGERIA	Gaurav X Y	29916	17/12/2007	\$14,516.56	18
ALGERIA	Frank V T	4149491	04/08/2007	\$14,390.31	17
ALGERIA	Sreenivasa K D	4151691	18/06/2007	\$14,243.46	15
ALGERIA	Peter E B	4141527	02/02/2007	\$14,229.69	18
ALGERIA	Lothar W H	28800	19/03/2007	\$14,164.02	17
ALGERIA	Khadija L F	4141791	28/04/2007	\$13,953.19	17
ALGERIA	Khadija M E	4140711	04/08/2007	\$13,913.20	16
ALGERIA	Cristian X N	4141933	09/03/2007	\$13,758.31	15
ALGERIA	Khadija M T	4140981	15/10/2007	\$13,432.84	16
ALGERIA	Paolo I X	4151783	20/09/2007	\$13,265.94	17
ALGERIA	Frank U L	4151671	11/03/2007	\$13,055.24	15
ALGERIA	Peter E B	4152591	28/11/2007	\$13,004.11	14
ALGERIA	Peter W Y	4150823	18/06/2007	\$12,990.00	15

Figure 13-81 Query - Adding a query filter

8. To filter the results to show only customers whose total quantity is greater than or equal to 18 (Figure 13-82):

**Note:** Initially we demonstrate this step using the value 15 and then modify the filter to 18.

- a. Click the **Unit Quantity** header.
- b. Click the **Filter** toolbar button.
- c. In the Filter pane, in the From field, type 15.
- d. For Apply the filter to, select **Values in the report**.
- e. Click **OK**.

The filter is applied and listed under the query heading at the top of the page.

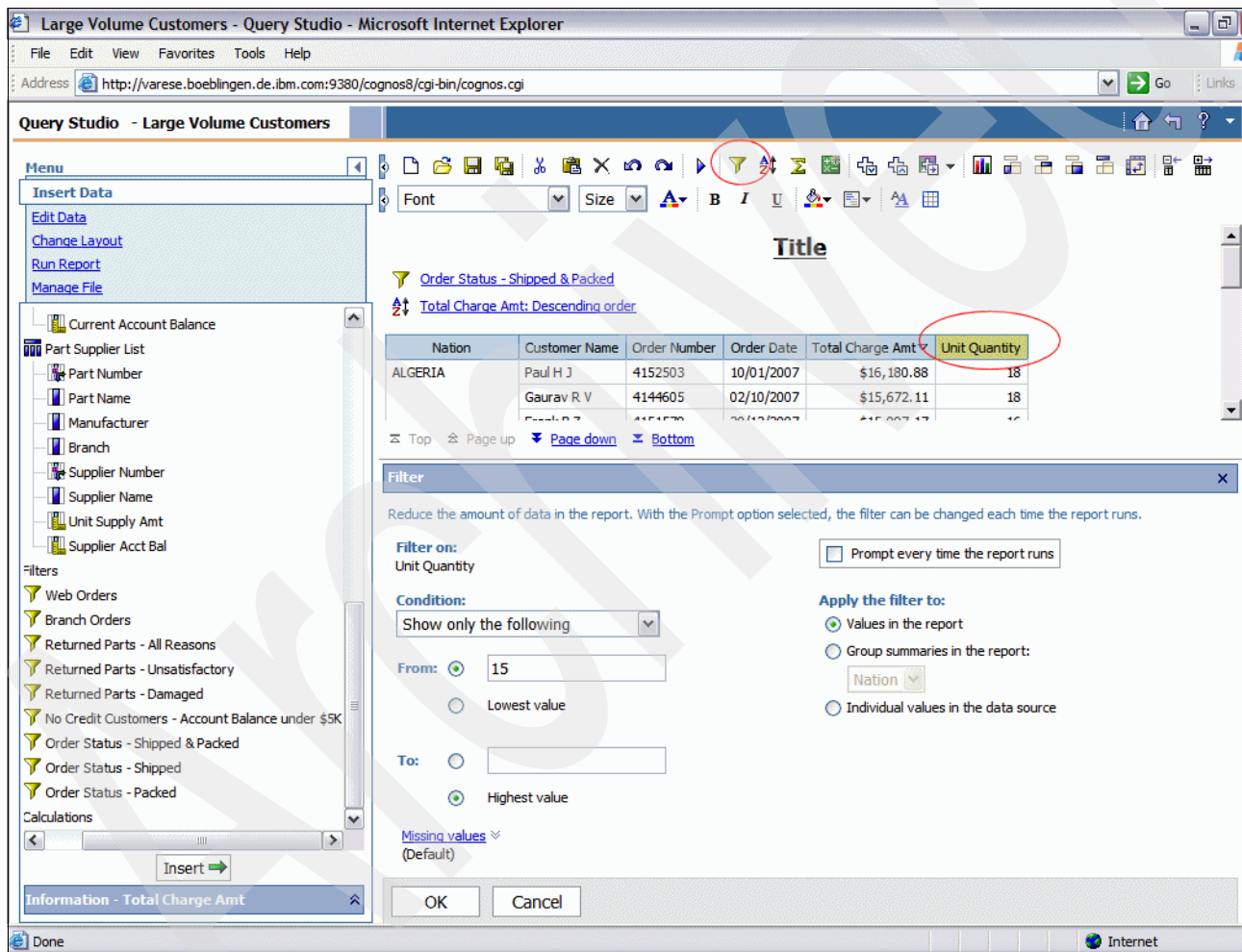


Figure 13-82 Query - Adding a column filter



9. Modify the filter to reduce the number of records returned (Figure 13-83):
  - a. Click the filter link **Unit Quantity: Greater than or Equal to 15** at the top of the query.
  - b. Change the value from 15 to 18 in the filter pane.
  - c. Click **OK**.

The number of records returned in the result set decreases.

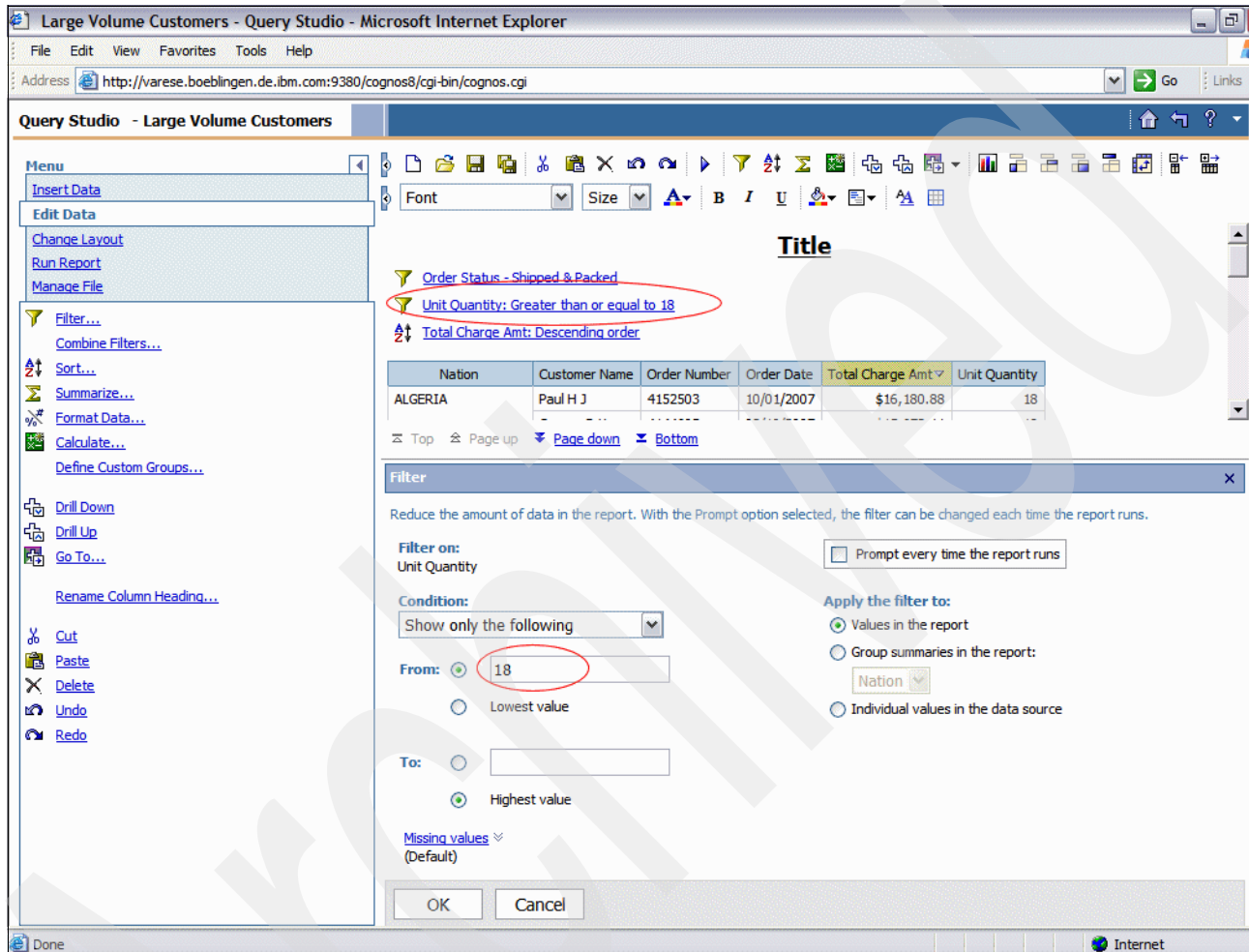


Figure 13-83 Query - Modifying the column filter

10. Right-click the **Total Charge Amt** field and select **Define Conditional Styles**. In the Define conditional styles pane, type 15,000 as the threshold and select the **Very Good** and **Average** styles for the highest and lowest definitions. Click **OK** to apply the conditional styles to the result set.

Different colors are shown in the Total Charge Amt field within the query, highlighting the relevant data that meets the defined conditional styles (Figure 13-84).

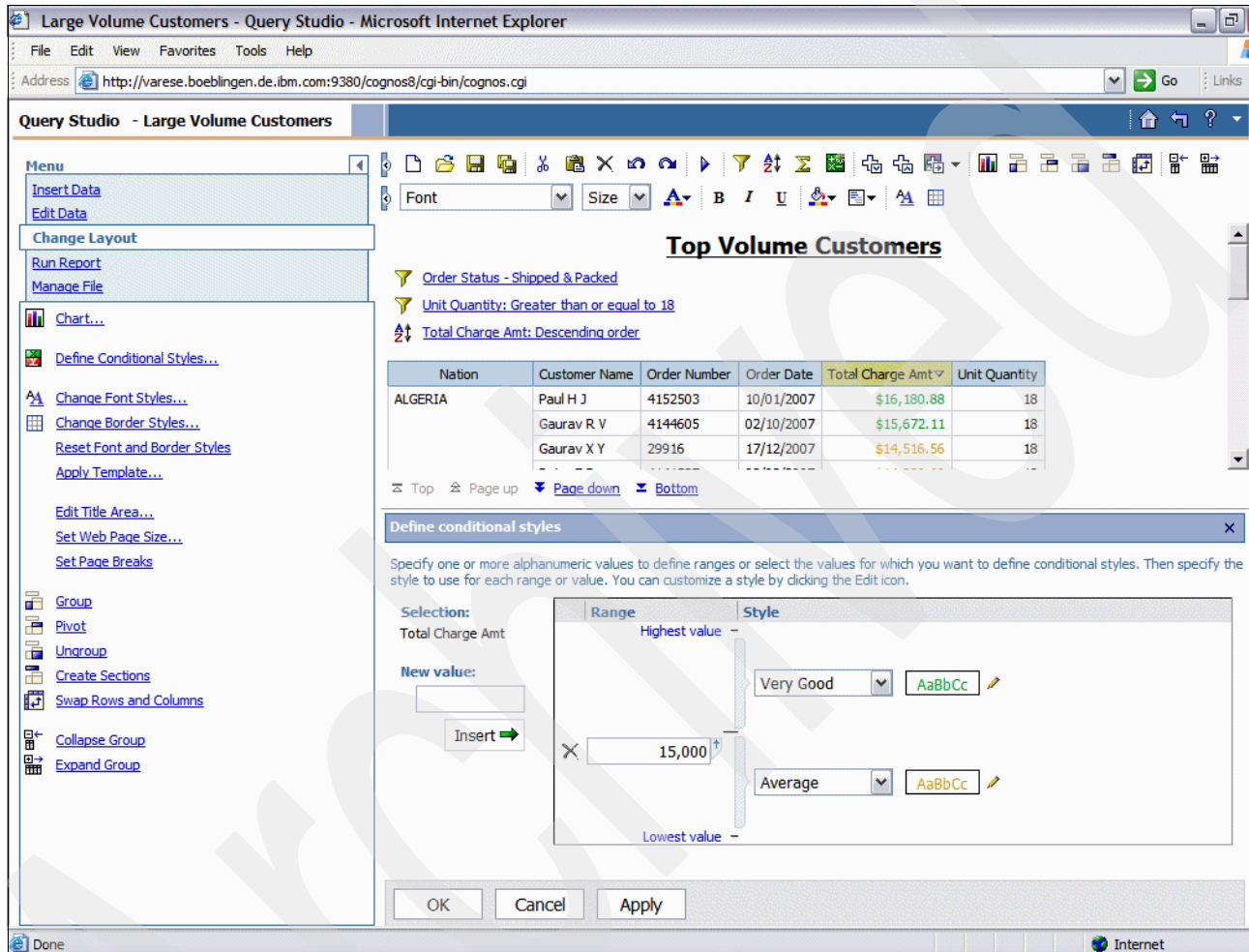


Figure 13-84 Query - Adding the conditional formatting

11. To modify the title of the query, click the default title once and enter the new title information into the Edit Title Area pane shown at the bottom of the browser.

The ad hoc query for top volume customers is now complete. Query Studio provides a number of options to run and share this result.



12. From the menu on the left side of the Query Studio interface, click **Run Report**. To demonstrate the option of exporting the query results to PDF, click the option **View in PDF Format**. Figure 13-85 shows the query result in PDF format.

The screenshot shows a web browser window titled 'Cognos Viewer - Microsoft Internet Explorer'. The address bar shows 'http://varese.boeblingen.de.ibm.com:9380/cognos8/cgi-bin/cognos.cgi'. The main content area displays a report titled 'RedParts Distribution' with the subtitle 'Top Volume Customers'. The report includes a filter section with three criteria: 'Order Status - Shipped & Packed', 'Unit Quantity: Greater than or equal to 18', and 'Total Charge Amt: Descending order'. Below the filter is a table with the following data:

Nation	Customer Name	Order Number	Order Date	Total Charge Amt	Unit Quantity
ALGERIA	Paul H J	4152503	10/01/2007	\$16,180.88	18
	Gaurav R V	4144605	02/10/2007	\$15,672.11	18
	Gaurav X Y	29916	17/12/2007	\$14,516.56	18
	Peter E B	4141527	02/02/2007	\$14,229.69	18
ALGERIA				\$60,599.25	72
ARGENTINA	Michael H M	29532	06/07/2007	\$14,244.01	18
	Paolo M M	29180	08/11/2007	\$13,408.27	18
ARGENTINA				\$27,652.27	36
BRAZIL	Frank F I	4149297	28/06/2007	\$14,932.58	18
BRAZIL				\$14,932.58	18
CANADA	Cristian W O	4148627	01/07/2007	\$16,044.21	18
	Sam H P	30500	23/12/2007	\$15,698.74	18
	Cristian W O	4141439	23/01/2007	\$15,246.58	18

Figure 13-85 Query - Viewing the report as a PDF

## 13.6 Multidimensional analysis with Analysis Studio

In this section, we show how to perform simple guided analysis based on data that was taken from our DDS and placed into an OLAP cube. We demonstrate the analysis by using Analysis Studio within Cognos 8 BI. Analysis Studio can be used to analyze data both within OLAP data sources, such as Cognos PowerCubes, and within relational Framework Manager packages that have dimensions defined and published. Figure 13-41 on page 346 shows an example of dimensions that are defined in Framework Manager.

We base our analysis on a business requirement listed in Table 5-2 on page 73. The examples provided are simple in nature and do not demonstrate the complete comprehensive functionality of Analysis Studio or OLAP data sources. For example, typically a number of exploration techniques, such as defining sets or adding calculations, can be applied when using Analysis Studio. These have not been included, and the steps that are provided only demonstrate some of the techniques that can be applied to perform the following analysis.

The business requirement is to perform some analysis on *part return rates*. In our example, we show, for a given financial quarter YTD, the highest return rates. We then explore this further to determine these return rates by manufacturer and customer nation. Based on the results of this analysis and other information that has been given to our call center operators, we have an idea that these return rates may be linked to the types of packaging that are used when distributing our parts to customers. We want to investigate this and determine whether this is impacting all regional promotions or only some promotions. Therefore, we continue to analyze return rates by customer promotion campaigns and part container types.

Multidimensional analysis provides a quick way to perform guided analysis and exploration by navigating through information by using conformed organization reporting structures and measures. By using Analysis Studio, you can select members within dimensions as rows and columns for describing the measures defined within the data. You can then slice and dice, drill up, drill down, and summarize the information to your requirements. As you discover details of interest, you can further investigate the results by changing the selections that you have made.

In the following steps, we assume that the package RedPart Order (Analysis), which uses the OLAP PowerCube, is already created and published to the Cognos BI Server as discussed in “Creating a data source for the PowerCube” on page 365.

1. In the Cognos 8 portal, select **Launch** → **Analysis Studio** in the top right corner of the window (Figure 13-86).

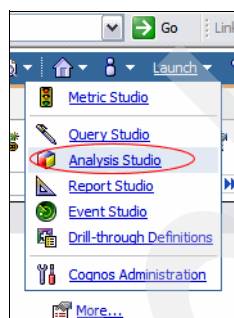


Figure 13-86 Launch analysis studio

Analysis Studio opens and depending on the setting, the Welcome window is displayed on which you can start using a blank analysis or a default analysis. The default analysis builds an initial analysis by selecting default dimensions for the row and column and a default measure.

2. For our scenario, click **Blank Analysis** and click **OK** as shown in Figure 13-87.

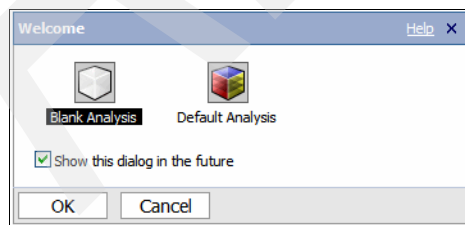


Figure 13-87 Analysis - Blank analysis

3. In our scenario, in the Insertable Objects pane, expand the **All Customers** dimension and drag the **All Customers** hierarchy to the rows position within the work area (Figure 13-88).

By doing this, we can explore the return rates of parts for customers that have returned their orders. The customers dimension is represented by the level displaying customer nations, with a row total label displaying the text *All Customers*.

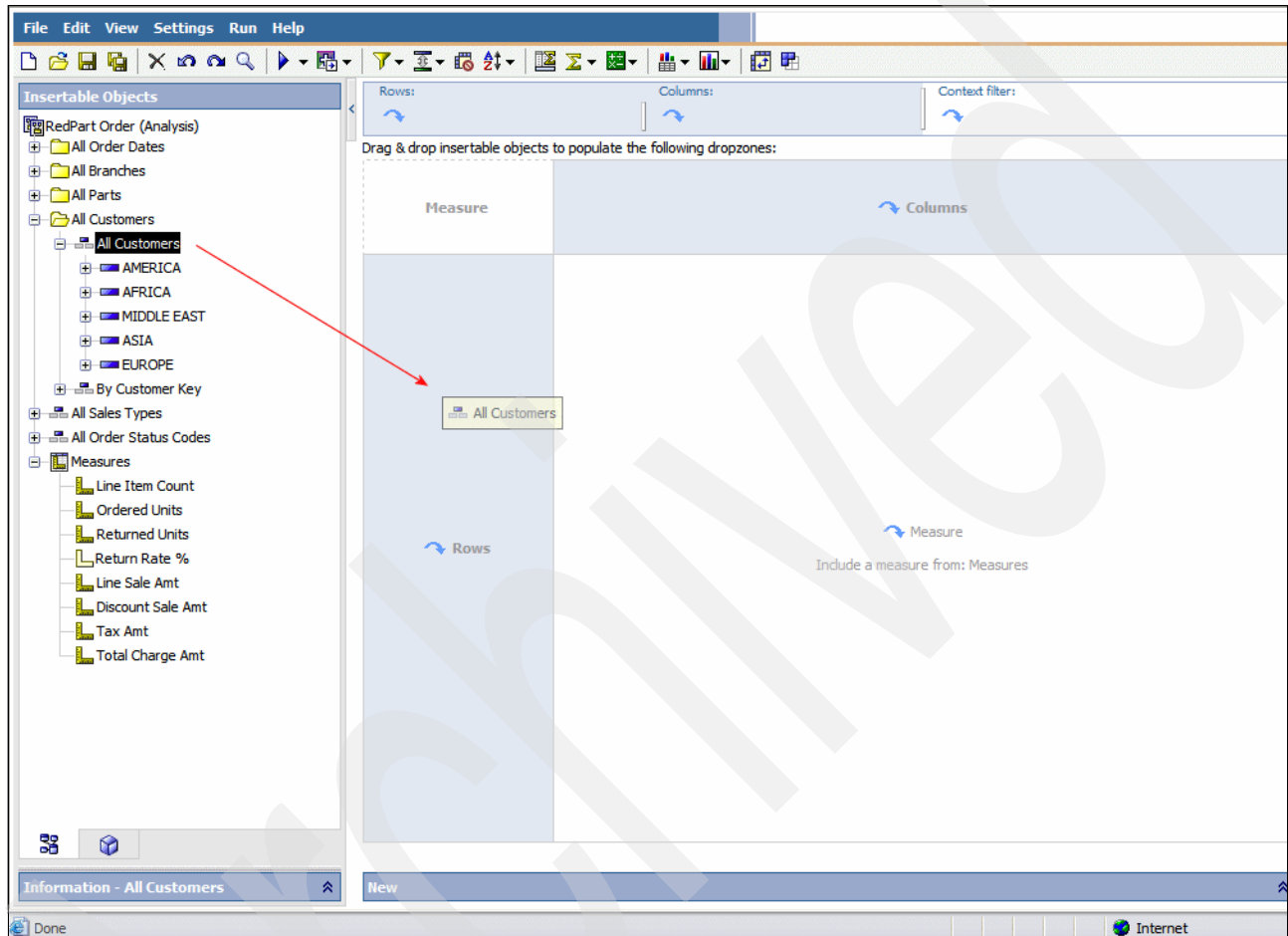


Figure 13-88 Analysis - Selecting rows

4. To investigate return rates for parts, include the part dimension in our analysis. In the Insertable Objects pane, expand the **All Parts** dimension and drag the **All Parts** hierarchy to the column position in the work area (Figure 13-89).

The part dimension is displayed as columns, represented by the part manufacturers.

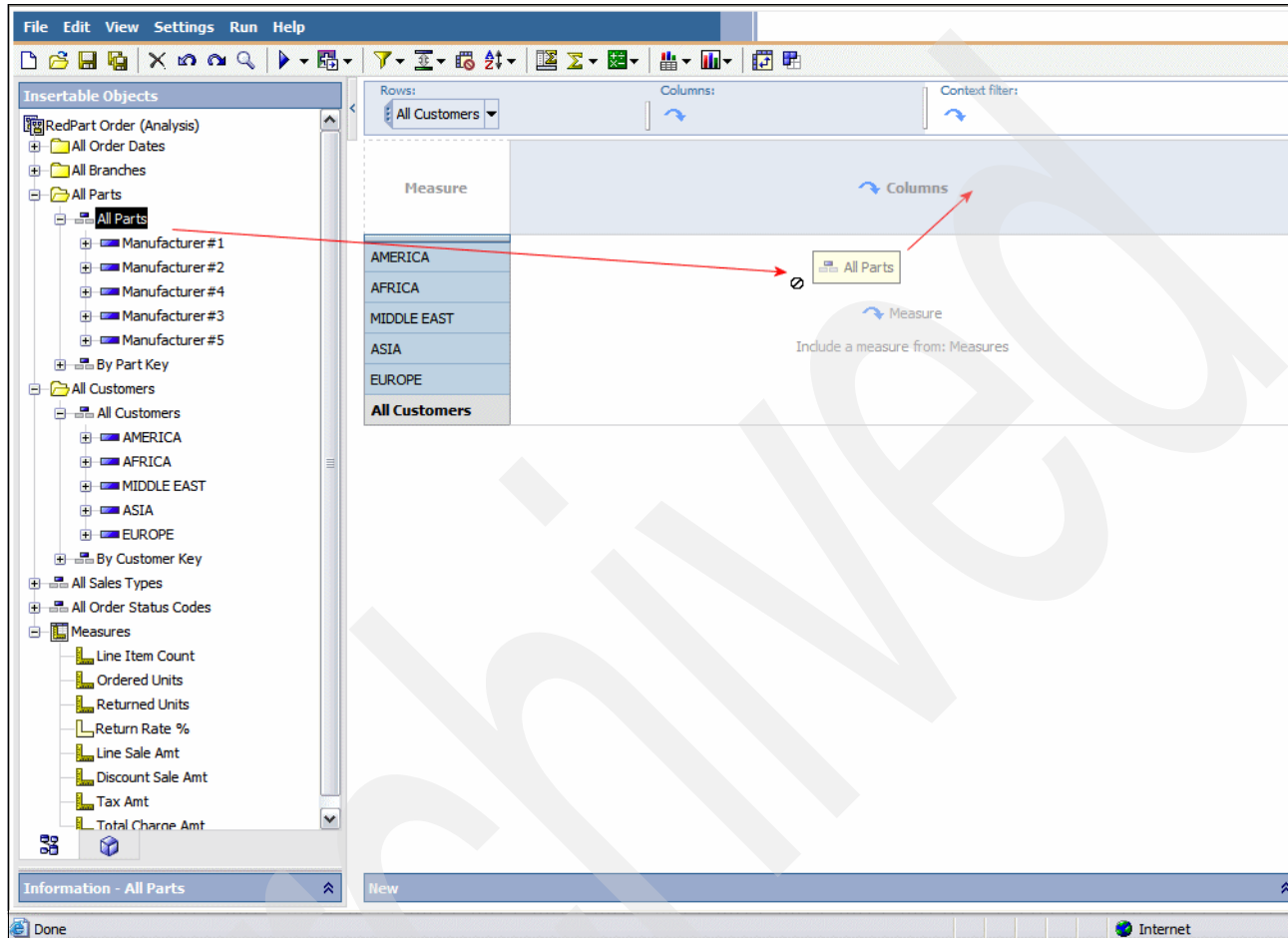


Figure 13-89 Analysis - Selecting columns

Several corporate measures or facts are available from the published PowerCube. At this point we can choose all of them or a subset of them to investigate. We can also define new calculated measures. In this case, we want to specifically investigate return rates.

5. In the Insertable Objects pane, expand the **Measures** folder and drag **Return Rate %** to the Measure position in the work area (Figure 13-90).

The result is a cross tab analysis of return rate % by part manufacturers and customer nations. The intersection of the total row and column for manufacturer 2 shows that, for all customers who ordered parts from this manufacturer, there is a 30.46% return rate of parts. This is based on all orders and not specific to a time frame.

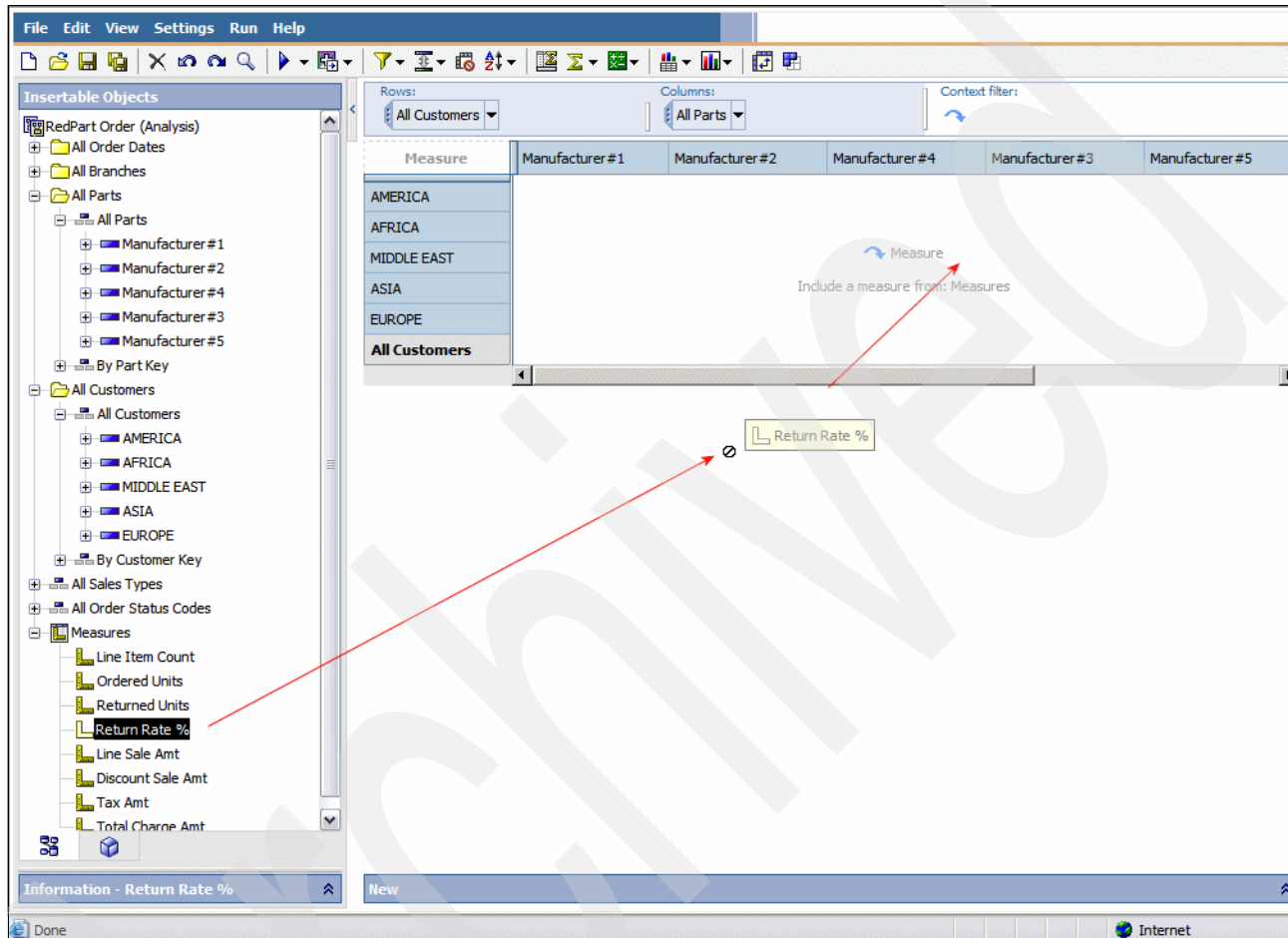


Figure 13-90 Analysis - Selecting a measure

- To investigate Manufacturer 2 further and determine if they are in fact the highest source of return rates for the current year, see what the return rate is for this year up to the end of the last financial quarter. In the Insertable Objects pane, expand the **All Order Dates** dimension and the **YTD** hierarchy in this dimension. Drag the **2007 Qtr 3** category to the Context Filter position in the work area (Figure 13-91).

The result is that the data is sliced based on the filter 2007 Qtr 3. We see that the 2007 Qtr3 YTD return rate for Manufacturer 2 is 28.78%, but also notice that this is not the highest return rate. Further analysis is required.

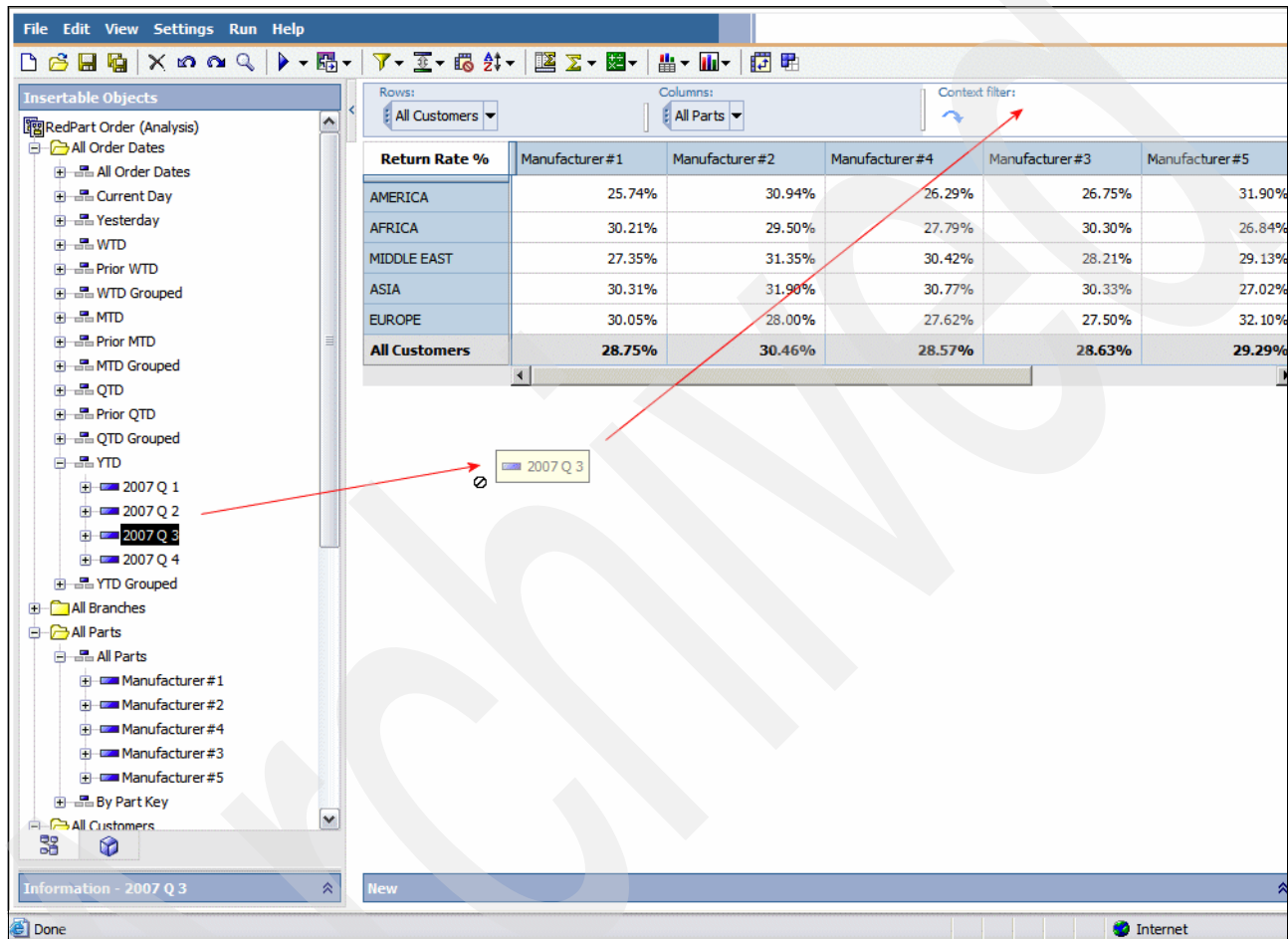


Figure 13-91 Analysis - Selecting a filter

- Click the **Swap Rows and Columns** toolbar button to pivot the row and column information in the cross tab analysis shown in the work area. This is the second to last button on the right above the overview area of the window. The rows and columns are then swapped.
- Sort the manufacturer rows in descending order by the manufacturer subtotal for Return Rate %. The column header reads All Customers. Right-click the **All Customers** subtotal and select **Sort** → **Descending**.

The part manufacturers are now ordered from highest return rate for all customers to the lowest return rate for all customers. By doing this, we can determine that, in fact, Manufacturer 5 is producing the highest return rate for the year so far.

- Insert a standard column chart. Click the **Chart Type** toolbar button and select **Column Chart** → **Standard** (Figure 13-92). The column chart is inserted above the cross-tab analysis.

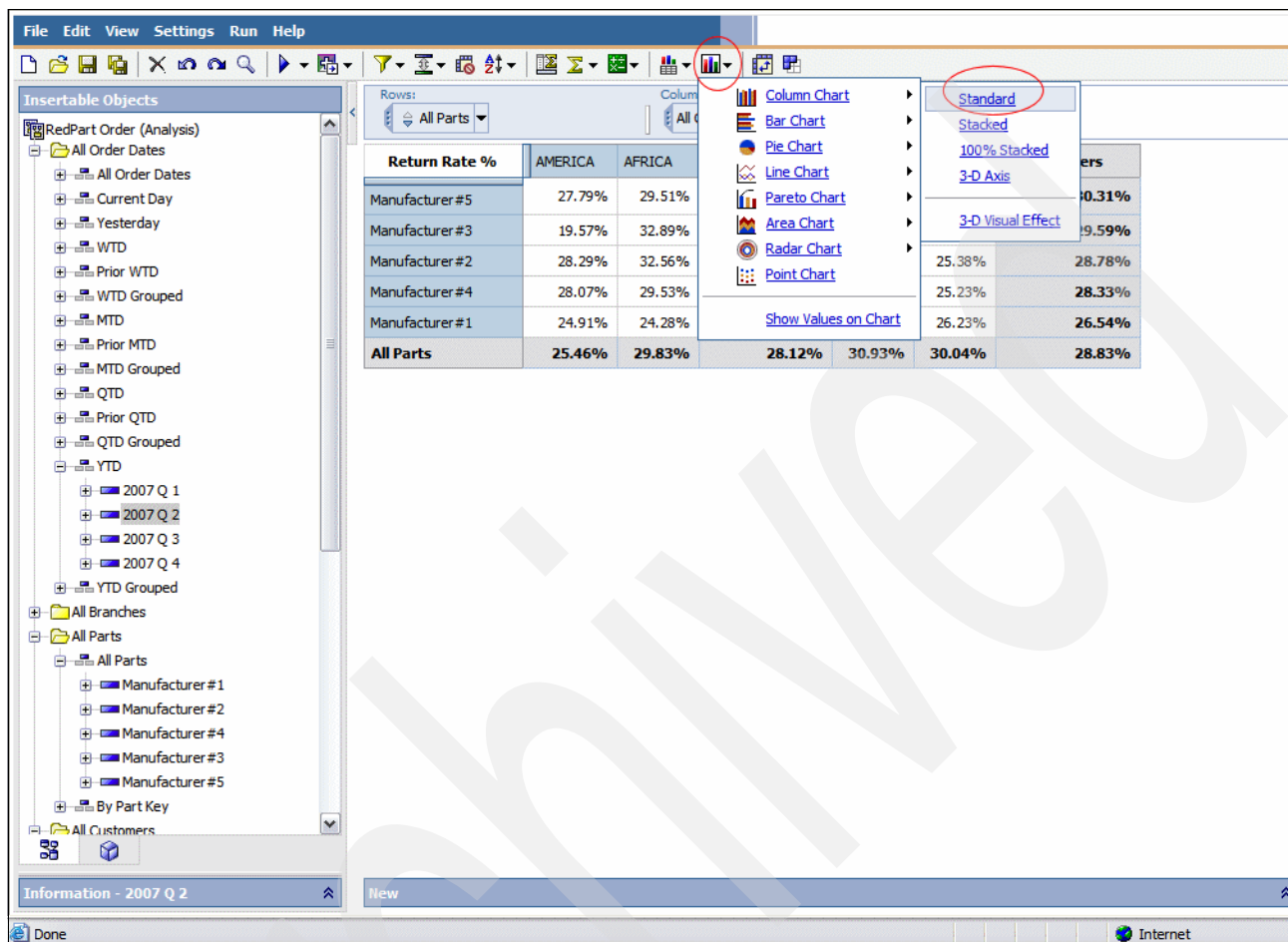


Figure 13-92 Analysis - Including charts



10. By using the cross tab and chart that are displayed, we can determine that, although Manufacturer 5 has the highest customer return rate, the largest impact of this seems to be from Europe. Further analysis is now possible, such as drilling down to a particular nation within Europe or part brand within manufacturer to further analyze and solve why the part return rate is so high. However, at this stage in our example, we want to name the report and save it for reference.

- a. Click the **Report** menu and **Report Options**.
- b. In the Report options pane (Figure 13-93), on the Title page, for Title, type RedParts Distribution, and for Subtitle, type Return Rate Analysis. Click the **OK** button.

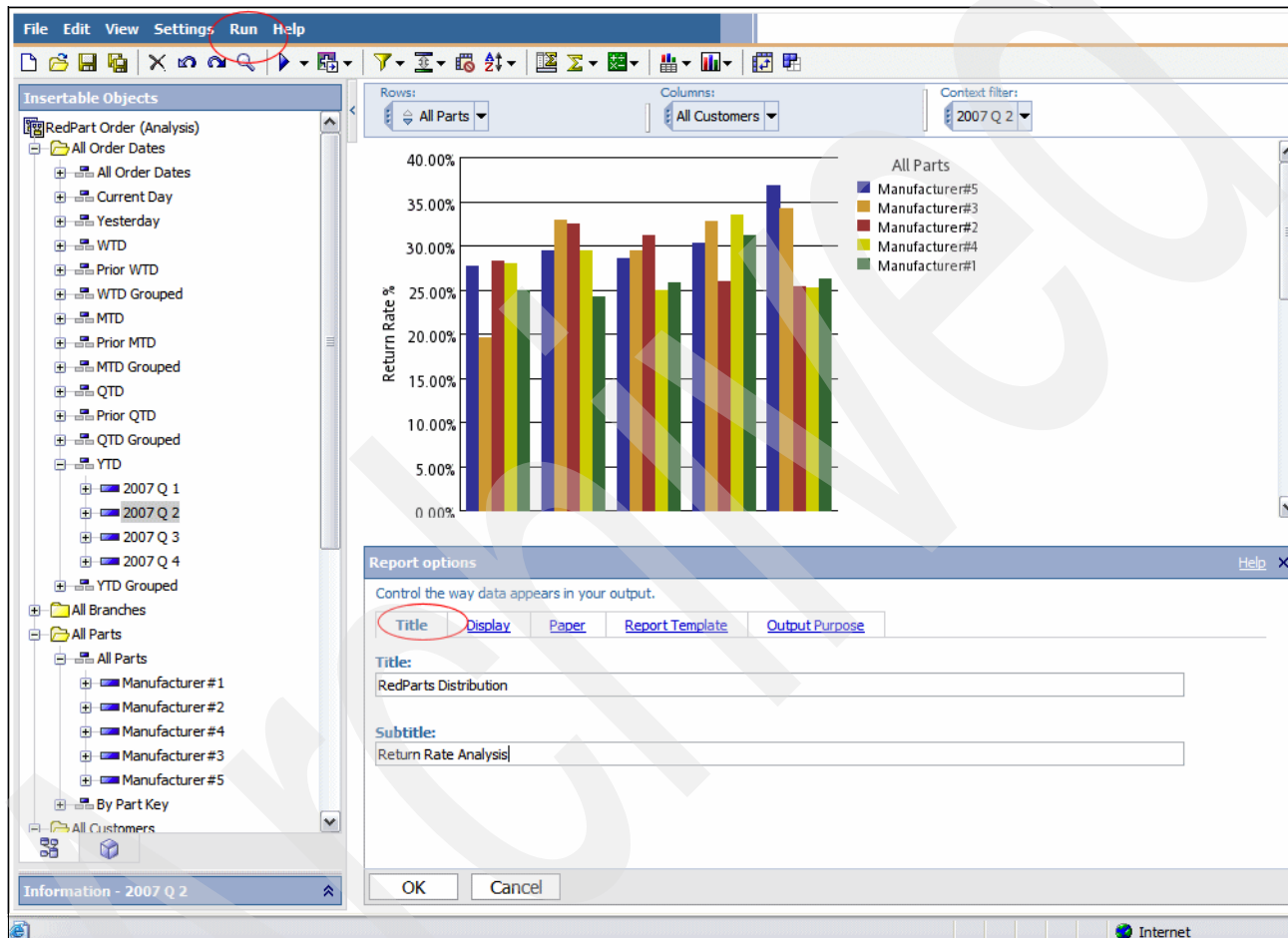


Figure 13-93 Analysis - Adding a title



11. To save the analysis:

- Click the **Save** or **Save As** toolbar buttons (highlighted in Figure 13-94).
- Navigate to the server **Analysis** folder in the folder Redbook Scenario - RedParts Distribution Warehouse.
- Click **Save**.

The analysis is available from this location within the Cognos Connection portal.

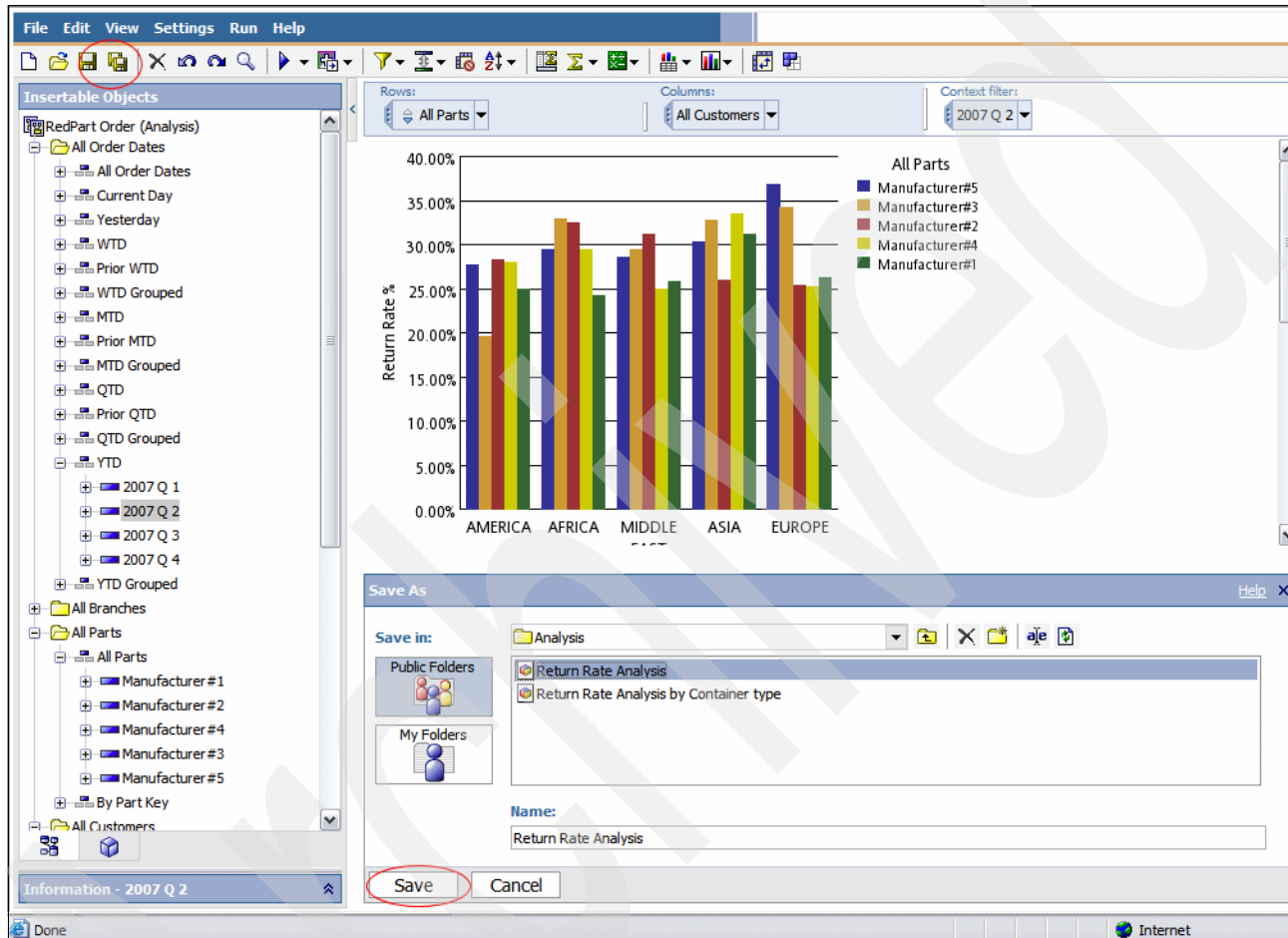


Figure 13-94 Analysis - Saving the layout

Figure 13-95 shows the completed Return Rate analysis by Part Manufacturers and Customer Nations. Our analysis has shown that, for the current year up to the end of the last financial quarter, manufacturer #5 is having the largest impact on customer return rates, based on returned European orders.

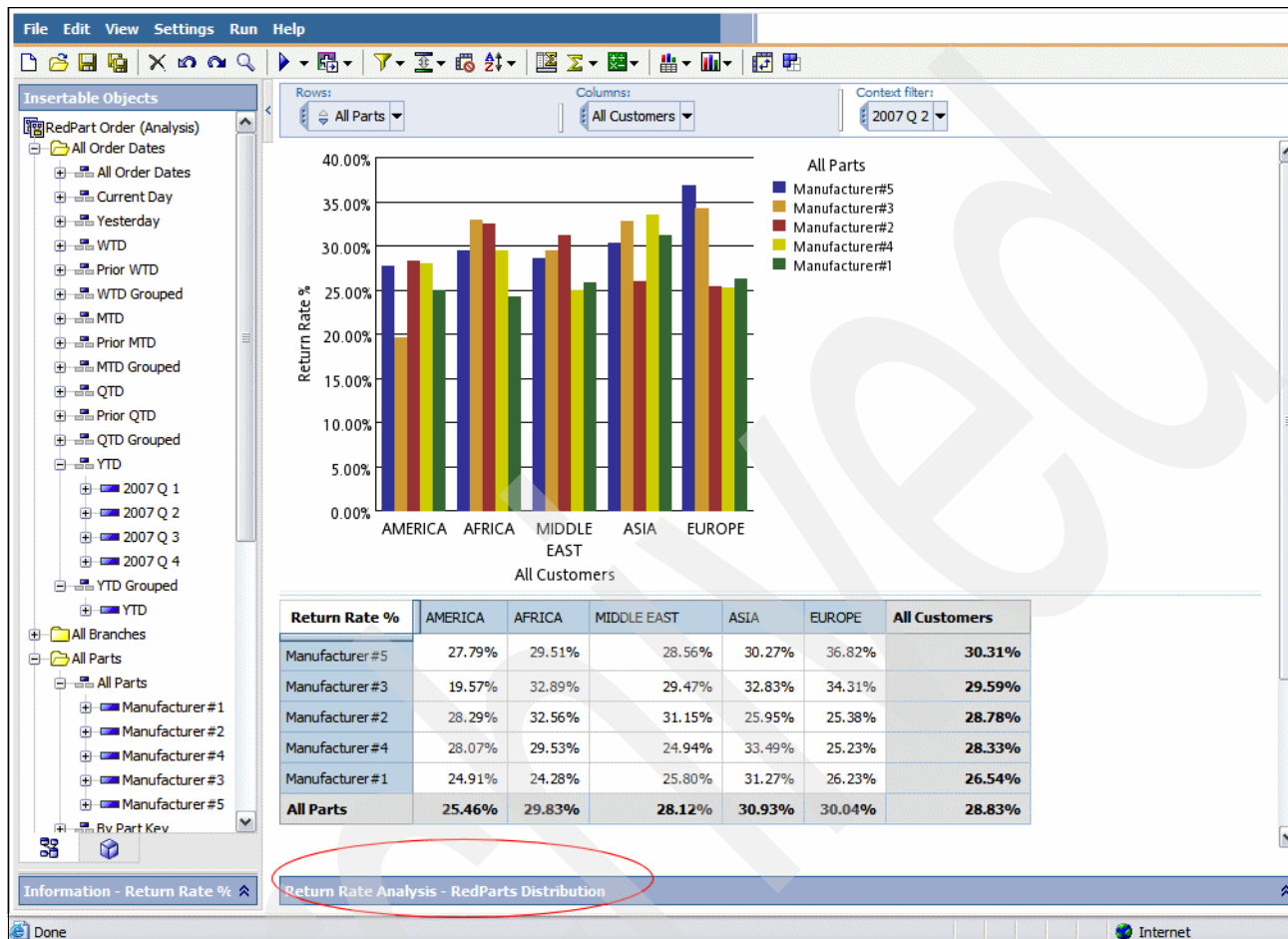


Figure 13-95 Analysis - Completed RedParts sample 1

In our simple scenario, based on earlier results and information provided from the RedParts Distribution call center, further analysis is required. In the second example, the analysis can continue to include the packaging for parts sent to customers and the regional promotions that were running for the customer's nation. Figure 13-96 on page 395 shows example output that might be reached during this further analysis.

To create this exact output, repeat the techniques in sample analysis 1, to apply the following information as highlighted in Figure 13-96:

1. Drag the alternate part hierarchy for container types (**By Part Key**) to the rows.
2. Drag the alternate customer hierarchy for economic promotions (**By Customer Key**) to the columns.
3. Ensure that the Return Rate % measure is still shown within the measure work area.
4. Replace the chart as shown in Figure 13-96 with a Standard Pie Chart.
5. Click the first instance of the Container rows within the cross tab and click the **Suppress Rows or Columns** toolbar button (left of the Sort toolbar button).

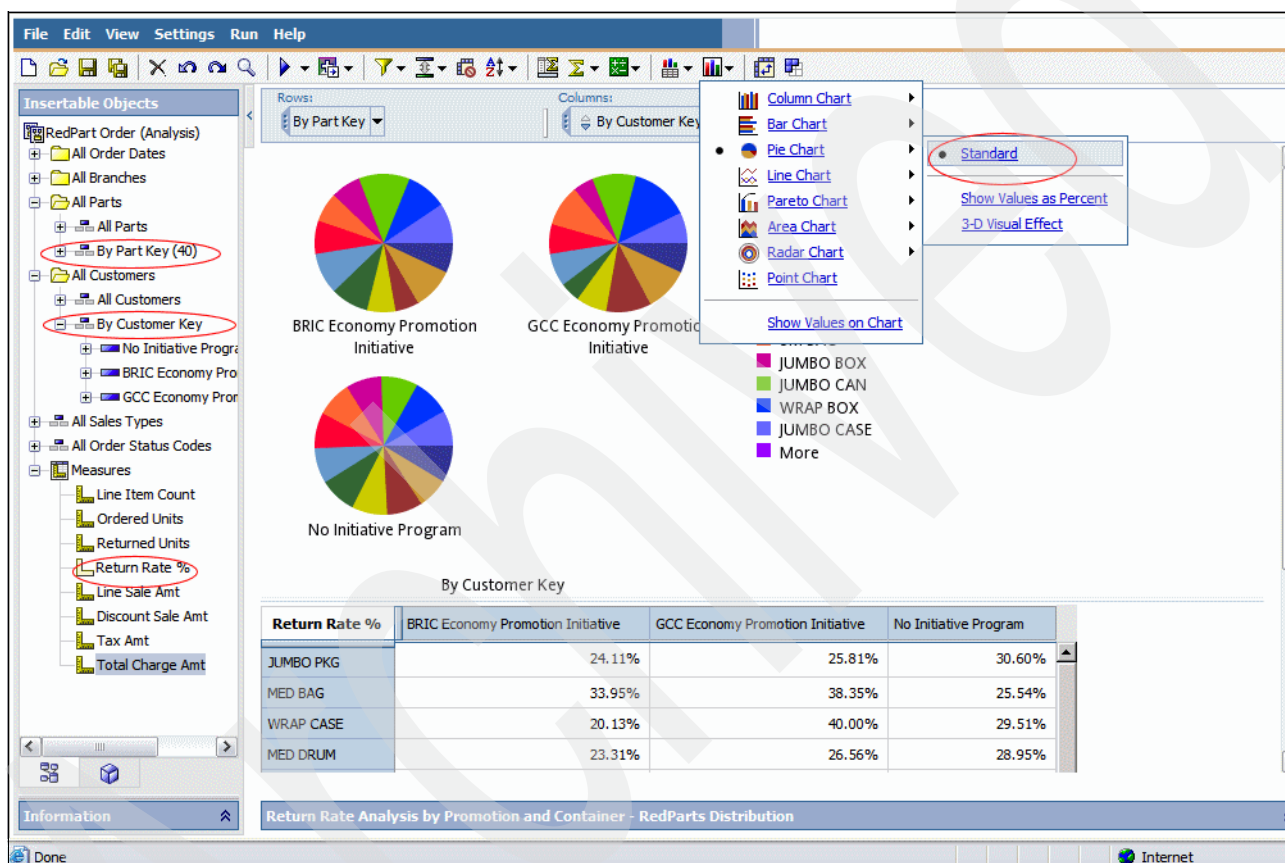


Figure 13-96 Analysis - Building RedParts sample 2

6. Remove the subtotals column from the cross tab (Figure 13-97):
  - a. Select **Settings** → **Totals and Subtotals**.
  - b. Deselect the second last check box.
  - c. Click **OK**.

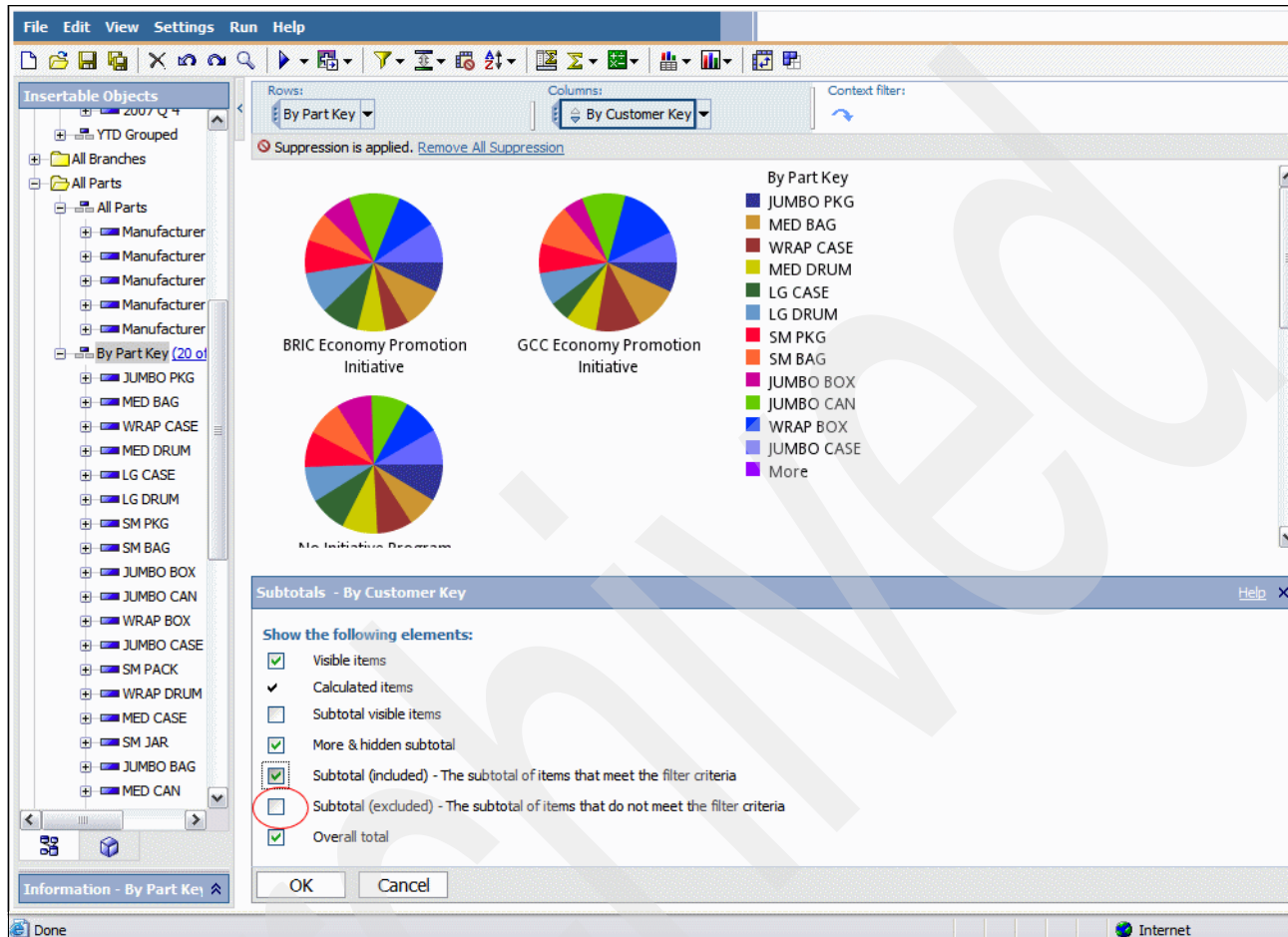


Figure 13-97 Analysis - Removing the subtotals

7. Modify the title to Return Rate Analysis by Promotion and Container, and save to the Cognos Connection portal as a separate analysis result.

The analysis examples are now complete. By using Analysis Studio, we were able to investigate the DB2 z/OS data warehouse information by using defined reporting structures and measures. We were able to analyze different views of data until we saw something of interest. Analysis by discovery is useful in providing information when required to those who require it during a decision making process.

## Reporting with DataQuant, QMF, and AlphaBlox

Many analytical tools are available that can be used for business intelligence (BI) reporting on a data warehouse that is implemented with DB2 for z/OS. Examples include Cognos, Business Objects, Crystal Reports, DataQuant, QMF, and AlphaBlox, among others.

In the scenario described in this book, we use Cognos to implement the required queries. See Chapter 13, “Reporting and analysis with Cognos 8 BI” on page 305. However, IBM has additional products that can be considered when the reporting product is selected. We briefly explore some of the additional options.

This chapter contains the following sections:

- ▶ 14.1, “DataQuant” on page 398
- ▶ 14.2, “QMF” on page 409
- ▶ 14.3, “AlphaBlox” on page 411

## 14.1 DataQuant

DataQuant is a query, reporting, and data visualization solution for both relational and multidimensional data sources that became available in 2007. This product provides enterprises with a means of rapidly developing and distributing page-based reports and interactive visual dashboards.

DataQuant consists of an Eclipse-based content authoring environment and a Web-based runtime environment. The product presents features that are a natural evolution of QMF and the Visionary tools.

### 14.1.1 When to consider DataQuant

Consider using DataQuant in the following situations:

- ▶ Where there is a need for a turnkey solution that allows organizations to rapidly create and distribute BI queries, reports, and interactive dashboards across the enterprise
- ▶ Where there is a need for self-service reporting, empowering lines of business and users with the ability to answer their own business questions while decreasing demands on IT
- ▶ For customers who consider resource governing, granular security, and object tracking to be important factors in an enterprise-wide BI solution
- ▶ Where quick prototyping and rapid solution development are more important than complex analytical features
- ▶ For customers who find competitor solutions too complex and costly
- ▶ For customers who already have reporting capabilities implemented with QMF

### 14.1.2 DataQuant functions

While many analytical tools require extensive programming and lengthy deployment times, DataQuant for z/OS and Multiplatforms provides an easy-to-use environment that you can immediately use to quickly and easily develop BI solutions. This includes a wealth of charts, controls, and graphics that you can drag to quickly create dashboards and reports. In addition, DataQuant includes the following features:

- ▶ Over 100 built-in mathematical and analytical functions
- ▶ The ability to create compound, multipage reports that concurrently draw data from multiple data sources across your enterprise
- ▶ Multiplatform support including System z, Linux on System z, Linux, System i™, Windows, IBM AIX, and Solaris
- ▶ Support for interactive dashboards that permit users to dynamically access relevant enterprise data, on demand, using intuitive drill-down and information zooming facilities
- ▶ Drag-and-drop development of online analytical processing (OLAP) analytics, SQL queries, embedded subqueries, tabular reports, visual reports, and pivot tables
- ▶ OLAP query editor with support for MDX OLAP-based engines, connecting via XML for Analysis (XMLA)
- ▶ Full compatibility with an existing QMF infrastructure and objects
- ▶ Support for DB2, Informix®, and most other popular database management systems
- ▶ Support for a wide variety of report formats, including XML, HTML, Microsoft Excel®, and PDF



- ▶ Full support for the Eclipse Foundation's Business Intelligence and Reporting Tools (BIRT) report format, which complements DataQuant's native tabular and visual reporting formats
- ▶ A rich set of Java APIs, Web service APIs, and a command library interface, allowing DataQuant content to be directly embedded within a custom-developed or third-party application infrastructure
- ▶ One-click function to populate Excel with DataQuant output with all formatting retained

Thin client and rich client deployment options in DataQuant include both an Eclipse-based rich client desktop application and an WebSphere-based, thin-client Web application. The Eclipse-based offering provides a powerful, intuitive and highly productive rich desktop environment within which queries, reports, and visual dashboards (Figure 14-1) can be quickly authored, tested, and deployed.

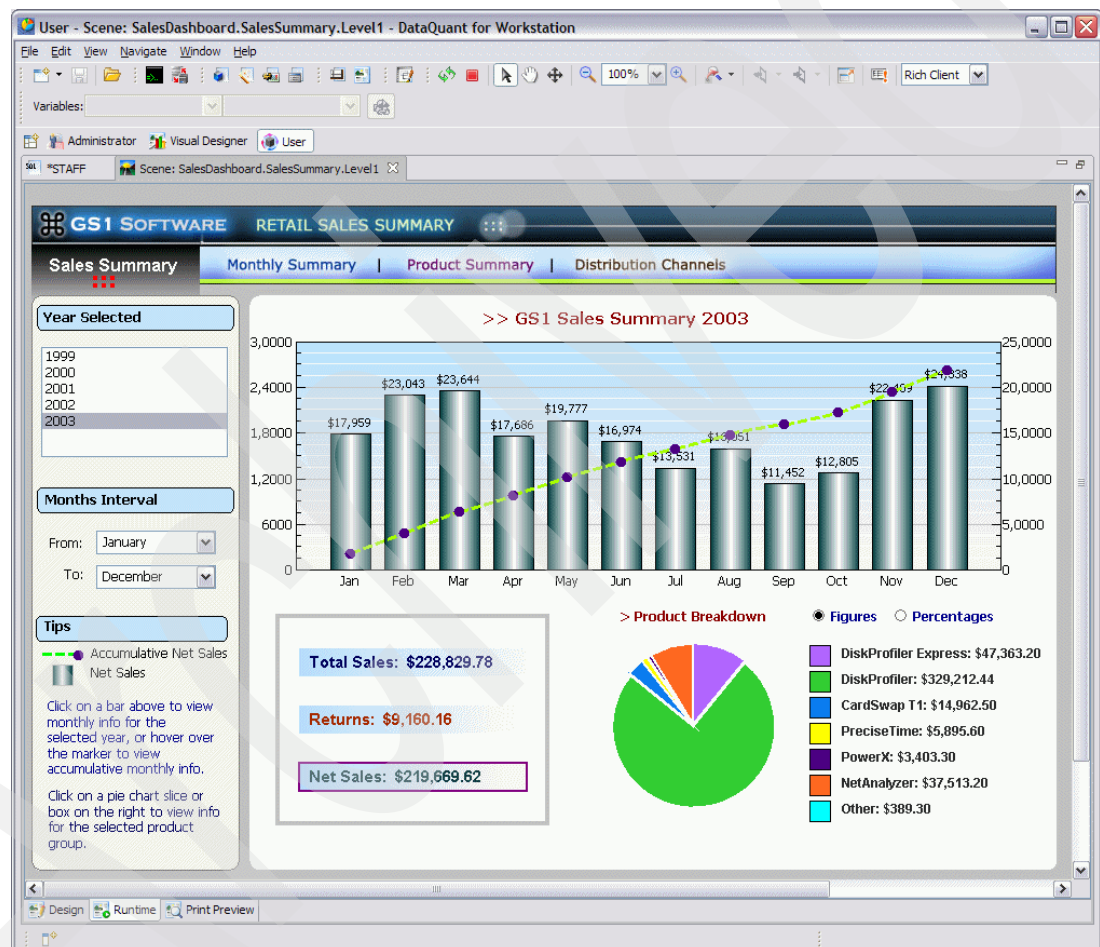


Figure 14-1 DataQuant dashboard

The high-performance runtime environment (thin client) based on WebSphere software extends key functionality to browser-based users across multiple platforms. It provides access to all DataQuant BI content, the ability to create queries and reports, and the ability to perform ad hoc, visual drag-and-drop data analysis. The share BI content internally and externally options in DataQuant for WebSphere include a robust service-oriented architecture (SOA). It provides a flexible infrastructure that enables you to easily share BI solution components (queries, reports, dashboards, and so on) with partners or clients over secure Web connections.

By using the SOA capabilities and rich security infrastructure of DataQuant, organizations can distribute BI assets to both internal or external users via a standard, secure Internet connection. Available to both Web and rich desktop application users, these SOA capabilities provide user and group-specific access to BI assets without requiring knowledge or direct access to the underlying databases and data repositories that power them. This way, you can readily share your BI solutions with users both inside and outside of the firewall, all with zero client-side administration. These capabilities provide a build once, service many infrastructure as required by SOA deployments.

Security and personalization integrated BI systems require a high level of security to protect important information. Regulatory requirements for information continue to grow. Therefore, you must be able to secure business information at the appropriate level.

DataQuant provides granular access control, tailoring the view and usage of available reports and the visualizations and data on a per user or group basis. For example, technical users may see a traditional database-centric view, where business users see a role-specific view of relevant reports and dashboards. You may opt to tailor offline schemas to each user or group so that individuals see only those tables and columns that are relevant to their job function or business area.

DataQuant also supports single sign-on, allowing users to log on to all enterprise assets using a single account. DataQuant logon information can be automatically passed to all databases or derived from specific accounts designated as appropriate for the particular user.

### 14.1.3 A small DataQuant report for RedParts Distribution

In the following scenario, we use the DataQuant visual report, because no interaction or configuration of the output is required. If interaction is required, dashboards are the right choice. We use the following steps to create the visual report:

1. Define the data source.

**Data source definition:** The data source is not defined yet, so we include this as the first step. If the data source is already defined, only steps 2 through 4 are necessary.

2. Create the query.
3. Create the visual report as base for further modifications.
4. Modify the report (to insert bar chart).

The following scenario assumes that DataQuant for Workstation is successfully installed on any of the supported platforms, and a DataQuant repository is created. This repository is used by DataQuant to save its meta information.

#### Defining the data source

To define the data source:

1. While in the administrator perspective, click the **Repositories** tab.
2. Expand the repository that you want to use to define your new data source.
3. Right-click the data source icon and select **New** → **Data Source**.



4. In the New Data Source window (Figure 14-2):
  - a. Enter the configuration data that required for the new data source. In this example, the data for the DDS database on z/OS described in 5.5, “The operational and dimensional data model” on page 85.
  - b. Verify the data and click the **Test Connection** button to test the connection to the data source.
  - c. If the test executed successfully, click **Next** to complete the definition of this new data source.

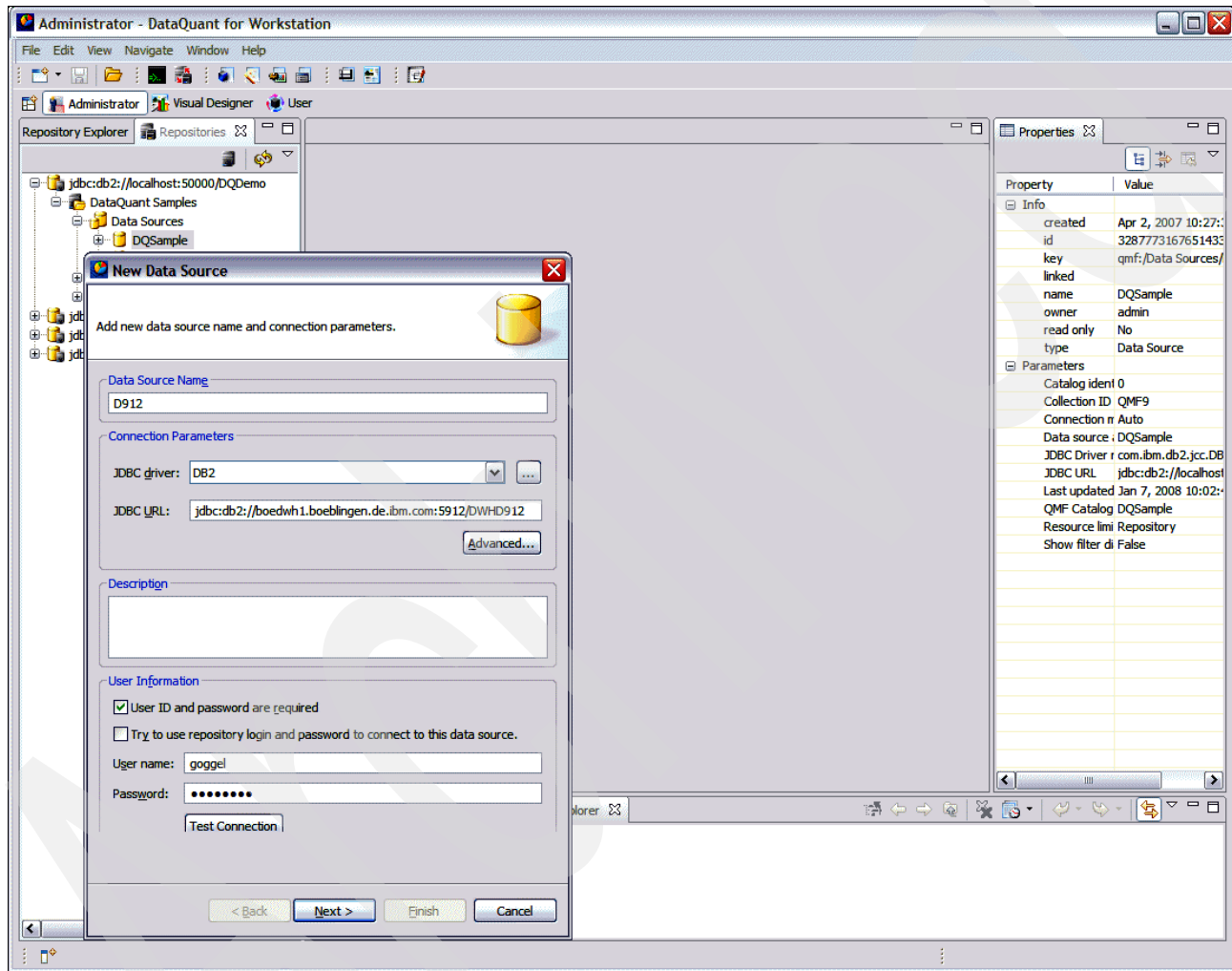


Figure 14-2 DataQuant - Defining the source data

After the new data source is created by the administrator, it is available for the administrator to use. If any user is able to use the new data source, a link must be created from the data source to the view to which the user has access. To create this link:

1. Click the **Repository** tab (while still in the administrator perspective) and drag the data source to the appropriate view. In this example, the view is “Lothar”.
2. To verify that the data source is visible in the view, switch to the user perspective and open the “Lothar” view.

You can now browse all tables in D912. Expand the data source D912 down to the table level. Figure 14-3 shows the tables in our DDS database. Right-click the **ORDER\_TRANSACTION\_FACT** table and select **Open with** → **table viewer** to display the contents of the selected table.

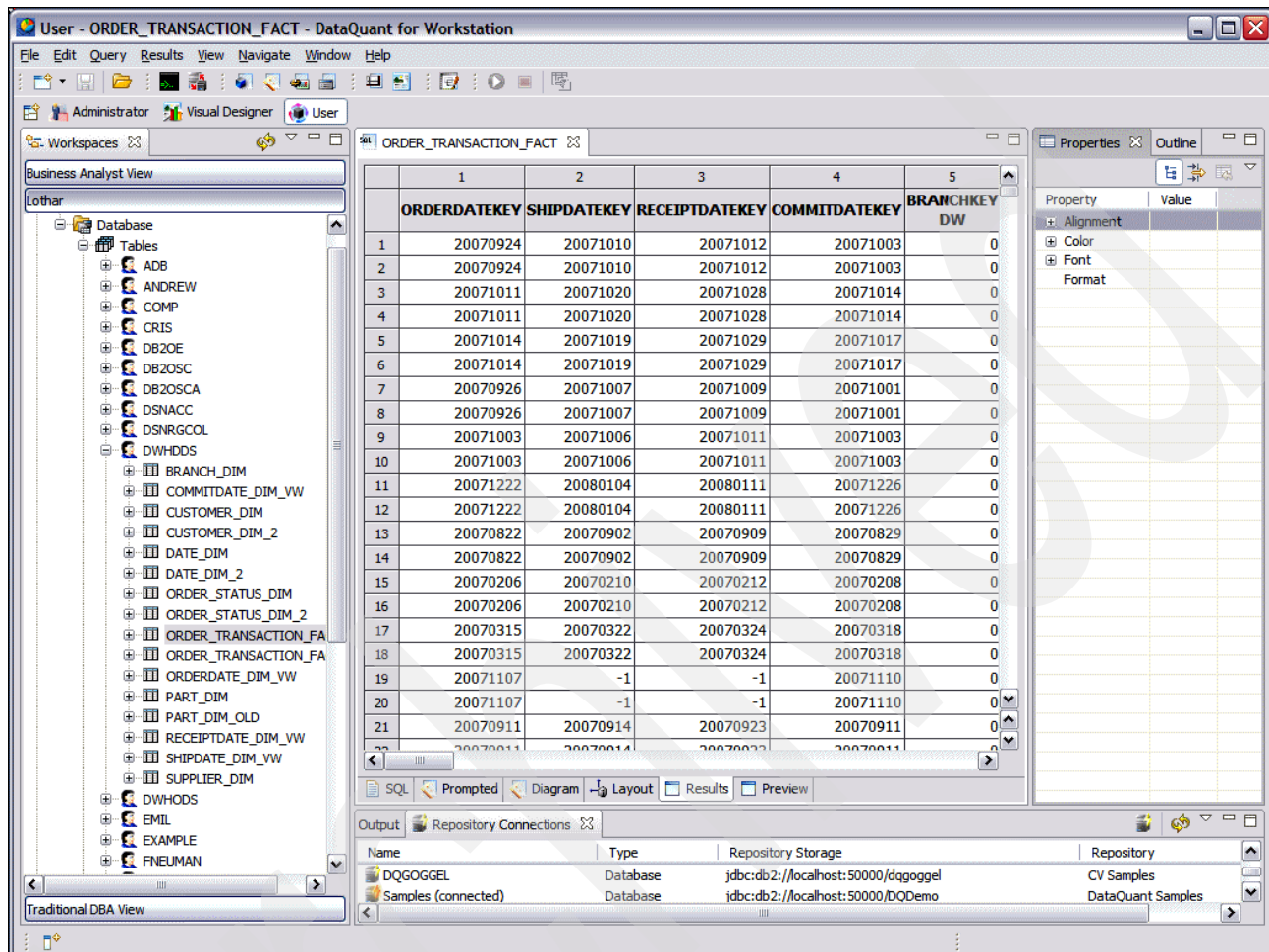


Figure 14-3 DataQuant - Displaying the table

## Creating the query

By using DataQuant for Workstation and DataQuant for WebSphere, you can create SQL statements that query a relational data source in any of the following ways:

- ▶ Write your own SQL statements
- ▶ Open queries that have been created and saved to a file or a database
- ▶ Create SQL statements by using the prompted interface
- ▶ Create SQL statements visually by using diagram interface (DataQuant for Workstation only)
- ▶ Create queries by using the Draw Query command (DataQuant for Workstation only)

Example 14-1 shows the sample query that we created.

*Example 14-1 Sample query for DataQuant*

```
SELECT A.ORDERKEY, B.CUST_NAME, C.PART_NAME
FROM DWHDDS.ORDER_TRANSACTION_FACT A INNER JOIN DWHDDS.CUSTOMER_DIM B ON A.
    CUSTKEY_DW = B.CUSTKEY_DW INNER JOIN DWHDDS.PART_DIM C ON A.PARTKEY_DW = C
    .PARTKEY_DW
WHERE (A.QUANTITY > 2)
```

Figure 14-4 shows the same query in the Diagram Query Builder.

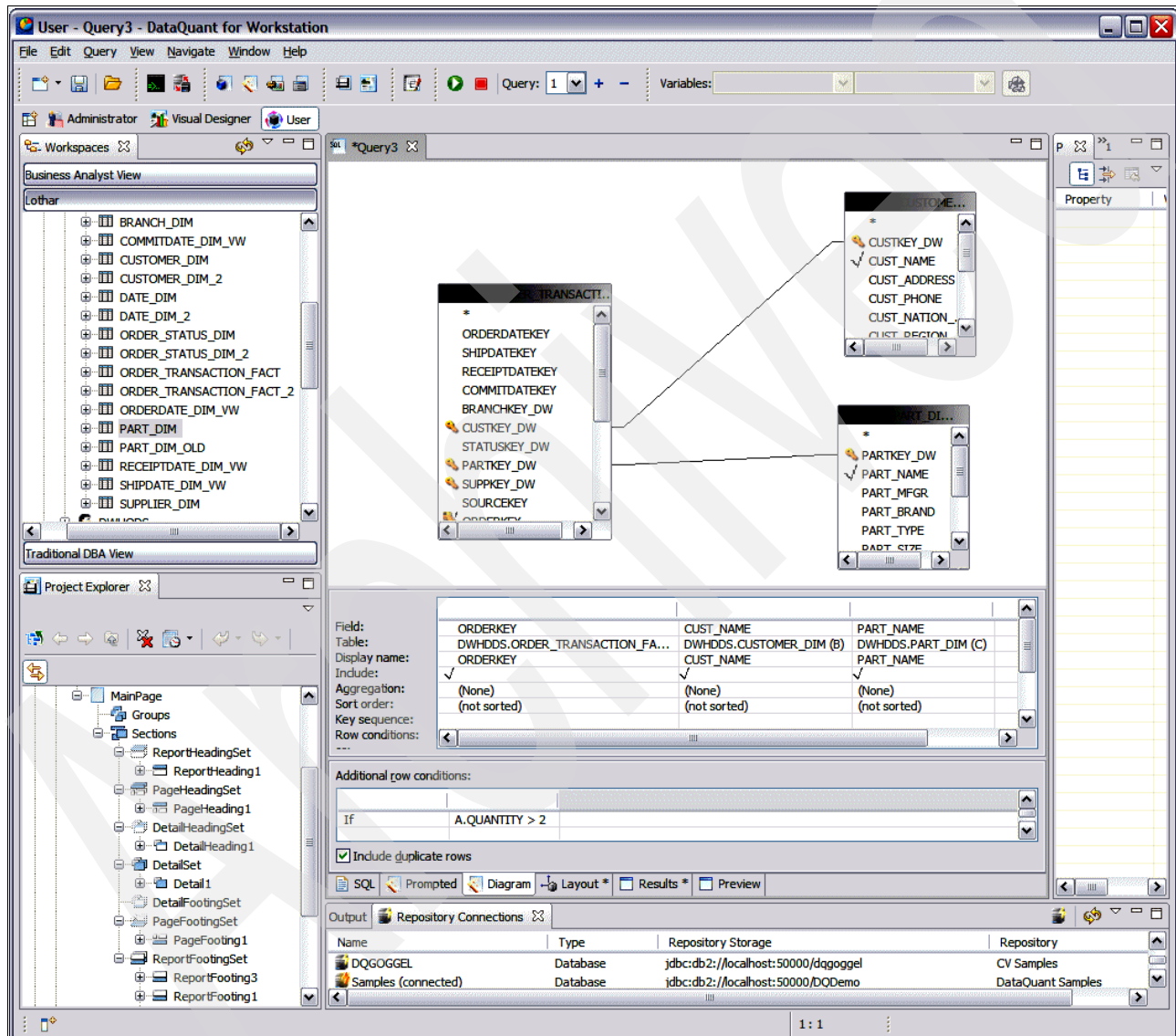


Figure 14-4 DataQuant Diagram Query Builder

Figure 14-5 shows the prompted Query Builder.

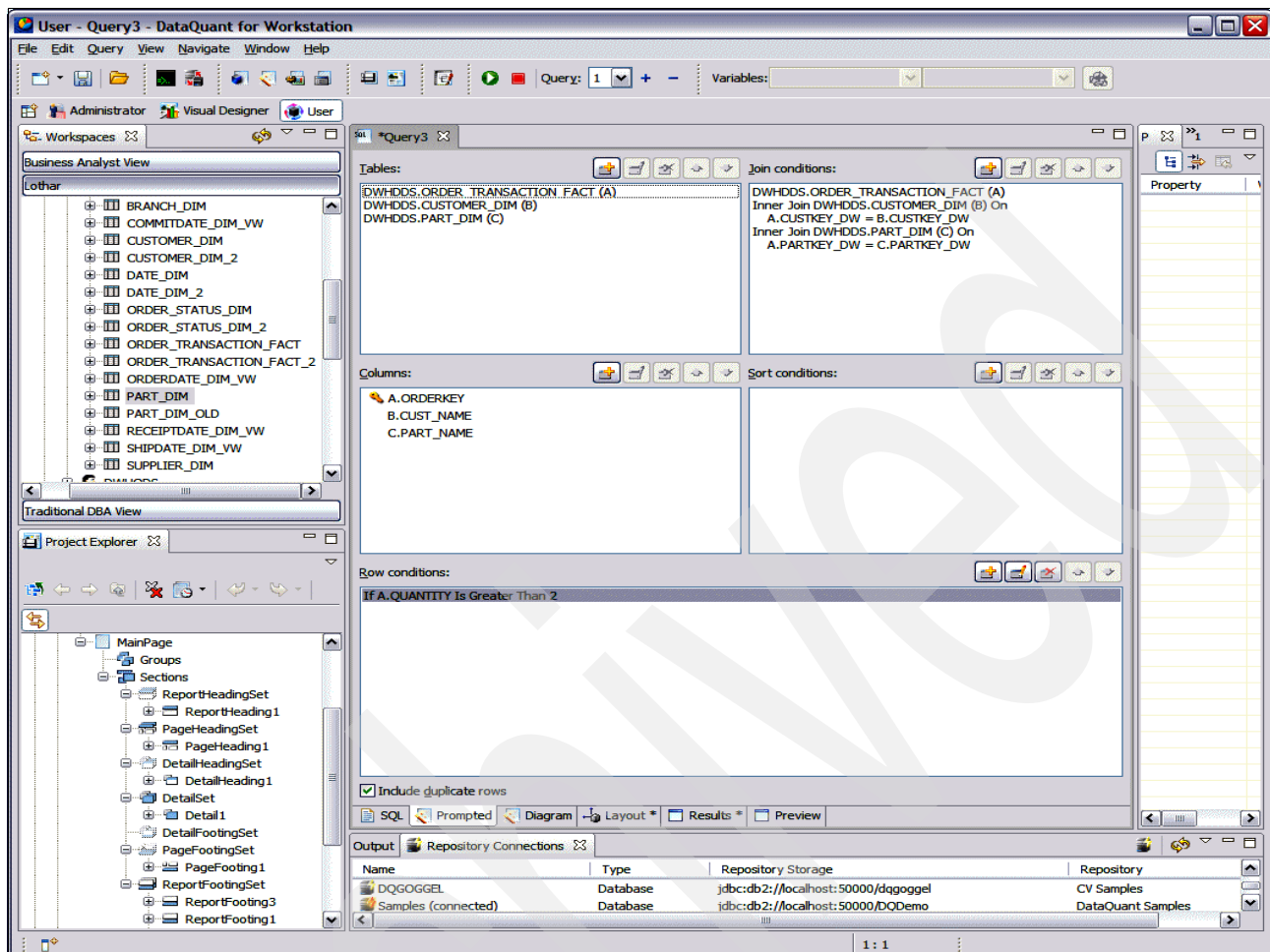


Figure 14-5 DataQuant Prompted Query Builder

While the Diagram Query Builder and the prompted Query Builder are well suited to create syntactical correct queries quickly, both builders do not support every possible SQL query.

In our example, the query is available in error free SQL, because we use the same query with Cognos, AlphaBlox, and DataQuant. Therefore, we paste the query into the SQL Query Builder panel. Figure 14-6 shows the query SQL.

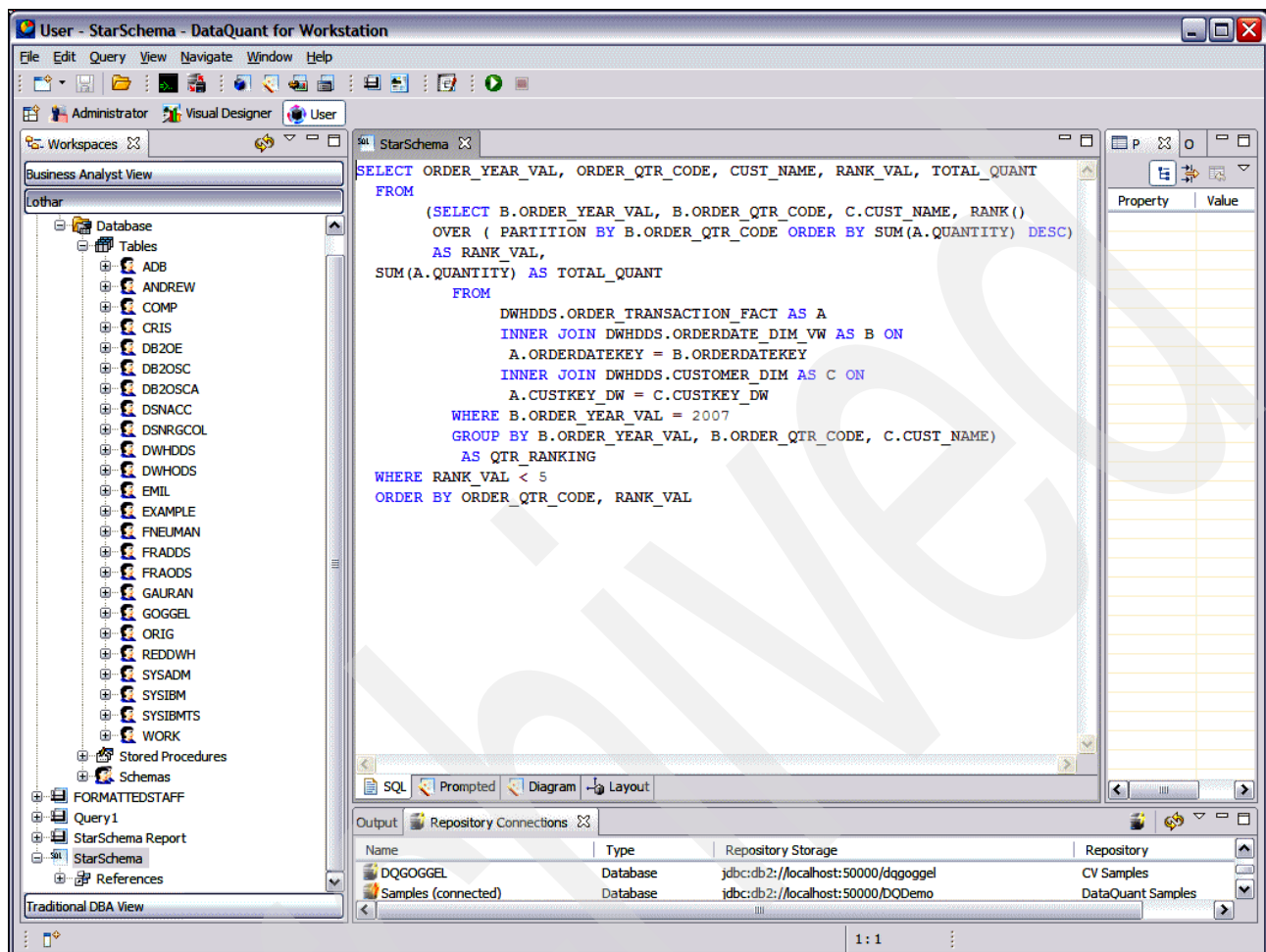


Figure 14-6 DataQuant SQL query editor



After the query is created by using either method, it is ready for execution. The execution of the query results in the output shown in Figure 14-7. It shows the ranking of the customers in each quarter of the year.

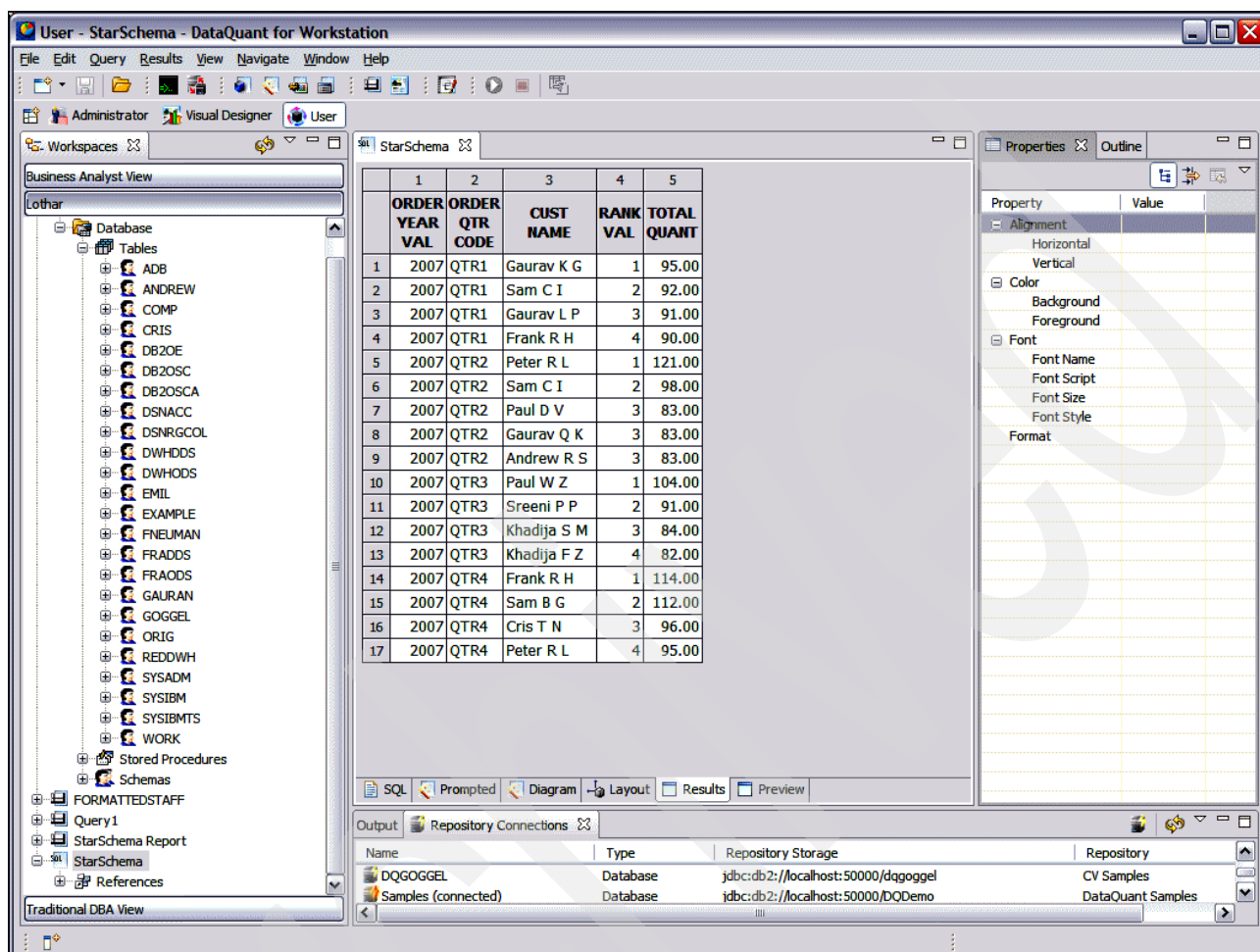


Figure 14-7 DataQuant tabular result

## Creating a visual report

Visual reports are page-based, printable reports that have the following key features:

- ▶ Support for the full range of dashboard graphics, including charts, geospatial layouts, and graphical primitives
- ▶ The ability to incorporate nested tables and charts in each report section, driven by independent queries that optionally draw one or more parameters from the outer query

For example, an employee report may include a salary history bar chart in each employee section of the report.

- ▶ The ability to incorporate fixed pages that are printed along with the report

These pages may be used for report title pages, summary data, and report appendices. Due to their graphical nature, visual reports are authored in the Visual Designer perspective in DataQuant for Workstation. DataQuant for WebSphere has the ability to run and display visual reports but does not have the ability to edit them.

To create a visual report in DataQuant for Workstation, perform the following steps. The query that we want to use is already open.

1. Click the **Display a report** button in the application toolbar or select **Results** → **Display Report item** from the menu bar.
2. In the Display Report window, select the first radio button and click **OK** to create a visual report using the format of the current query. A new visual report view is presented in Design mode as shown in Figure 14-8.
3. Select the Visual Designer perspective by either clicking the **Open Perspective** toolbar button or by selecting **Window** → **Open Perspective** → **Other item** from the menu bar. The Visual Designer perspective is displayed.

The Project Explorer lists the key components of the visual reports. The report content is fully represented by the MainPage item. The MainPage object is driven by the visual report query. The sections under MainPage represent the headings, detail (data) areas, and footers (totals) across each of the grouped sections (department and job in this case).

Given this basic structure, we are now free to add additional graphical elements in the headers, footers, or detail sections. As a simple example, we add a bar chart to the Report Footing Set, which charts the quantity of parts for each customer.

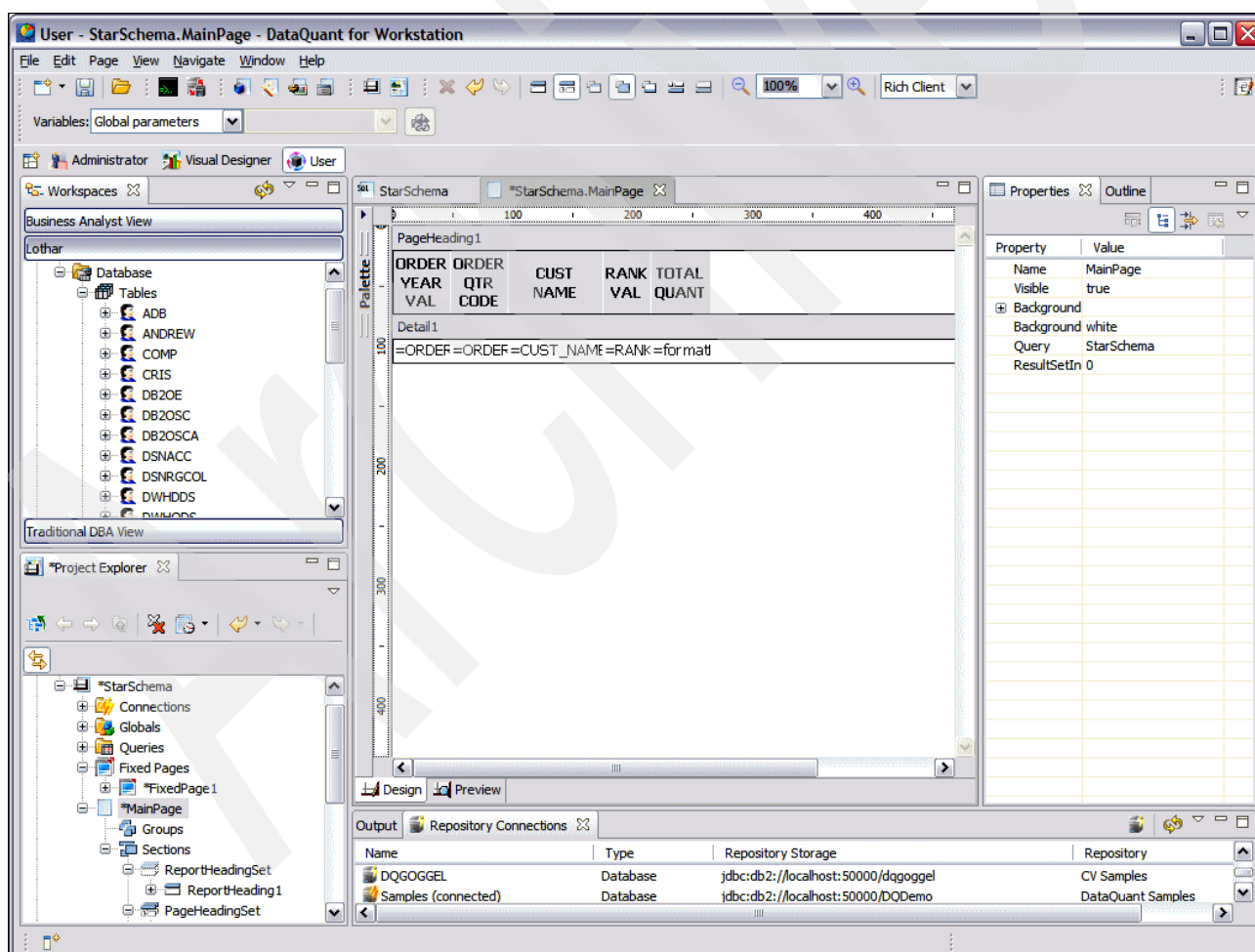


Figure 14-8 DataQuant - Visual report design view

To insert the bar chart below the result table, select the **Report Footing Set** section in the Project Explorer. Enlarge the graphical area of the Footing Set, and drag the bar chart icon from the palette to the enlarged area as shown in Figure 14-9.

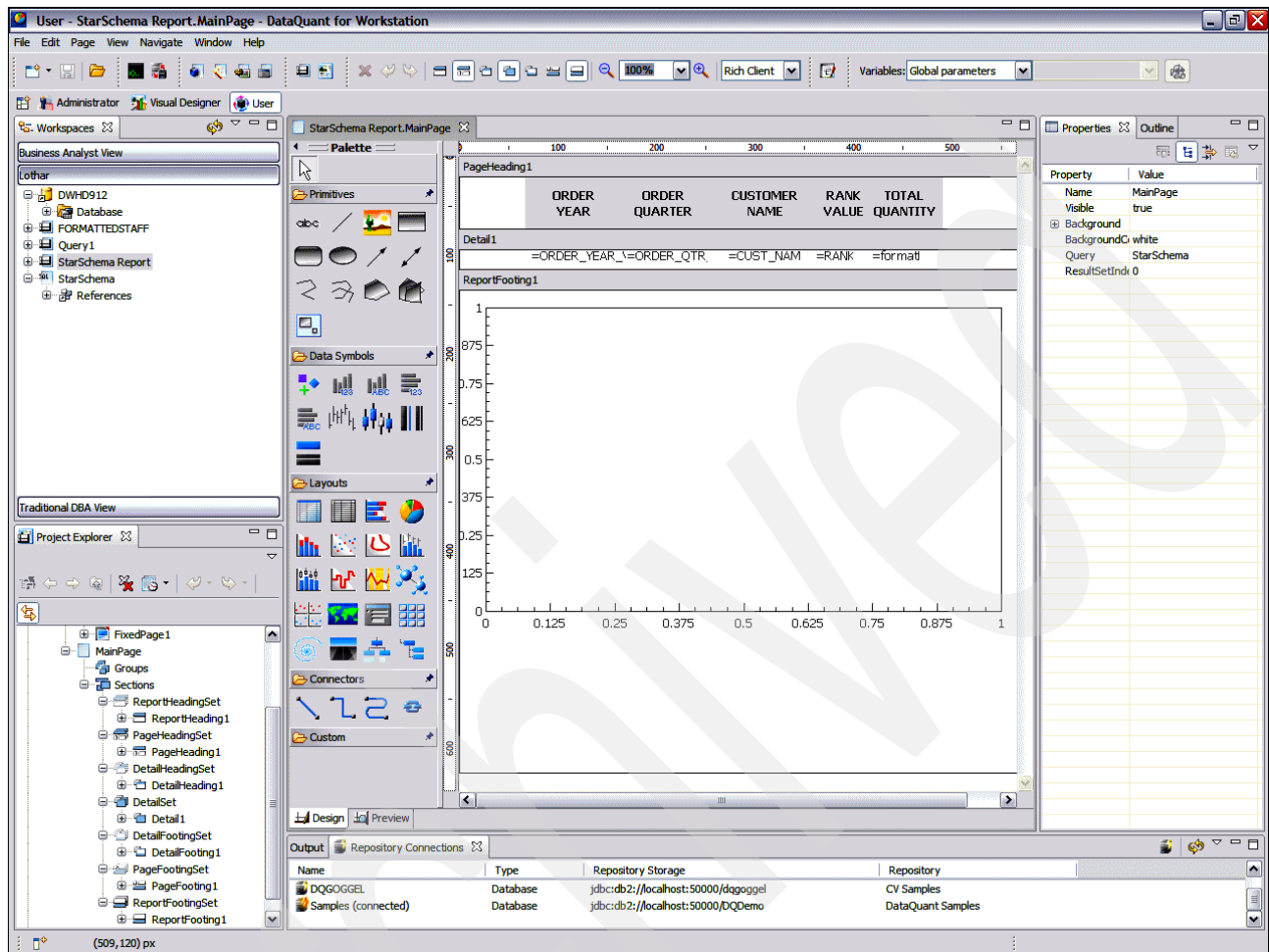


Figure 14-9 DataQuant - Design view, adding the bar chart



After aligning the tabular columns and the bar chart, you see the result as shown in Figure 14-10.

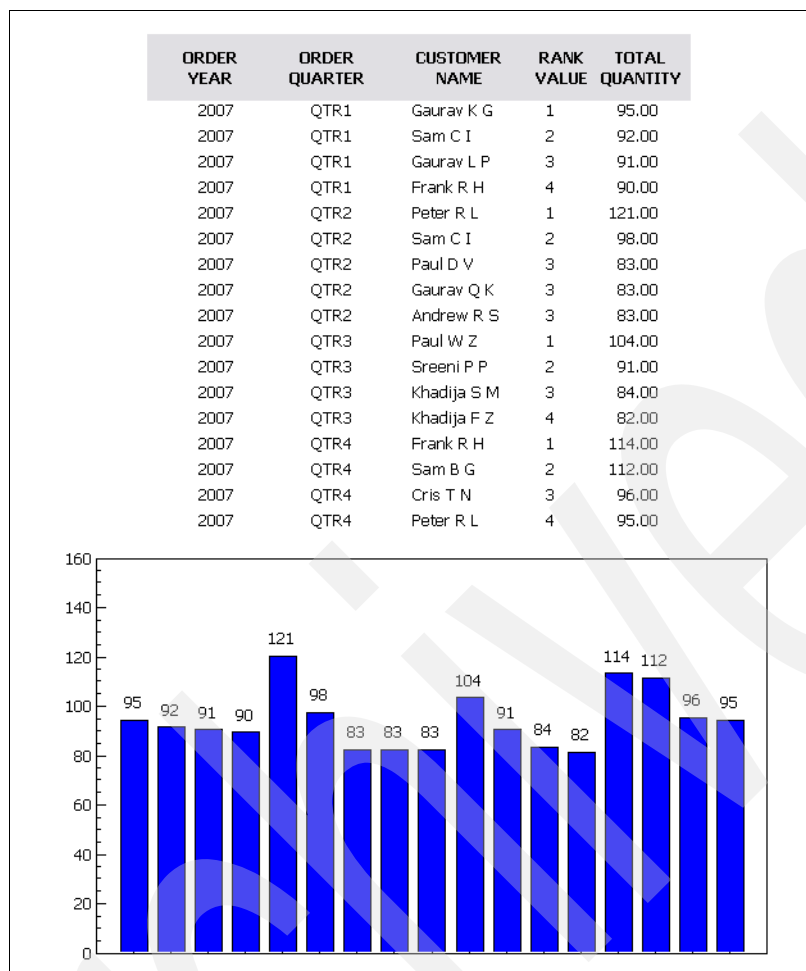


Figure 14-10 DataQuant - RedParts report

This completes the small report for RedParts Distribution.

## 14.2 QMF

As mentioned previously, one big advantage of DataQuant is its ability to use already existing QMF objects. QMF has been available many years and is a family of products.

The QMF family consists of the following products:

- DB2 QMF for TSO/CICS

DB2 QMF for TSO/CICS is tightly integrated with the z/OS system architecture. It provides a fluid yet secure environment where applications, data, and business processes have access to existing resources managed with great flexibility, higher levels of utilization, and lower overall costs. Customers who run DB2 QMF for TSO/CICS are well positioned to deal with the exponential explosion of data, with the capability to sense and respond to market shifts and changing demands as they occur. QMF for TSO/CICS provides powerful data access, manipulation, and presentation functions that scale to many different

database knowledge levels and can be customized in many ways to meet specific business needs.

- ▶ **DB2 QMF for Workstation**

QMF for Workstation is an Eclipse-based desktop application that provides a development environment within which all QMF objects can be created, managed, and executed. QMF for Workstation extends DB2 QMF function to virtually all workstation operating systems. QMF for Workstation offers multiple interfaces that help you build relational and OLAP queries according to your level of SQL expertise. When query result data is returned, an assortment of intuitive editors can help you analyze, aggregate, and format the query results online. You can also create and easily distribute reports that can vary in format from classic paged reports to more visually rich formats. QMF for Workstation's administrative features make it easy to configure connectivity to data sources and protect resource consumption on a per-user and per-group basis.

- ▶ **DB2 QMF for WebSphere**

DB2 QMF for WebSphere is the DB2 QMF family's browser-based portal to business information on demand. As a Web application, QMF for WebSphere provides a substantial subset of the QMF for Workstation query and reporting capabilities using a pure HTML thin-client deployment model. QMF for WebSphere makes it easy to provide the most frequently used QMF query and reporting capabilities to a large number of users quickly and easily. The thin-client model eliminates the need to install or maintain any additional software on multiple user machines. You can access QMF for WebSphere from any machine that has a Web browser. Support is provided for a variety of Web browsers across a number of platforms.

The functions that are provided by DB2 QMF for Workstation and DB2 QMF for WebSphere in Version 9.1 are a subset of the functions provided by DataQuant for Workstation and DataQuant for WebSphere in Version 1.2 respectively. Both products provide the same user interface and the same support to create queries and reports. They also share the same governor functions and connectivity. The big difference is the extent of the support for visual reports and dashboards. DataQuant provides a lot of graphical elements that can be used to create visual reports such as bar charts, pie charts, horizon charts, and candle charts. The interactive dashboards are only provided by DataQuant but not by QMF.

- ▶ **DB2 QMF High Performance Option for TSO/CICS**

DB2 QMF High Performance Option is a multifaceted tool that helps database administrators manage the QMF environment. DB2 QMF HPO consists of two major components:

- **QMF HPO/Manager**

By using QMF HPO/Manager, you can govern (pre-emptively and in real time) ad hoc and dynamic query and reporting activities. With easily collected, detailed information, you can more precisely control CPU resource usage at varying levels according to any number of schedules applied to QMF user groups.

- **QMF HPO/Compiler**

By using QMF HPO/Compiler, you can track and identify heavily used queries and reports and then automatically convert them into efficient COBOL programs. Most query and reporting jobs can be greatly streamlined to reduce CPU resource consumption, DB2 catalog contention, DB2 Optimizer overhead, and dynamic SQL security concerns.

## 14.3 AlphaBlox

AlphaBlox V9.5 for Linux, UNIX, and Windows is a platform for the rapid assembly and broad deployment of integrated analytics embedded within applications. It has an open, extensible architecture based on Java 2 platform, Enterprise Edition (J2EE) standards, and an industry standard for developing Web-based enterprise applications. AlphaBlox leverages AJAX technology, automatically handling many details of application behavior without the need for complex programming.

### 14.3.1 When to consider AlphaBlox

Consider using AlphaBlox in the following situations:

- ▶ Where there is a need for customized analytical solutions that are tightly integrated and embedded within the customer's existing BI infrastructure
- ▶ Where inline analytics are required; that is where analytical application content is embedded in a business process
- ▶ For customers who require full control of the solution's appearance and behavior, from data gathering and analysis to the appearance and usage of the user interface
- ▶ Where there is a need to include sophisticated analytical capabilities
- ▶ As a development environment for customized Web applications instead of packaged

### 14.3.2 AlphaBlox functions

AlphaBlox V9.5 delivers various modular and reusable Blox® components, as well as an application framework, a powerful programming model, and a variety of development tools for assembling analytic applications. For its runtime environment, AlphaBlox leverages standard J2EE application servers. It can be installed on leading commercial J2EE application servers such as WebSphere Application Server and others.

AlphaBlox V9.5 for Linux, UNIX, and Windows now enables developers to build analytic, browser-based applications with a new, easy-to-use, point-and-click user interface. It also provides functionality to make it easier to design, develop, and test analytic applications and new features to help users interact with data and perform calculations.

AlphaBlox provides the ability to rapidly create custom, Web-based applications that fit into the corporate infrastructure and have the ability to reach a wide range of users, both inside and outside the corporate firewall. Applications that are built with the AlphaBlox platform run in standard Web browsers, allowing real-time, highly customizable multidimensional analysis in a Web browser.

The following features, among others, are available in the AlphaBlox platform:

- ▶ The ability to access and interact with data in multidimensional and relational databases
- ▶ The ability to create structured reports sourced from relational databases
- ▶ The ability to choose from a variety of charts to display data
- ▶ The ability to create applications that write data back to the database, which is particularly useful in "what-if" financial planning applications
- ▶ With multidimensional data sources, the ability to allow users to interact with the different levels of data (for example, filter and drill down) to interactively display the exact view of the data desired

- ▶ User access to an intuitive user interface, making analysis of the data easy and powerful
- ▶ Access to multiple data sources by a single application
- ▶ Integration into a variety of enterprise infrastructure components, including application servers (WebSphere and BEA WebLogic)
- ▶ The availability of a variety of APIs so developers can create custom applications

AlphaBlox APIs are written in the Java programming language. Application developers can access them by using Java executed on the server or via JavaScript™ that is interpreted in the browser.

### 14.3.3 A small AlphaBlox Web application for RedParts Distribution

Based on the AlphaBlox installation that we had on Linux on System z, we create a small sample that issues a query against the DDS database. The following steps are required to create an AlphaBlox application:

1. Define the data source.
2. Define an empty application.
3. Create an application home page.
4. Create the Blox.
5. Embed the Blox in a JavaServer Page (JSP™).

#### Defining the data source

AlphaBlox can work with data sources that are defined in AlphaBlox or in WebSphere Application Server. For this sample, we use the AlphaBlox internal data source definition.

To define the data source:

1. Go to the DB2 AlphaBlox home page of your installation. In our installation, we type the following URL:  
<http://1nxdwh1:9080/AlphabloxAdmin/home>
2. Click the **Administration** tab and the **Data Sources** subtab.
3. Click the **create** button.
4. In the Data Sources panel, enter the configuration values for this data source definition as shown in Figure 14-11 on page 413.
5. Save the entries, and test the access to the data source.

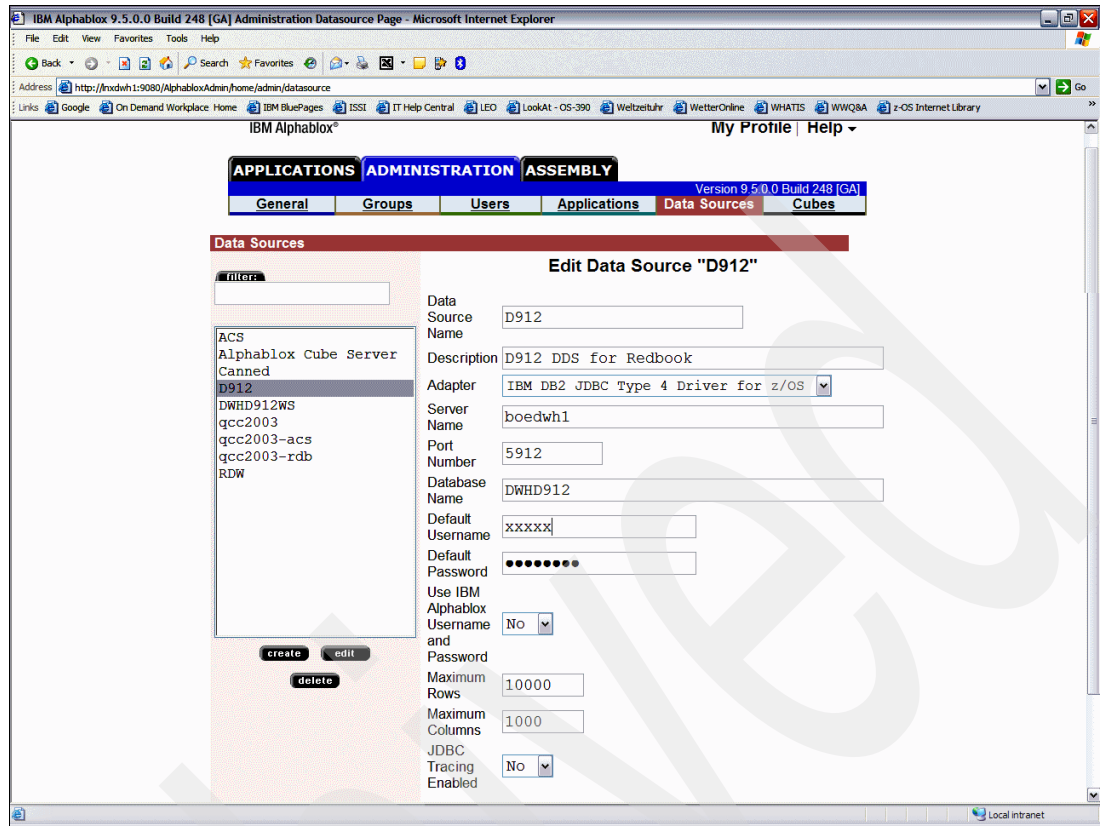


Figure 14-11 AlphaBlox - Defining the data source

## Defining your application

To create an application by using the J2EE development approach, you must create a directory structure with a WEB-INF directory that contains an application descriptor file (web.xml). The simplest way to create this structure in DB2 AlphaBlox is to create a new application by using the Application page within the DB2 AlphaBlox Admin Pages.

1. Create the application RedParts and folder:
  - a. Log into the DB2 AlphaBlox Admin Pages as a user who is a member of the administrators group.
  - b. Click the **Administration** tab and the **Applications** subtab.
  - c. Click the **create** button.

- d. In the Create Application panel, complete the configuration values as shown in Figure 14-12.
- e. Click **Save** to create the application.

IBM Alphablox® My Profile | Help ▾

**APPLICATIONS ADMINISTRATION ASSEMBLY**

Version 9.5.0.0 Build 248 [GA]

**Applications**

filter:

AbxRetail  
AlphabloxReporting  
AlphabloxTooling  
BloxBuilder  
BloxSampler  
DHTMLQueryBuilder  
EMail  
FastForward  
MyApp

**Create Application**

Name: RedParts

Import J2EE Application: No

Display Name: RedParts Distribution

Description:

Home URL:

Image URL:

To test Home and Image URLs, save the application then edit the application

Default Saved State:

Write Privileges:

Security Role:

Session Timeout:

Restore Saved Application State: No

Include on Applications Page: Yes

create edit

Figure 14-12 AlphaBlox - Creating the application

2. Install the application in WebSphere Application Server. In the previous task, AlphaBlox created the Web application /opt/IBM/Alphablox/installableApps/RedParts.ear. This Web application must be deployed in the WebSphere Application Server. To deploy this application:
  - a. Open the WebSphere Administrative Console.
  - b. Select **Applications** → **Install New Application**.

- c. In the Preparing for application installation panel (Figure 14-13), select **Remote file system** and specify the path to the AlphaBlox application. Click **Next** to complete the definition.

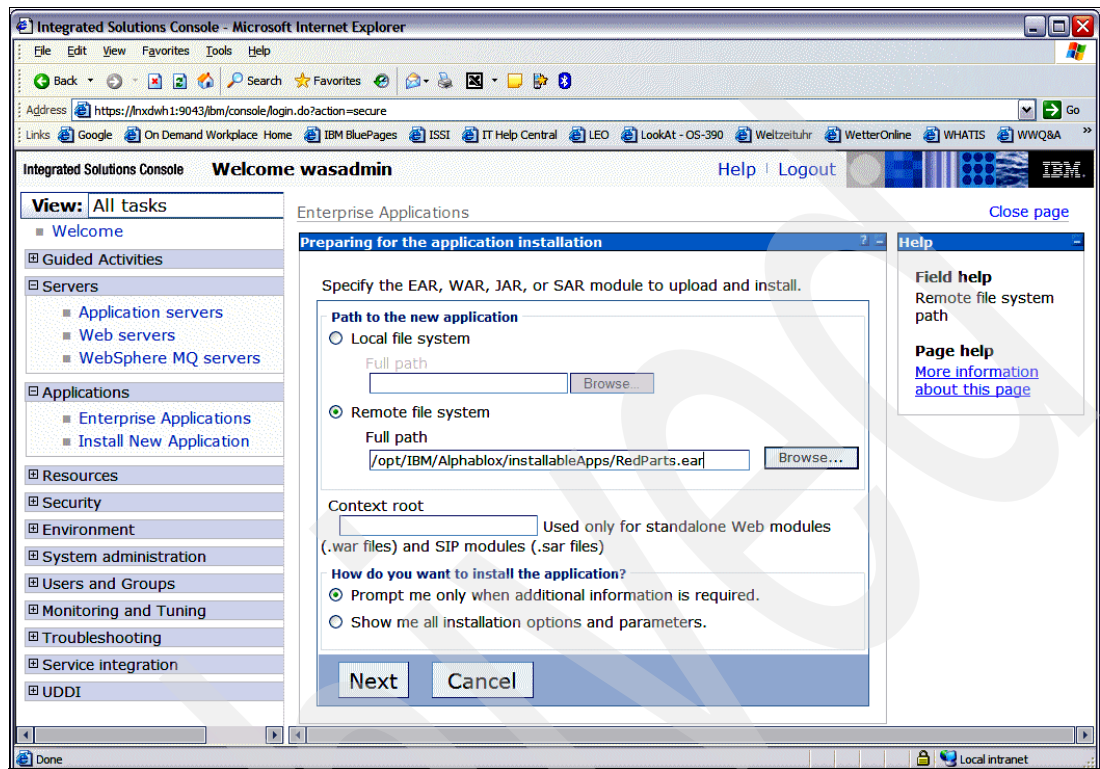


Figure 14-13 WebSphere Administrative Console - Deploying the application

3. Define the User Role mapping for this new application. This mapping defines the roles that an authenticated user can assume. The user can have either of the following roles:

- AlphaBlox Administrator
- AlphaBlox User

To define the user role mapping, in the WebSphere Administrative Console, select **Applications** → **Enterprise Applications**. Then select **RedParts** → **Security role to user/group mapping**.

In the Enterprise Applications panel (Figure 14-14), each user is defined that is authenticated by the WebSphere Application Server as a valid AlphaBlox user.

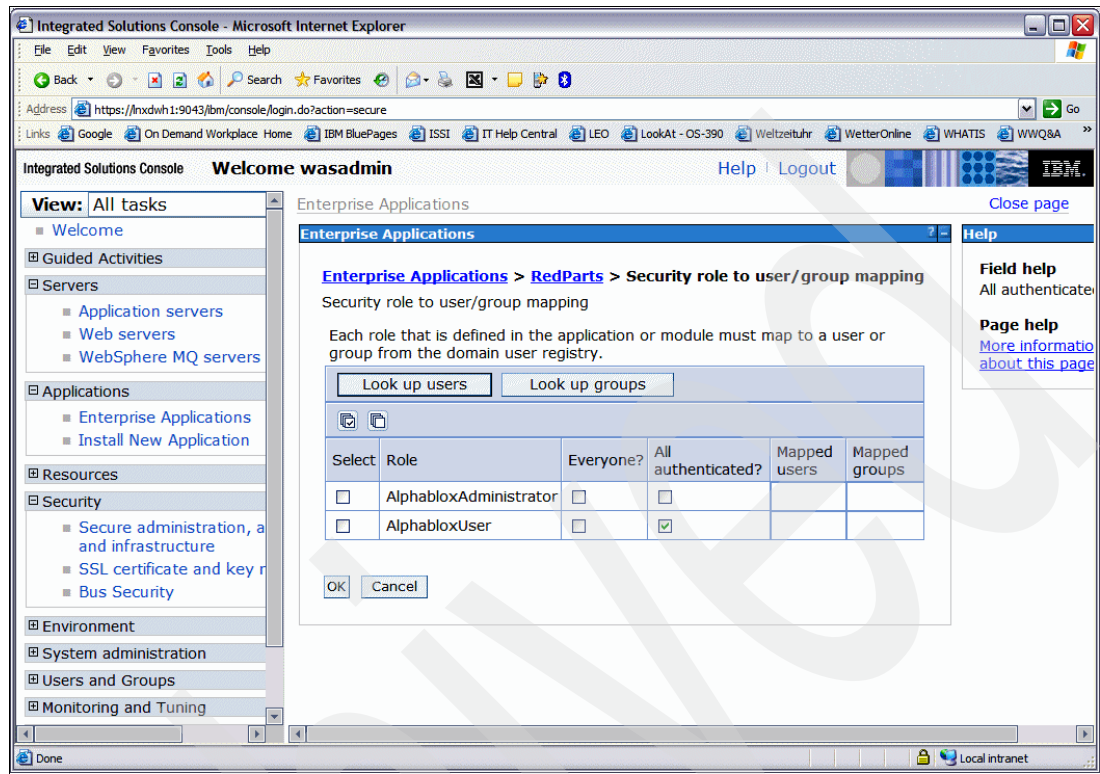


Figure 14-14 WebSphere Administrative Console - Defining user roles for the application



After completion of the definition steps, the application is successfully installed in the Application Server. Figure 14-15 shows the AlphaBlox Application page. Because AlphaBlox is already installed and configured for other applications, under RedParts Distribution, you see other applications, such as the My App and BI Retail Solution, and AlphaBlox system applications such as AlphaBlox Query Builder.

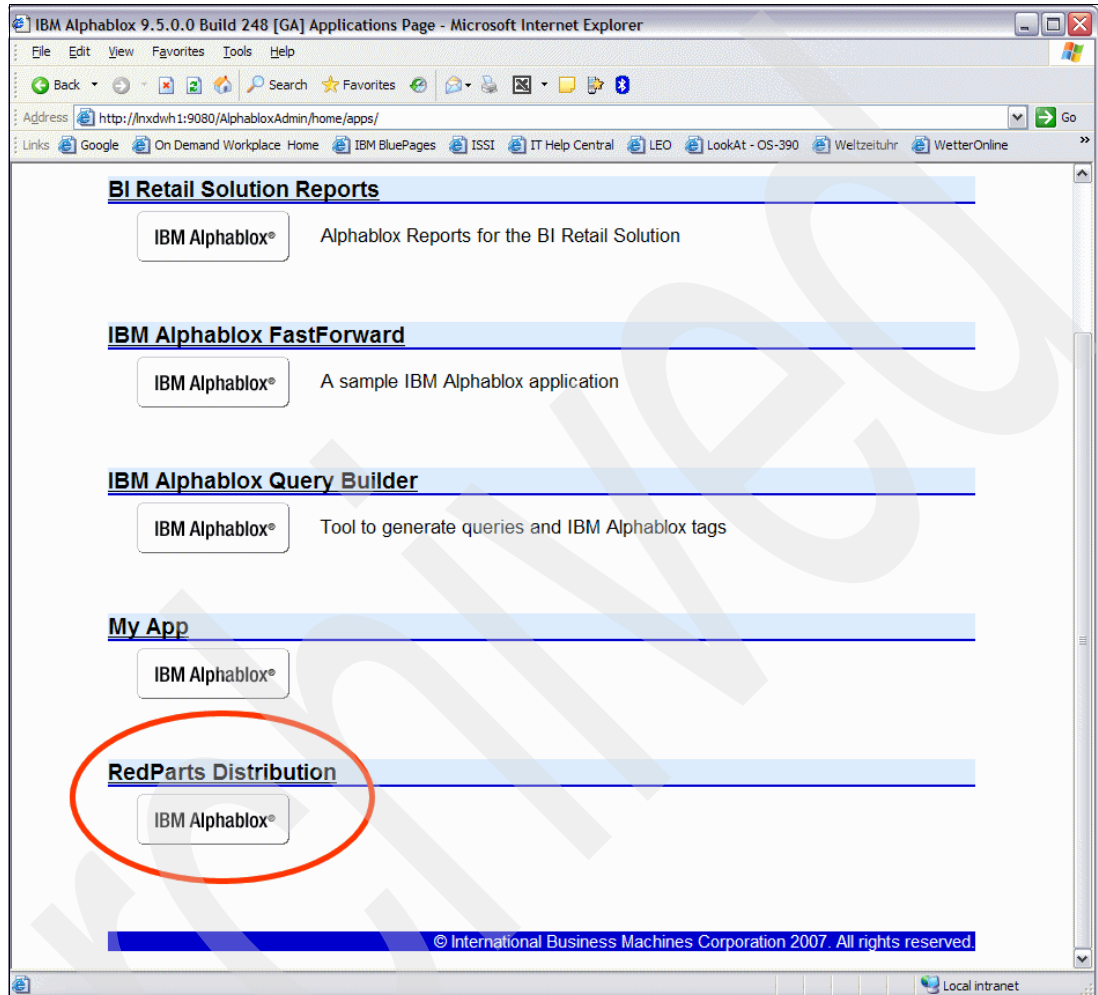


Figure 14-15 AlphaBlox list of applications

4. Create a home page for the Application in the WebSphere Directory. The home page is a simple HTML file (Example 14-2) that displays a title and invokes the PresentBloxView.jsp that contains the query and display functions.

*Example 14-2 RedParts home page*

```
/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/installedApps/lnxdwh1Node01Cell/RedParts.ear/RedParts.war/index.html
```

```
<html>
<head>
<title>ReadParts Distribution</title>
</head>
<body>
<h2>RedParts Distribution Application</h2>
<p>
```

```
<a href="PresentBloxView.jsp">Simple PresentBlox View</a>
</p>
</body> </html>
```

5. Create the Blox and the PresentBloxView.jsp to display the Blox:

- Open a browser to the DB2 AlphaBlox Admin Pages. By default, the Applications tab is visible.
- Click **DB2 AlphaBlox Query Builder** to open the application.
- Connect to the database:
  - Click the **Connection Settings** button.
  - In the Database Connection window, select the D912 data source. Click the **Connect** button. The window closes, and you see that "Connected" is not longer displayed in the Database Status section of the Query Builder page.
- Paste the query into the query window. Click **Execute Query**. The query is executed on z/OS. The result is displayed in the Query Builder as shown in Figure 14-16.

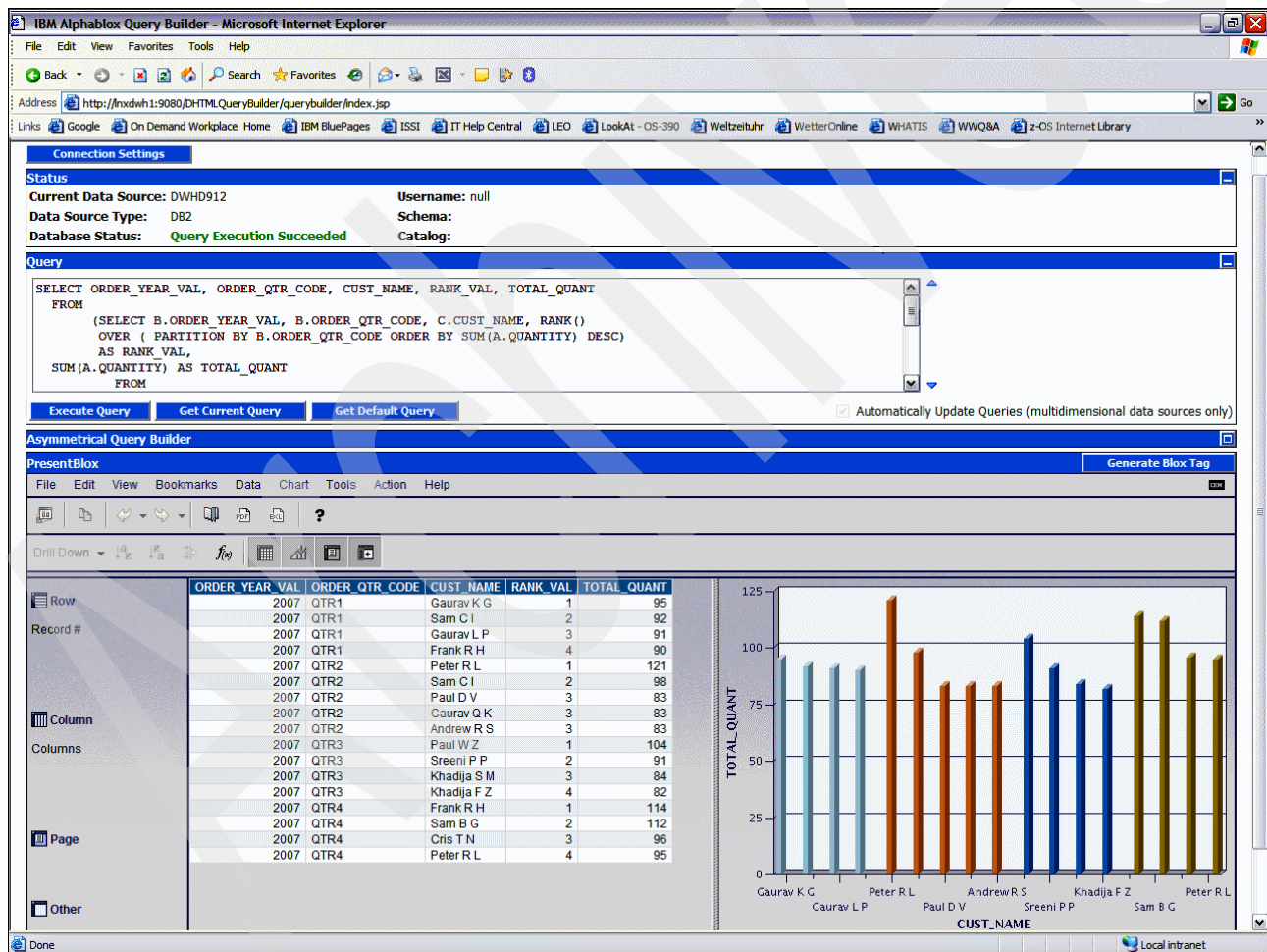


Figure 14-16 AlphaBlox Query Builder

Example 14-3 shows the query that we used.

*Example 14-3 AlphaBlox sample query*

```
SELECT ORDER_YEAR_VAL, ORDER_QTR_CODE, CUST_NAME, RANK_VAL, TOTAL_QUANT
FROM
  (SELECT B.ORDER_YEAR_VAL, B.ORDER_QTR_CODE, C.CUST_NAME, RANK()
   OVER ( PARTITION BY B.ORDER_QTR_CODE ORDER BY SUM(A.QUANTITY) DESC)
   AS RANK_VAL,
   SUM(A.QUANTITY) AS TOTAL_QUANT
   FROM
     DWHDDS.ORDER_TRANSACTION_FACT AS A
     INNER JOIN DWHDDS.ORDERDATE_DIM_VW AS B ON
     A.ORDERDATEKEY = B.ORDERDATEKEY
     INNER JOIN DWHDDS.CUSTOMER_DIM AS C ON
     A.CUSTKEY_DW = C.CUSTKEY_DW
   WHERE B.ORDER_YEAR_VAL = 2007
   GROUP BY B.ORDER_YEAR_VAL, B.ORDER_QTR_CODE, C.CUST_NAME)
   AS QTR_RANKING
WHERE RANK_VAL < 5
ORDER BY ORDER_QTR_CODE, RANK_VAL
```

6. Create the Blox and the JSP that displays the Blox. While you are still in the Query Builder, click the **Create Blox** button. AlphaBlox creates the Blox for you as shown in Figure 14-17.

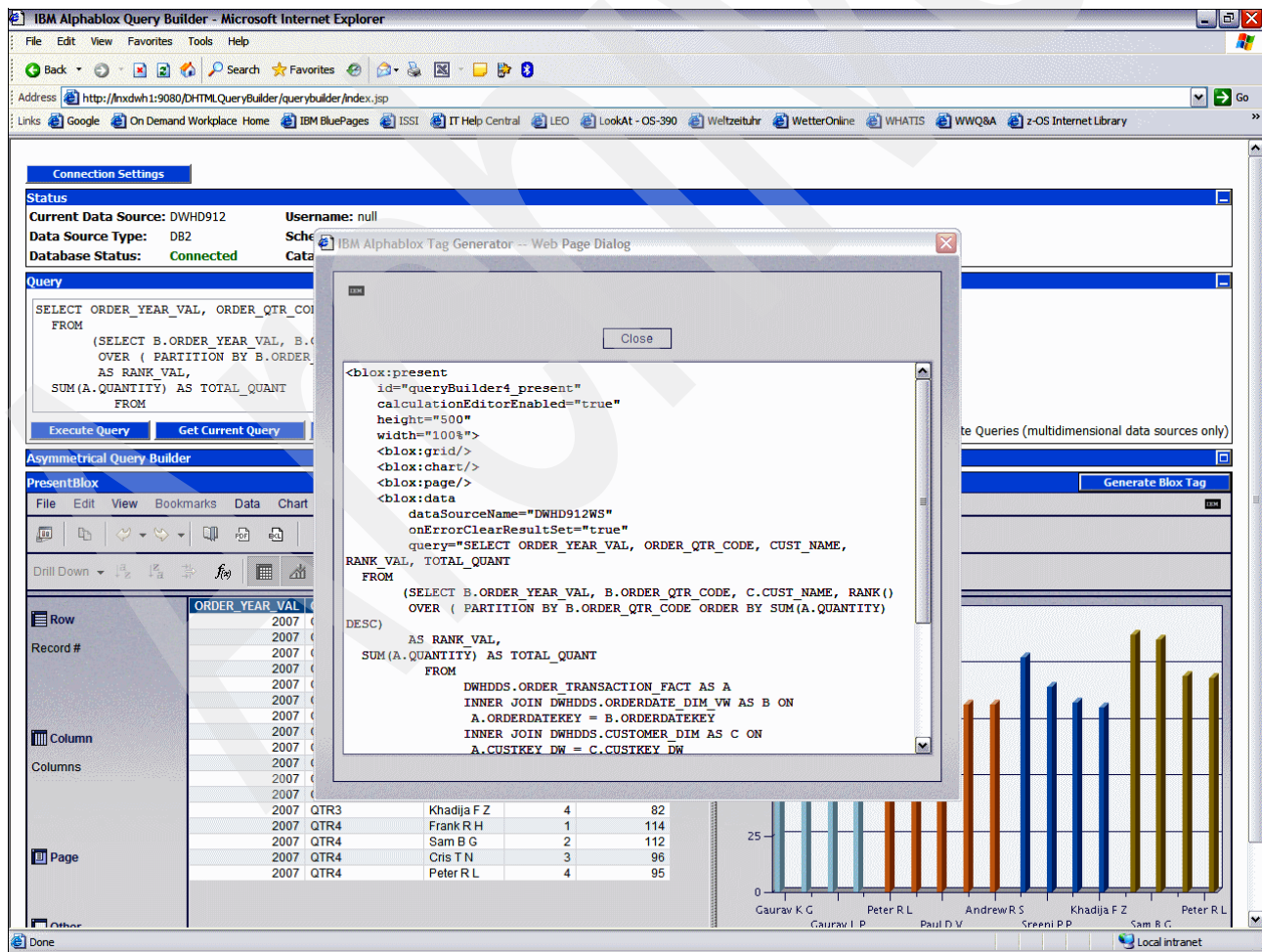


Figure 14-17 AlphaBlox Blox builder output

7. Use the text as created to build the JSP file:

- a. Add the taglib as first line in the JSP.
- b. Add the Blox as created previously.
- c. Add the HTML code to display the Blox.

Example 14-4 shows the complete JSP.

*Example 14-4 RedParts JSP*

/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/installedApps/Inxdwh1Node01Cell/RedParts.ear/RedParts.war/PresentBloxView.jsp

```
<%@ taglib uri="bloxtld" prefix="blox" %>
<blox:present
  id="queryBuilder4_present"
  calculationEditorEnabled="true"
  height="500"
  visible="false"
  width="100%">
  <blox:grid/>
  <blox:chart/>
  <blox:page/>
  <blox:data
    dataSourceName="DWH0912WS"
    onClearResultSet="true"
    query="SELECT ORDER_YEAR_VAL, ORDER_QTR_CODE, CUST_NAME, RANK_VAL, TOTAL_QUANT
FROM
(SELECT B.ORDER_YEAR_VAL, B.ORDER_QTR_CODE, C.CUST_NAME, RANK(
OVER ( PARTITION BY B.ORDER_QTR_CODE ORDER BY SUM(A.QUANTITY) DESC)
AS RANK_VAL,
SUM(A.QUANTITY) AS TOTAL_QUANT
FROM
      DWHDDS.ORDER_TRANSACTION_FACT AS A
      INNER JOIN DWHDDS.ORDERDATE_DIM_VW AS B ON
      INNER JOIN DWHDDS.ORDERDATE_DIM_VW AS B ON
      A.ORDERDATEKEY = B.ORDERDATEKEY
      INNER JOIN DWHDDS.CUSTOMER_DIM AS C ON
      A.CUSTKEY_DW = C.CUSTKEY_DW
WHERE B.ORDER_YEAR_VAL = 2007
GROUP BY B.ORDER_YEAR_VAL, B.ORDER_QTR_CODE, C.CUST_NAME)
AS QTR_RANKING
WHERE RANK_VAL < 5
ORDER BY ORDER_QTR_CODE, RANK_VAL
"
    useAliases="true"/>
  <blox:toolbar/>
  <blox:dataLayout/>
  <bloxui:calculationEditor />
</blox:present>
<html>
<head>
<blox:header/>
</head>
<body>
<h2>RedParts Distribution Blox</h2>
<p>
```



```

<blox:display bloxRef="queryBuilder4_present"/>
</p>
</body>
</html>

```

After this JSP is completed, the small Web application is finished. To view the JSP, type the URL in a browser and enter the query. See Figure 14-18.

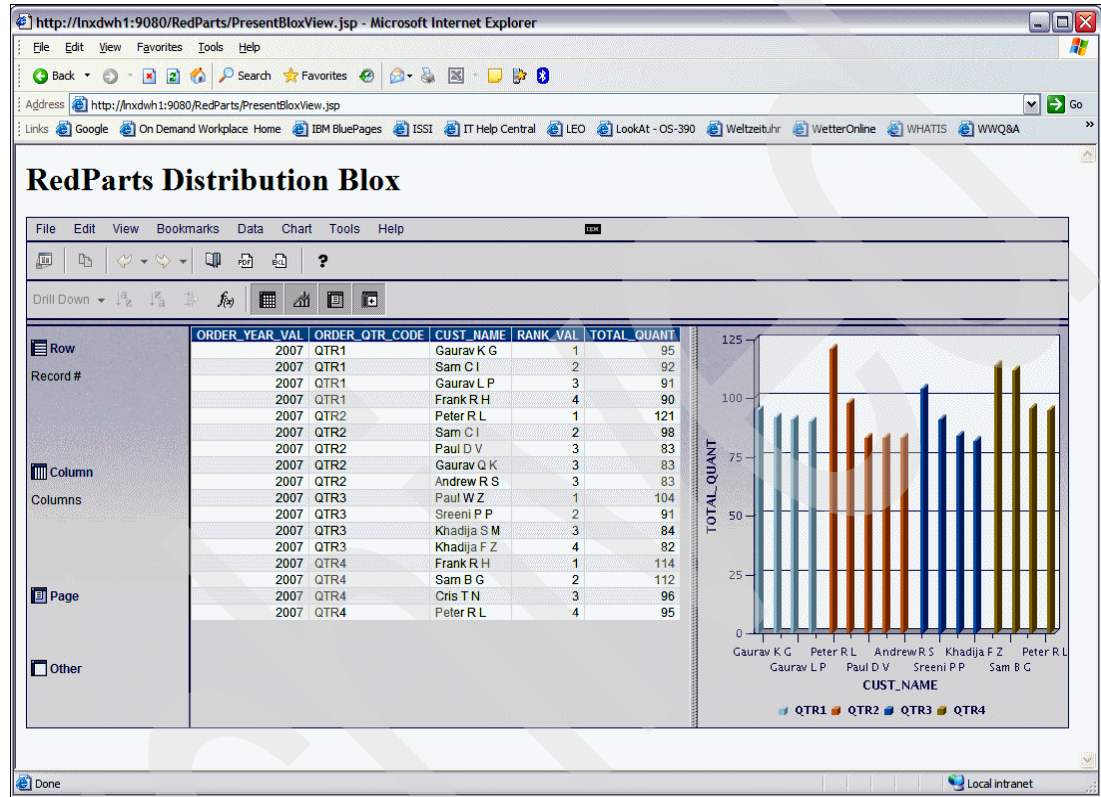


Figure 14-18 AlphaBlox - Output of RedParts JSP

#### 14.3.4 Recommendations to configure AlphaBlox on Linux on System z

AlphaBlox is an application that executes in the Web server environment. In our implementation, this is the WebSphere Application Server on Linux on System z. AlphaBlox comes with a built-in cubing engine, which executes on Linux on System z and provides a cache for the active cubes. This cache can be enabled and disabled for each cube. If the cache is enabled, AlphaBlox retrieves the data once by using SQL and Distributed Relational Database Architecture (DRDA) from z/OS. Any further request to the cube is processed completely in Linux on System z. If the cache is disabled, each request results in an SQL query that must be processed by DB2 on z/OS. This has two disadvantages:

- ▶ Longer response time
- ▶ CPU utilization on z/OS

Therefore, we make the following recommendations:

► Recommendation 1

Activate the cache for each cube, unless there are good reasons why you want to calculate each cube for each query again.

► Recommendation 2

Because AlphaBlox is a Java application that executes in the Web server environment, the Web server settings define the execution environment for this application. We found that the Java maximum heap size should be at least 1563 MB. Figure 14-19 shows the definition in the WebSphere Administrative Console.

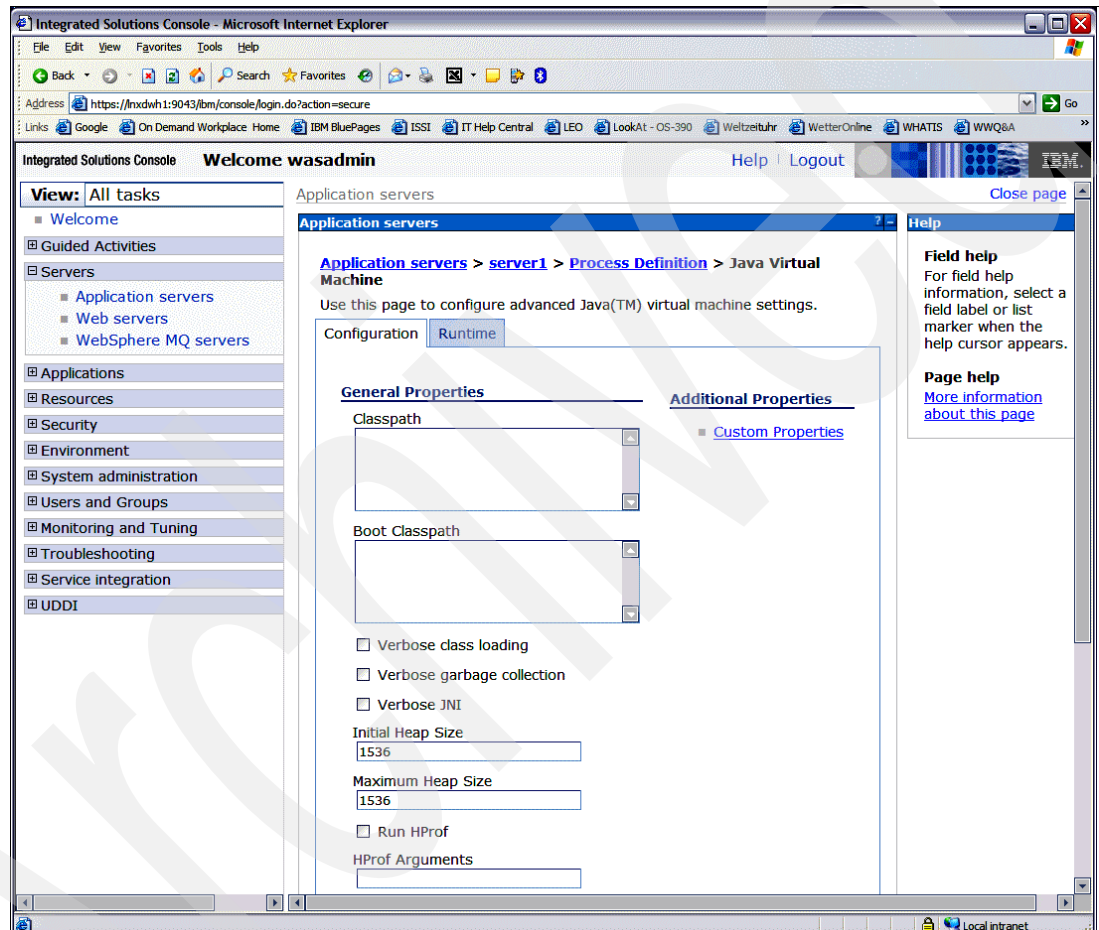


Figure 14-19 WebSphere Administrative Console - Java environment for RedParts application



## Part 3

# Appendixes

In this part, we provide additional information that supports the contents of this book:

- ▶ Appendix A, “Index compression jobs” on page 425
- ▶ Appendix B, “Schema definitions” on page 431
- ▶ Appendix C, “Additional material” on page 447

Archived



## Index compression jobs

In this appendix, we list the jobs that are used for index compression tests. These two examples are used in 7.1, “Index compression” on page 100.

Example A-1 shows one of the programs that was used for these tests.

*Example: A-1 ASM program used for performance evaluation*

```

*-----
* SG24-7637-00 performance asm program sample
*-----
RED01   CSECT
        TITLE 'Redbook performance'
        SAVE (14,12)
        LR   R12,R15
        USING red01,R12,13
        ST   R13,SAVEAREA+4
        LA   R13,SAVEAREA
        B    BEGIN
        EXEC SQL INCLUDE SQLCA
        EXEC SQL INCLUDE SQLDA

*-----
BEGIN    EQU    *
        WTO    ' '
        WTO    'Starting program RED01'

*-----
* Connect to db2
*-----
* Load DSNALI
L_DSNALI DS    0F
        CLC    EPDSNALI,BINZERO
        BNE    CONNDB2
        LOAD   EP=DSNALI
        ST     R0,EPDSNALI
        B      CONNDB2

* Connect to DB2
CONNDB2 DS    0H
        LA     R5,SUBSYSTEM
        LA     R6,TECB

```



```

        CLC    SQLCODE,=F'0'
        BNE    OUTLOOP2
* Print retrieve data to console
        MVC    BUFFER24,blanks
        MVC    BUFFER24,DATA001
        MVC    WT0002+12(24),BUFFER24
WT0002   WTO    '
* End print
        B      FETCH
OUTLOOP2 EQU    *
        L      R2,SQLCODE
        CVD    R2,AUXP
        UNPK   AUXZ,AUXP
        OI     AUXZ+7,X'F0'
        MVC    TRACE,BLANKS
        MVC    TRACE(15),=C'End SQL CODE  :'
        MVC    TRACE+16(8),AUXZ
        MVC    BUFFER24,TRACE
        MVC    WT0003+12(24),BUFFER24
WT0003   WTO    '--->

        EXEC   SQL CLOSE CUR001

*-----
* End program
*-----
PGMEXIT EQU    *
        WTO    'Closing PGM RED01'
        WTO    ' '
        L      R13,SAVEAREA+4
        RETURN (14,12),RC=0

*-----
* Definitions
*-----
EPDSNALI DC    A(0)
BINZERO  DC    F'0'
SUBSYSTM DC    CL4'D912'
PLANNM   DC    CL8'RED01 '
CONNECT  DC    CL12'CONNECT '
OPEN     DC    CL12'OPEN '
DISC     DC    CL12'DISCONNECT '
CALLPA   DS    0F
        CALL  ,(*,*,*,*,*),VL,MF=L
CAFCALL  CALL  ,(*,*,*,*,*),VL,MF=L
CAFLEN   EQU    *-CAFCALL
TECB     DS    F
SECB     DS    F
RIBPTR   DS    F
CALLING  DS    F
SAVE_R1  DS    F
SAVE_R15 DS    F
SAVE_R0  DS    F
BUFFER24 DS    c124
AUXP     DS    PL8
AUXZ     DS    ZL8
DATA001  DS    CL18
DATA002  DS    CL18
NUMBR01  DS    PL8
NUMBR02  DS    PL8
TRACE    DS    CL80

```

```

BLANKS   DC      80C' '
SAVEAREA DS      18F
*-----
SQLAREA  EQU     *-SQLDSECT
R0        EQU     0
R1        EQU     1
R2        EQU     2
R3        EQU     3
R4        EQU     4
R5        EQU     5
R6        EQU     6
R7        EQU     7
R8        EQU     8
R9        EQU     9
R10       EQU    10
R11       EQU    11
R12       EQU    12
R13       EQU    13
R14       EQU    14
R15       EQU    15
          END

```

---

Example A-2 shows the JCL for assembly, bind, and execution.

*Example: A-2 JCL sample for assembly, bind and execution of performance model program*

---

```

//CRISDRED JOB (),'JOB HERE'....
//*-----
/* CHANGE RED01 BY PGM NAME
/*-----
/* DB2 PRECOMPILE
/*-----
//DB2PRE EXEC PGM=DSNHPC,PARM='HOST(ASM),SOURCE,APOST
//              VERSION(),ATTACH(CAF)'
//STEPLIB DD DSN=SYS1.SCEELKED,DISP=SHR
//         DD DSN=SYS1.DSN.V910.SDSNEXIT,DISP=SHR
//         DD DSN=SYS1.DSN.V910.SDSNLINK,DISP=SHR
//         DD DSN=SYS1.DSN.V910.SDSNLOAD,DISP=SHR
//         DD DSN=SYS1.DSN.V910.SDXRRESL,DISP=SHR
//DBRMLIB DD DSN=CRIS.UTIL.ASM.DBRM(RED01),DISP=(OLD,PASS),
//         DCB=(RECFM=FB,LRECL=80),SPACE=(TRK,(2,2,4),RLSE)
//SYSPRINT DD SYSOUT=*
//SYSTEM   DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSUT1   DD SPACE=(CYL,(1,1)),UNIT=VIO
//SYSUT2   DD SPACE=(CYL,(1,1)),UNIT=VIO
//SYSIN    DD DSN=CRIS.UTIL.ASM(RED01),DISP=SHR
//SYSCIN   DD DSN=&&OUTDB2,DISP=(MOD,PASS),
//         SPACE=(CYL,(5,1)),UNIT=VIO
//*-----
/* PRINT OUTPUT DB2 PRECOMPILER
/*-----
/* IF (RC LE 0) THEN
//PRINTSRC EXEC PGM=IEBGENER
//SYSPRINT DD DUMMY
//SYSUT1   DD DSN=&&OUTDB2,DISP=(OLD,PASS)
//SYSUT2   DD SYSOUT=*
//SYSIN    DD DUMMY
/* ENDIF
/*-----

```

```

// IF (RC LE 4) THEN
//AASM      EXEC PGM=ASMA90,PARM=(OBJECT,NODECK)
//SYSLIB    DD DSN=CRIS.UTIL.ASM,DISP=SHR
//          DD DSN=SYS1.SCEEMAC,DISP=SHR
//          DD DSN=SYS1.SCEESAMP,DISP=SHR
//          DD DSN=SYS1.MACLIB,DISP=SHR
//SYSUT1    DD UNIT=3390,SPACE=(TRK,(5,15))
//SYSUT2    DD UNIT=3390,SPACE=(TRK,(5,15))
//SYSUT3    DD UNIT=3390,SPACE=(TRK,(5,15))
//SYSLIN    DD DSN=&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,
//          SPACE=(80,(8000,5000))
//SYSPRINT  DD SYSOUT=*,
//          DCB=(RECFM=FB,LRECL=121,BLKSIZE=2420)
//SYSUT1    DD DSN=&SYSUT1,UNIT=SYSDA,
//          SPACE=(1024,(200,50),,CONTIG,ROUND),DCB=BLKSIZE=1024
/* SOURCE FOLDER
//SYSIN     DD DSN=&&OUTDB2,DISP=SHR
//      ENDIF
/*-----
// IF (RC LE 4) THEN
//LKED      EXEC PGM=IEWL,COND=(8,LT),
//          PARM='SIZE=(4096K,512K),XREF,AMODE=31,RMODE=ANY'
//SYSLIB    DD DSN=SYS1.SCEELKED,DISP=SHR
//          DD DSN=SYS1.DSN.V910.SDSNEXIT,DISP=SHR
//          DD DSN=SYS1.DSN.V910.SDSNLINK,DISP=SHR
//          DD DSN=SYS1.DSN.V910.SDSNLOAD,DISP=SHR
//          DD DSN=SYS1.DSN.V910.SDXRRESL,DISP=SHR
//SYSUT1    DD DSN=&SYSUT1,UNIT=SYSDA,
//          SPACE=(1024,(200,50),,CONTIG,ROUND),DCB=BLKSIZE=1024
/* MODULE PDS
//SYSLMOD   DD DSN=CRIS.UTIL.ASM.MOD(RED01),DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSLIN    DD DSN=&LOADSET,DISP=(OLD,DELETE)
//      ENDIF
/*-----
/*      BIND
/*-----
// IF (RC LE 4) THEN
//BINDPK    EXEC PGM=IKJEFT1A
//STEPLIB   DD DSN=SYS1.SCEELKED,DISP=SHR
//          DD DSN=SYS1.DSN.V910.SDSNEXIT,DISP=SHR
//          DD DSN=SYS1.DSN.V910.SDSNLINK,DISP=SHR
//          DD DSN=SYS1.DSN.V910.SDSNLOAD,DISP=SHR
//          DD DSN=SYS1.DSN.V910.SDXRRESL,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSTSPRT  DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
//DBRMLIB   DD DSN=CRIS.UTIL.ASM.DBRM(RED01),DISP=SHR
//SYSTSIN   DD *
DSN SYSTEM(D912)

BIND PLAN(REDBOOK) -
    QUAL(CRIS) -
    OWNER(CRIS) -
    PKL(*.CRIS.*) -
    NODEFER(PREPARE) -
    VALID(R) -
    ISOL(UR)

BIND PACKAGE (CRIS) -

```

```

OWNER(CRIS) -
QUALIFIER(CRIS) -
MEMBER(RED01) -
SQLERROR(NOPACKAGE) -
VALIDATE(BIND) -
    FLAG(I) -
ISOLATION(CS) -
RELEASE(COMMIT) -
EXPLAIN(YES) -
CURRENTDATA(NO) -
DYNAMICRULES(BIND) -
    ACTION(REPLACE)
/*
//  ENDIF
/*-----
/*  EXECUTE
/*-----
//  IF (RC LE 4) THEN
//EXECUTE EXEC PGM=RED01,PARM=('D912,REDBOOK')
//STEPLIB DD DSN=CRIS.UTIL.ASM.MOD,DISP=SHR
//        DD DSN=SYS1.DSN.D912.SDSNEXIT,DISP=SHR
//        DD DSN=SYS1.DSN.V910.SDSNEXIT,DISP=SHR
//        DD DSN=SYS1.DSN.V910.SDSNLINK,DISP=SHR
//        DD DSN=SYS1.DSN.V910.SDSNLOAD,DISP=SHR
//        DD DSN=SYS1.DSN.V910.SDXRRESL,DISP=SHR
//SYSPRINT DD SYSOUT=*
//  ENDIF

```

---

## Schema definitions

In this appendix, we provide details about the schema definitions that are used in the transactional and data warehouse environments. This appendix includes the following sections:

- ▶ Appendix B.1, “Schema definition for the transactional database” on page 432
- ▶ Appendix B.2, “Schema definition for the data warehouse database” on page 437

The Data Definition Language (DDL) statements for both schema definitions are also available for download as explained in Appendix C, “Additional material” on page 447.

## B.1 Schema definition for the transactional database

Example B-1 shows the data definition file that we use to create the transactional data model in DB2 z/OS that we describe in 5.4, “The transactional data model” on page 78. The script contains drop table statements before each definition that deletes table definitions before it recreates them with each run.

*Example: B-1 Schema definition for the transactional system*

---

```
-----
-- Create schema for OLTP_B system (branches)
-----

--          O R D E R S
-----

DROP TABLE OLTP_B.ORDER;

CREATE TABLE OLTP_B.ORDER
(
    ORDER_KEY INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY(START WITH 2001, INCREMENT BY 2
    CACHE 100),
    CUST_KEY INTEGER,
    STATUS CHAR(1),
    TOTALPRICE DECIMAL(12,2),
    ORDERDATE DATE,
    ORDERPRIORITY CHAR(15),
    CLERK CHAR(15),
    BRANCHKEY INTEGER,
    PRIMARY KEY(ORDER_KEY)
)
IN DBOLTP.TSORDERS ;

GRANT ALL ON TABLE OLTP_B.ORDER TO PUBLIC;

CREATE UNIQUE INDEX OLTP_B.XORDER on OLTP_B.ORDER
(
    ORDER_KEY
)
BUFFERPOOL BP1 ;

-----
--          S H I P M E N T
-----

DROP TABLE OLTP_B.SHIPMENT;

CREATE TABLE OLTP_B.SHIPMENT
(
    ORDER_KEY INTEGER NOT NULL,
    SHIPPRIORITY INTEGER,
    SHIPDATE DATE,
    COMMITDATE DATE,
    RECEIPTDATE DATE,
    SHIPINSTRUCT CHAR(25),
    SHIPMODE CHAR(10),
    PRIMARY KEY( ORDER_KEY )
)
IN DBOLTP.TSSHIP;

GRANT ALL ON TABLE OLTP_B.SHIPMENT TO PUBLIC;
```



```

CREATE UNIQUE INDEX OLTP_B.XSHIP ON OLTP_B.SHIPMENT
(
    ORDER_KEY
)
BUFFERPOOL BP1 ;

-----
--                L I N E I T E M
-----
DROP TABLE OLTP_B.LINEITEM;

CREATE TABLE OLTP_B.LINEITEM
(
    ORDER_KEY INTEGER NOT NULL,
    PART_KEY INTEGER,
    SUPP_KEY INTEGER,
    LINENUMBER INTEGER,
    QUANTITY DECIMAL(12,2),
    EXTENDEDPRICE DECIMAL(12,2),
    DISCOUNT DECIMAL(12,2),
    TAX DECIMAL(12,2),
    RETURNFLAG CHAR(1),
    LINESTATUS CHAR(1)
)
IN DBOLTP.TSLINEIT;

GRANT ALL ON TABLE OLTP_B.LINEITEM TO PUBLIC;

CREATE INDEX OLTP_B.XLINEITEM ON OLTP_B.LINEITEM
(
    ORDER_KEY
)
BUFFERPOOL BP1;

-----
-- Create schema for OLTP_W system (web orders)
-----

-----
--                O R D E R S
-----
DROP TABLE OLTP_W.ORDER;

CREATE TABLE OLTP_W.ORDER
(
    ORDER_KEY INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY(START WITH 2000, INCREMENT BY 2,
    CACHE 100),
    CUST_KEY INTEGER,
    STATUS CHAR(1),
    TOTALPRICE DECIMAL(12,2),
    ORDERDATE DATE,
    ORDERPRIORITY CHAR(15),
    CLERK CHAR(15),
    BRANCHKEY INTEGER,
    PRIMARY KEY(ORDER_KEY)
)
IN DBOLTP.TSORDERS ;

```

```

GRANT ALL ON TABLE OLTP_W.ORDER TO PUBLIC;

CREATE UNIQUE INDEX OLTP_W.XORDER on OLTP_W.ORDER
(
    ORDER_KEY
)
BUFFERPOOL BP1 ;

```

```

-----
--                S H I P M E N T
-----

```

```

DROP TABLE OLTP_W.SHIPMENT;

```

```

CREATE TABLE OLTP_W.SHIPMENT
(
    ORDER_KEY INTEGER NOT NULL,
    SHIPPRIORITY INTEGER,
    COMMITDATE DATE,
    SHIPDATE DATE,
    RECEIPTDATE DATE,
    SHIPINSTRUCT CHAR(25),
    SHIPMODE CHAR(10),
    PRIMARY KEY( ORDER_KEY )
)
IN DBOLTP.TSSHIP;

```

```

GRANT ALL ON TABLE OLTP_W.SHIPMENT TO PUBLIC;

```

```

CREATE UNIQUE INDEX OLTP_W.XSHIP ON OLTP_W.SHIPMENT
(
    ORDER_KEY
)
BUFFERPOOL BP1 ;

```

```

-----
--                L I N E I T E M
-----

```

```

DROP TABLE OLTP_W.LINEITEM;

```

```

CREATE TABLE OLTP_W.LINEITEM
(
    ORDER_KEY INTEGER NOT NULL,
    PART_KEY INTEGER,
    SUPP_KEY INTEGER,
    LINENUMBER INTEGER,
    QUANTITY DECIMAL(12,2),
    EXTENDEDPRICE DECIMAL(12,2),
    DISCOUNT DECIMAL(12,2),
    TAX DECIMAL(12,2),
    RETURNFLAG CHAR(1),
    LINESTATUS CHAR(1)
)
IN DBOLTP.TSLINEIT;

```

```

GRANT ALL ON TABLE OLTP_W.LINEITEM TO PUBLIC;

```

```

CREATE INDEX OLTP_W.XLINEITEM ON OLTP_W.LINEITEM
(
    ORDER_KEY
)

```

```

)
BUFFERPOOL BP1;

-----
-- Create schema for OLTP Customer/Parts/Supplier
-----

-----
--          C U S T O M E R
-----

DROP TABLE OLTP.CUSTOMER;

CREATE TABLE OLTP.CUSTOMER
(
    CUST_KEY INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY(START WITH 1001 INCREMENT BY 2,
    CACHE 100),
    NAME VARCHAR(25),
    ADDRESS VARCHAR(40),
    COUNTRY CHAR(25),
    PHONE CHAR(15),
    ACCTBAL DECIMAL(12,2),
    MKTSEGMENT CHAR(10),
    PRIMARY KEY (CUST_KEY)
)
IN DBOLTP.TSCUSTOM;

GRANT ALL ON TABLE OLTP.CUSTOMER TO PUBLIC;

CREATE UNIQUE INDEX OLTP.XCUST ON OLTP.CUSTOMER
(
    CUST_KEY
)
BUFFERPOOL BP1;

-----
--          P A R T
-----

DROP TABLE OLTP.PART;

CREATE TABLE OLTP.PART
(
    P_PARTKEY INTEGER NOT NULL,
    P_NAME VARCHAR(55),
    P_MFGR CHAR(25),
    P_BRAND CHAR(10),
    P_TYPE VARCHAR(25),
    P_SIZE INTEGER,
    P_CONTAINER CHAR(10),
    P_RETAILPRICE DECIMAL(12,2),
    PRIMARY KEY( P_PARTKEY )
)
IN DBOLTP.TSPART;

GRANT ALL ON TABLE OLTP.PART TO PUBLIC;

CREATE UNIQUE INDEX OLTP.XPART ON OLTP.PART
(
    P_PARTKEY
)

```

```
BUFFERPOOL BP1;
```

```
-----  
--          P A R T S U P P  
-----
```

```
DROP TABLE OLTP.PARTSUPP;
```

```
CREATE TABLE OLTP.PARTSUPP  
(  
    ps_partkey integer not null,  
    ps_suppkey integer not null,  
    ps_availqty integer,  
    ps_supplycost decimal(12,2)  
)  
IN DBOLTP.TSPARTSU;
```

```
GRANT ALL ON TABLE OLTP.PARTSUPP TO PUBLIC;
```

```
CREATE INDEX OLTP.XPARTSU ON OLTP.PARTSUPP  
(  
    PS_PARTKEY, PS_SUPPKEY  
)  
BUFFERPOOL BP1;
```

```
-----  
--          S U P P L I E R  
-----
```

```
DROP TABLE OLTP.SUPPLIER;
```

```
CREATE TABLE OLTP.SUPPLIER  
(  
    S_SUPPKEY INTEGER NOT NULL,  
    S_NAME CHAR(25),  
    S_ADDRESS VARCHAR(40),  
    S_NATIONKEY INTEGER,  
    S_PHONE CHAR(15),  
    S_ACCTBAL DECIMAL(12,2),  
    PRIMARY KEY(S_SUPPKEY)  
)  
IN DBOLTP.TSSUPP;
```

```
GRANT ALL ON TABLE OLTP.SUPPLIER TO PUBLIC;
```

```
CREATE UNIQUE INDEX OLTP.XSUPP ON OLTP.SUPPLIER  
(  
    S_SUPPKEY  
)  
BUFFERPOOL BP1;
```

```
-----  
--          N A T I O N  
-----
```

```
DROP TABLE OLTP.NATION;
```

```
CREATE TABLE OLTP.NATION  
(  
    n_nationkey integer not null,  
    n_name char(25),  
    n_regionkey integer,
```

```

        n_comment varchar(152),
        primary key(n_nationkey)
    )
in DBOLTP.TSNATION;

GRANT ALL ON TABLE OLTP.NATION TO PUBLIC;

CREATE UNIQUE INDEX OLTP.XNATION1 ON OLTP.NATION
(
    n_nationkey
)
BUFFERPOOL BP1;

CREATE UNIQUE INDEX OLTP.XNATION2 ON OLTP.NATION
(
    N_NAME
)
BUFFERPOOL BP1;

-----
--          R E G I O N
-----

DROP TABLE OLTP.REGION;

CREATE TABLE OLTP.REGION
(
    R_REGIONKEY INTEGER NOT NULL,
    R_NAME CHAR(25),
    R_COMMENT VARCHAR(152),
    PRIMARY KEY( R_REGIONKEY)
)
in DBOLTP.TSREGION ;

GRANT ALL ON TABLE OLTP.REGION TO PUBLIC;

CREATE UNIQUE INDEX OLTP.XREGION ON OLTP.REGION
(
    R_REGIONKEY
)
BUFFERPOOL BP1;

```

---

## B.2 Schema definition for the data warehouse database

Example B-2 shows the data definition file that we use to create the operational data store (ODS) of the data warehouse data model in DB2 z/OS that we describe in 5.5, “The operational and dimensional data model” on page 85. Notice the use of the DROP statement for recreation.

*Example: B-2 ODS schema definitions*

---

```

SET CURRENT_SCHEMA = DWHODS;

-----
-- CUSTOMER
-----

DROP TABLE DWHODS.CUSTOMER;
CREATE TABLE DWHODS.CUSTOMER
(
    C_CUSTKEY INTEGER NOT NULL,

```

```

C_NAME VARCHAR(25),
C_ADDRESS VARCHAR(40),
C_NATIONKEY INTEGER,
C_PHONE CHAR(15),
C_ACCTBAL DECIMAL(12,2),
C_MKTSEGMENT CHAR(10),
C_COMMENT VARCHAR(117),
C_LAST_UPDATE TIMESTAMP NOT NULL WITH DEFAULT,
PRIMARY KEY (C_CUSTKEY)
)
IN DBTENG.B.TSCUSTOM;

CREATE UNIQUE INDEX DWHODS.IX_CUSTOMER
  ON CUSTOMER (C_CUSTKEY) USING STOGROUP TENG.B;

-----
-- ORDERS
-----
DROP TABLE DWHODS.ORDERS;
CREATE TABLE DWHODS.ORDERS
(
O_ORDERKEY INTEGER NOT NULL,
O_CUSTKEY INTEGER,
O_ORDERSTATUS CHAR(1),
O_TOTALPRICE DECIMAL(12,2),
O_ORDERDATE DATE,
O_ORDERPRIORITY CHAR(15),
O_CLERK CHAR(15),
O_SHIPPRIORITY INTEGER,
O_COMMENT VARCHAR(79),
O_LAST_UPDATE TIMESTAMP NOT NULL WITH DEFAULT,
O_BRANCHKEY INTEGER,
O_SOURCEKEY CHAR(1) NOT NULL,
PRIMARY KEY(O_ORDERKEY,O_SOURCEKEY)
) IN DBTENG.B.TSORDERS ;

CREATE UNIQUE INDEX DWHODS.IX_ORDERS
  ON ORDERS (O_ORDERKEY) USING STOGROUP TENG.B;

-----
-- NATION_LOOKUP
-----
DROP TABLE DWHODS.NATION_LOOKUP;
CREATE TABLE DWHODS.NATION_LOOKUP
(
N_NATIONKEY INTEGER NOT NULL,
N_NAME CHAR(25),
N_REGIONKEY INTEGER,
N_COMMENT VARCHAR(152),
N_LAST_UPDATE TIMESTAMP NOT NULL WITH DEFAULT,
PRIMARY KEY(N_NATIONKEY)
)
IN DBTENG.B.TSNATION ;

CREATE UNIQUE INDEX DWHODS.IX_NATION_LOOKUP
  ON NATION_LOOKUP (N_NATIONKEY) USING STOGROUP TENG.B;

-----
-- BRANCH_LOOKUP
-----
DROP TABLE DWHODS.BRANCH_LOOKUP;

```

```

CREATE TABLE DWHODS.BRANCH_LOOKUP
(
    B_BRANCHKEY INTEGER NOT NULL,
    B_REGIONKEY INTEGER,
    B_NATIONKEY INTEGER,
    B_BRANCH_TYPE VARCHAR(15),
    B_BRANCH_NAME VARCHAR(25),
    B_LAST_UPDATE TIMESTAMP NOT NULL WITH DEFAULT ,
    PRIMARY KEY (B_BRANCHKEY)
) IN DBTENG.B.TSBRNHLK;

CREATE UNIQUE INDEX DWHODS.IX_BRANCH_LOOKUP
    ON BRANCH_LOOKUP (B_BRANCHKEY) USING STOGROUP TENG.B;
-----
-- REGION_LOOKUP
-----
DROP TABLE DWHODS.REGION_LOOKUP;
CREATE TABLE DWHODS.REGION_LOOKUP
(
    R_REGIONKEY INTEGER NOT NULL,
    R_NAME CHAR(25),
    R_COMMENT VARCHAR(152),
    R_LAST_UPDATE TIMESTAMP NOT NULL WITH DEFAULT,
    PRIMARY KEY ( R_REGIONKEY)
)
IN DBTENG.B.TSREGION ;

CREATE UNIQUE INDEX DWHODS.IX_REGION_LOOKUP
    ON REGION_LOOKUP (R_REGIONKEY) USING STOGROUP TENG.B;
-----
-- LINEITEM
-----
DROP TABLE DWHODS.LINEITEM;
CREATE TABLE DWHODS.LINEITEM
(
    L_ORDERKEY INTEGER NOT NULL,
    L_PARTKEY INTEGER,
    L_SUPPKEY INTEGER,
    L_LINENUMBER INTEGER NOT NULL,
    L_QUANTITY DECIMAL(12,2),
    L_EXTENDEDPRICE DECIMAL(12,2),
    L_DISCOUNT DECIMAL(12,2),
    L_TAX DECIMAL(12,2),
    L_RETURNFLAG CHAR(1),
    L_LINESTATUS CHAR(1),
    L_SHIPDATE DATE,
    L_COMMITDATE DATE,
    L_RECEIPTDATE DATE,
    L_SHIPINSTRUCT CHAR(25),
    L_SHIPMODE CHAR(10),
    L_COMMENT VARCHAR(44),
    L_LAST_UPDATE TIMESTAMP NOT NULL WITH DEFAULT,
    L_SOURCEKEY CHAR(1) NOT NULL,
    PRIMARY KEY(L_ORDERKEY,L_LINENUMBER,L_SOURCEKEY)
) IN DBTENG.B.TSLINEIT ;

CREATE UNIQUE INDEX DWHODS.IX_LINEITEM
    ON LINEITEM(L_ORDERKEY) USING STOGROUP TENG.B;
-----
-- STATUS_LOOKUP
-----

```

```

DROP TABLE DWHODS.STATUS_LOOKUP;
CREATE TABLE DWHODS.STATUS_LOOKUP
  (ST_STATUSKEY CHAR(1) NOT NULL,
   ST_STATUS_DESC VARCHAR(40),
   ST_LAST_UPDATE TIMESTAMP NOT NULL WITH DEFAULT,
   PRIMARY KEY (ST_STATUSKEY)
  ) IN DBTENG.TSSTATLK;

CREATE UNIQUE INDEX DWHODS.IX_STATUS_LOOKUP
  ON STATUS_LOOKUP (ST_STATUSKEY) USING STOGROUP TENG;
-----
-- SUPPLIER
-----
DROP TABLE DWHODS.SUPPLIER;
CREATE TABLE DWHODS.SUPPLIER
  (
   S_SUPPKEY INTEGER NOT NULL CONSTRAINT PRI PRIMARY KEY,
   S_NAME CHAR(25),
   S_ADDRESS VARCHAR(40),
   S_NATIONKEY INTEGER,
   S_PHONE CHAR(15),
   S_ACCTBAL DECIMAL(12,2),
   S_COMMENT VARCHAR(101),
   S_LAST_UPDATE TIMESTAMP NOT NULL WITH DEFAULT
  )
  IN DBTENG.TSSUPPLI ;

CREATE UNIQUE INDEX DWHODS.IX_SUPPLIER
  ON SUPPLIER (S_SUPPKEY) USING STOGROUP TENG;
-----
-- PART
-----
DROP TABLE DWHODS.PART;
CREATE TABLE DWHODS.PART
  (
   P_PARTKEY INTEGER NOT NULL CONSTRAINT PRI PRIMARY KEY,
   P_NAME VARCHAR(55),
   P_MFGR CHAR(25),
   P_BRAND CHAR(10),
   P_TYPE VARCHAR(25),
   P_SIZE INTEGER,
   P_CONTAINER CHAR(10),
   P_RETAILPRICE DECIMAL(12,2),
   P_COMMENT VARCHAR(23),
   P_LAST_UPDATE TIMESTAMP NOT NULL WITH DEFAULT
  )
  PARTITION BY RANGE (P_PARTKEY)
  (PARTITION 1 ENDING AT (1250),
   PARTITION 2 ENDING AT (2500),
   PARTITION 3 ENDING AT (3750),
   PARTITION 3 ENDING AT (3750),
   PARTITION 4 ENDING AT (4000),
   PARTITION 5 ENDING AT (5250),
   PARTITION 6 ENDING AT (6500),
   PARTITION 7 ENDING AT (7750),
   PARTITION 8 ENDING AT (8000),
   PARTITION 9 ENDING AT (9250),
   PARTITION 10 ENDING AT (10500))
  IN DBTENG.TSPART ;

```



```

CREATE UNIQUE INDEX DWHODS.IX_PARTKEY
  ON PART (P_PARTKEY) USING STOGROUP TENGB;
-----
-- PART SUPPLIER
-----
DROP TABLE DWHODS.PARTSUPP;
CREATE TABLE DWHODS.PARTSUPP
(
  PS_PARTKEY INTEGER NOT NULL,
  PS_SUPPKEY INTEGER NOT NULL,
  PS_AVAILQTY INTEGER,
  PS_SUPPLYCOST DECIMAL(12,2),
  PS_COMMENT VARCHAR(199),
  PS_LAST_UPDATE TIMESTAMP NOT NULL WITH DEFAULT,
  PRIMARY KEY(PS_PARTKEY,PS_SUPPKEY)
)
IN DBTENGB.TSPARTSU ;

CREATE UNIQUE INDEX DWHODS.IX_PARTSUPP
  ON PARTSUPP (PS_PARTKEY,PS_SUPPKEY) USING STOGROUP TENGB;

```

---

Example B-3 shows the data definition file that we use to create the DDS of the data warehouse data model in DB2 z/OS that we describe in Chapter 6, “The system environment” on page 91.

---

*Example: B-3 DDS schema definitions*

---

```

SET CURRENT_SCHEMA = DWHDDS;
-----
-- ORDER STATUS DIMENSION
-----
CREATE TABLE DWHDDS.ORDER_STATUS_DIM
(STATUSKEY_DW INTEGER NOT NULL,
  ORDER_RETURNFLAG CHAR(1),
  ORDER_LINESTATUS CHAR(1),
  ORDER_LINESTATUS_DESC VARCHAR(15),
  ORDER_SHIP_PRIORITY INTEGER,
  ORDER_SHIPMODE CHAR(10),
  ORDER_SHIPINSTRUCT CHAR(23),
  ORDER_PRIORITY CHAR(11),
  ORDER_STATUS CHAR(1),
  ORDER_STATUS_DESC VARCHAR(15),
  LAST_UPDATE TIMESTAMP NOT NULL WITH DEFAULT,
  PRIMARY KEY (STATUSKEY_DW)
) IN DBTENGB.TSORSTDM;

CREATE UNIQUE INDEX IX_STATUSKEY_DW
  ON ORDER_STATUS_DIM (STATUSKEY_DW) USING STOGROUP TENGB;
-----
-- CUSTOMER DIMENSION
-----
CREATE TABLE DWHDDS.CUSTOMER_DIM
(CUSTKEY_DW INTEGER NOT NULL,
  CUST_NAME VARCHAR(25),
  CUST_ADDRESS VARCHAR(40),
  CUST_PHONE CHAR(15),
  CUST_NATION_NAME VARCHAR(25),
  CUST_REGION_NAME VARCHAR(25),
  CUST_MKTSEGMENT CHAR(10),
  CUST_KEY_SRC INTEGER,

```

```

        SCD_EFFECTIVE_DT TIMESTAMP,
        SCD_END_DT TIMESTAMP,
        SCD_RECENTFLAG CHAR(1),
        LAST_UPDATE TIMESTAMP NOT NULL WITH DEFAULT,
        PRIMARY KEY (CUSTKEY_DW)
    ) IN DBTENG.B.TSCUSTDM;

CREATE UNIQUE INDEX IX_CUSTKEY
    ON CUSTOMER_DIM (CUSTKEY_DW) USING STOGROUP TENGB;
-----
-- BRANCH DIMENSION
-----

CREATE TABLE DWHDDS.BRANCH_DIM
    (BRANCHKEY_DW INTEGER NOT NULL,
     BRANCH_REGION_NAME VARCHAR(25),
     BRANCH_NATION_NAME VARCHAR(25),
     BRANCH_TYPE VARCHAR(15),
     BRANCH_NAME VARCHAR(25),
     BRANCH_KEY_SRC INTEGER,
     LAST_UPDATE TIMESTAMP NOT NULL WITH DEFAULT,
     PRIMARY KEY (BRANCHKEY_DW)
    ) IN DBTENG.B.TSBRNHDM;

CREATE UNIQUE INDEX DWHDDS.IX_BRANCHKEY_DW
    ON BRANCH_DIM (BRANCHKEY_DW) USING STOGROUP TENGB;
-----
-- DATE DIMENSION
-----

CREATE TABLE DWHDDS.DATE_DIM
    (DATEKEY_DW INTEGER NOT NULL,
     DATE_VAL DATE,
     DAY_YEAR_VAL SMALLINT,
     DAY_MONTH_VAL SMALLINT,
     DAY_WEEK_VAL SMALLINT,
     DAY_WEEK_CODE CHAR(3),
     DAY_WEEK_DESC VARCHAR(9),
     WEEKDAY_IND CHAR(1),
     WEEK_VAL SMALLINT,
     WEEK_START_DATE_VAL DATE,
     MONTH_VAL SMALLINT,
     MONTH_CODE CHAR(3),
     MONTH_DESC VARCHAR(9),
     QTR_VAL SMALLINT,
     QTR_CODE CHAR(4),
     YEAR_VAL SMALLINT,
     LAST_UPDATE TIMESTAMP NOT NULL WITH DEFAULT,
     PRIMARY KEY (DATEKEY_DW)
    ) IN DBTENG.B.TSDATEDM;

CREATE UNIQUE INDEX IX_DATEKEY_DW
    ON DATE_DIM (DATEKEY_DW) USING STOGROUP TENGB;

CREATE VIEW DWHDDS.RECEIPTDATE_DIM_VW AS
SELECT DATEKEY_DW AS RECEIPTDATEKEY,
       DATE_VAL AS RECEIPT_DATE_VAL,
       DAY_YEAR_VAL AS RECEIPT_DAY_YEAR_VAL,
       DAY_MONTH_VAL AS RECEIPT_DAY_MONTH_VAL,
       DAY_WEEK_VAL AS RECEIPT_DAY_WEEK_VAL,

```

```

DAY_WEEK_CODE AS RECEIPT_DAY_WEEK_CODE,
DAY_WEEK_DESC AS RECEIPT_DAY_WEEK_DESC,
WEEKDAY_IND AS RECEIPT_WEEKDAY_IND,
WEEK_VAL AS RECEIPT_WEEK_VAL,
WEEK_START_DATE_VAL AS RECEIPT_WEEK_START_DATE_VAL,
MONTH_VAL AS RECEIPT_MONTH_VAL,
MONTH_CODE AS RECEIPT_MONTH_CODE,
MONTH_DESC AS RECEIPT_MONTH_DESC,
QTR_VAL AS RECEIPT_QTR_VAL,
QTR_CODE AS RECEIPT_QTR_CODE,
YEAR_VAL AS RECEIPT_YEAR_VAL,
LAST_UPDATE AS LAST_UPDATE
FROM DATE_DIM;

```

```

CREATE VIEW DWHDDS.ORDERDATE_DIM_VW AS
SELECT DATEKEY_DW AS ORDERDATEKEY,
DATE_VAL AS ORDER_DATE_VAL,
DAY_YEAR_VAL AS ORDER_DAY_YEAR_VAL,
DAY_MONTH_VAL AS ORDER_DAY_MONTH_VAL,
DAY_WEEK_VAL AS ORDER_DAY_WEEK_VAL,
DAY_WEEK_CODE AS ORDER_DAY_WEEK_CODE,
DAY_WEEK_DESC AS ORDER_DAY_WEEK_DESC,
WEEKDAY_IND AS ORDER_WEEKDAY_IND,
WEEK_VAL AS ORDER_WEEK_VAL,
WEEK_START_DATE_VAL AS ORDER_WEEK_START_DATE_VAL,
MONTH_VAL AS ORDER_MONTH_VAL,
MONTH_CODE AS ORDER_MONTH_CODE,
MONTH_DESC AS ORDER_MONTH_DESC,
QTR_VAL AS ORDER_QTR_VAL,
QTR_CODE AS ORDER_QTR_CODE,
YEAR_VAL AS ORDER_YEAR_VAL,
LAST_UPDATE AS LAST_UPDATE
FROM DWHDDS.DATE_DIM;

```

```

CREATE VIEW DWHDDS.SHIPDATE_DIM_VW AS
SELECT DATEKEY_DW AS SHIPDATEKEY,
DATE_VAL AS SHIP_DATE_VAL,
DAY_YEAR_VAL AS SHIP_DAY_YEAR_VAL,
DAY_MONTH_VAL AS SHIP_DAY_MONTH_VAL,
DAY_WEEK_VAL AS SHIP_DAY_WEEK_VAL,
DAY_WEEK_CODE AS SHIP_DAY_WEEK_CODE,
DAY_WEEK_DESC AS SHIP_DAY_WEEK_DESC,
WEEKDAY_IND AS SHIP_WEEKDAY_IND,
WEEK_VAL AS SHIP_WEEK_VAL,
WEEK_START_DATE_VAL AS SHIP_WEEK_START_DATE_VAL,
MONTH_VAL AS SHIP_MONTH_VAL,
MONTH_CODE AS SHIP_MONTH_CODE,
MONTH_DESC AS SHIP_MONTH_DESC,
QTR_VAL AS SHIP_QTR_VAL,
QTR_CODE AS SHIP_QTR_CODE,
YEAR_VAL AS SHIP_YEAR_VAL,
LAST_UPDATE AS LAST_UPDATE
FROM DWHDDS.DATE_DIM;

```

```

CREATE VIEW DWHDDS.COMMITDATE_DIM_VW AS
SELECT DATEKEY_DW AS COMMITDATEKEY,
DATE_VAL AS COMMIT_DATE_VAL,
DAY_YEAR_VAL AS COMMIT_DAY_YEAR_VAL,
DAY_MONTH_VAL AS COMMIT_DAY_MONTH_VAL,
DAY_WEEK_VAL AS COMMIT_DAY_WEEK_VAL,

```

```

DAY_WEEK_CODE AS COMMIT_DAY_WEEK_CODE,
DAY_WEEK_DESC AS COMMIT_DAY_WEEK_DESC,
WEEKDAY_IND AS COMMIT_WEEKDAY_IND,
WEEK_VAL AS COMMIT_WEEK_VAL,
WEEK_START_DATE_VAL AS COMMIT_WEEK_START_DATE_VAL,
MONTH_VAL AS COMMIT_MONTH_VAL,
MONTH_CODE AS COMMIT_MONTH_CODE,
MONTH_DESC AS COMMIT_MONTH_DESC,
QTR_VAL AS COMMIT_QTR_VAL,
QTR_CODE AS COMMIT_QTR_CODE,
YEAR_VAL AS COMMIT_YEAR_VAL,
LAST_UPDATE AS LAST_UPDATE
FROM DATE_DIM;
-----
-- SUPPLIER DIMENSION
-----
CREATE TABLE DWHDDS.SUPPLIER_DIM
(SUPPKEY_DW INTEGER NOT NULL,
SUPP_NAME CHAR(25),
SUPP_ADDRESS VARCHAR(40),
SUPP_REGION_NAME VARCHAR(25),
SUPP_NATION_NAME VARCHAR(25),
SUPP_PHONE CHAR(15),
SUPP_KEY_SRC INTEGER,
LAST_UPDATE TIMESTAMP NOT NULL WITH DEFAULT,
PRIMARY KEY (SUPPKEY_DW)
) IN DBTENG.B.TSSUPPDM;

CREATE UNIQUE INDEX IX_SUPPKEY_DW
ON SUPPLIER_DIM (SUPPKEY_DW) USING STOGROUP TENG.B;
-----
-- PART DIMENSION
-----
CREATE TABLE DWHDDS.PART_DIM
(PARTKEY_DW INTEGER NOT NULL,
PART_NAME VARCHAR(55),
PART_MFGR CHAR(25),
PART_BRAND CHAR(10),
PART_TYPE VARCHAR(25),
PART_SIZE INTEGER,
PART_CONTAINER CHAR(10),
PART_KEY_SRC INTEGER,
LAST_UPDATE TIMESTAMP NOT NULL WITH DEFAULT,
PRIMARY KEY (PARTKEY_DW)
) IN DBTENG.B.TSPARTDM;

CREATE UNIQUE INDEX IX_PARTKEY_DW
ON PART_DIM (PARTKEY_DW) USING STOGROUP TENG.B;
-----
-- ORDER TRANSACTION FACT
-----
CREATE TABLE DWHDDS.ORDER_TRANSACTION_FACT
(ORDERDATEKEY INTEGER NOT NULL,
SHIPDATEKEY INTEGER NOT NULL,
RECEIPTDATEKEY INTEGER NOT NULL,
COMMITDATEKEY INTEGER NOT NULL,
BRANCHKEY_DW INTEGER NOT NULL,
CUSTKEY_DW INTEGER NOT NULL,
STATUSKEY_DW INTEGER NOT NULL,
PARTKEY_DW INTEGER NOT NULL,

```

```

SUPPKEY_DW INTEGER NOT NULL,
SOURCEKEY CHAR(1) NOT NULL,
ORDERKEY INTEGER NOT NULL,
LINENUMBER INTEGER NOT NULL,
QUANTITY DECIMAL(12,2),
QUANTITY_RETURNED DECIMAL(12,2),
RETAILPRICE DECIMAL(12,2),
EXTENDEDPRICE DECIMAL(12,2),
DISCOUNTPRICE DECIMAL(12,2),
TAXPRICE DECIMAL(12,2),
LAST_UPDATE TIMESTAMP NOT NULL WITH DEFAULT,
PRIMARY KEY (ORDERKEY, CUSTKEY_DW, PARTKEY_DW, SUPPKEY_DW,
LINENUMBER)
) IN DBTENG.B.TSTRANFT;
CREATE UNIQUE INDEX IX_TRANSACTION_FACT
ON ORDER_TRANSACTION_FACT(ORDERKEY, CUSTKEY_DW, PARTKEY_DW,
SUPPKEY_DW,LINENUMBER) USING STOGROUP TENG.B;

```

---

Archived

## Additional material

This book refers to additional material that can be downloaded from the Internet as described below.

### C.1 Locating the Web material

The Web material associated with this book is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser at:

<ftp://www.redbooks.ibm.com/redbooks/SG247637>

Alternatively, you can go to the IBM Redbooks Web site at:

[ibm.com/redbooks](http://ibm.com/redbooks)

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG247637.

### C.2 Using the Web material

The additional Web material that accompanies this book includes the following files:

<i>File name</i>	<i>Description</i>
<b>SG247637_OrderGenerationWebApplication.zip</b>	J2EE application for order generation and simulation of a transactional environment as used in 5.4, “The transactional data model” on page 78 and Chapter 12, “An operational business intelligence implementation” on page 295. See the following readme file for instructions.
<b>SG247637_Readme_OrderGenerationWebApplication.txt</b>	Instructions for importing the J2EE application into WebSphere Integration Developer 6.1.

#### **SG247637\_DataStageJobs.zip**

Jobs used for load and incremental update of the data warehouse as used in Chapter 10, “Full load using DataStage” on page 257 and Chapter 11, “Incremental update with DataStage” on page 281.

#### **SG247637\_Cognos8Models.zip**

Compressed documents as used in Chapter 13, “Reporting and analysis with Cognos 8 BI” on page 305.

#### **SG247637\_DDS\_ODS\_DDL.zip**

Schema definitions as listed in Appendix B, “Schema definitions” on page 431 and used in 5.4, “The transactional data model” on page 78 and Chapter 6, “The system environment” on page 91.

### **C.2.1 System requirements for downloading the Web material**

The following system configuration is recommended:

<b>Hard disk space:</b>	2 MB minimum
<b>Operating System:</b>	Windows
<b>Processor:</b>	Intel® 386 or higher
<b>Memory:</b>	16 MB

### **C.2.2 How to use the Web material**

Create a subdirectory (folder) on your workstation, and extract the contents of the Web material compressed file into this folder.



# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

## IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 451.

- ▶ *Best Practices for SAP BI using DB2 9 for z/OS*, SG24-6489
- ▶ *Building the Operational Data Store on DB2 UDB Using IBM Data Replication, WebSphere MQ Family, and DB2 Warehouse Manager*, SG24-6513
- ▶ *DB2 for OS/390 and Data Compression*, SG24-5261
- ▶ *DB2 for z/OS: Data Sharing in a Nutshell*, SG24-7322
- ▶ *Disaster Recovery with DB2 UDB for z/OS*, SG24-6370
- ▶ *Disk Storage Access with DB2 for z/OS*, REDP-4187
- ▶ *From Multiplatform Operational Data to Data Warehousing and Business Intelligence*, SG24-5174
- ▶ *GDPS Family: An Introduction to Concepts and Facilities*, SG24-6374
- ▶ *How does the MIDAW Facility Improve the Performance of FICON Channels Using DB2 and other workloads?*, REDP-4201
- ▶ *IBM DB2 9 for z/OS: New Tools for Query Optimization*, SG24-7421
- ▶ *IBM System Storage DS8000 Series: Architecture and Implementation*, SG24-6786
- ▶ *IBM System z Strengths and Values*, SG24-7333
- ▶ *Index Compression with DB2 9 for z/OS*, REDP-4345
- ▶ *System Programmer's Guide to: Workload Manager*, SG24-6472
- ▶ *WebSphere Information Integrator Q Replication: Fast Track Implementation Scenarios*, SG24-6487
- ▶ *Workload Management for DB2 Data Warehouse*, REDP-3927

## Other publications

These publications are also relevant as further information sources:

- ▶ *DB2 Version 9.1 for z/OS Administration Guide*, SC18-9840
- ▶ *DB2 Version 9.1 for z/OS Application Programming and SQL Guide*, SC18-9841
- ▶ *DB2 Version 9.1 for z/OS Installation Guide*, GC18-9846
- ▶ *DB2 Version 9.1 for z/OS Performance Monitoring and Tuning Guide*, SC18-9851
- ▶ *IBM BatchPipes OS/390 V2R1 BatchPipeWorks User Guide*, SA22-7457
- ▶ *WebSphere DataStage Parallel Job Advanced Developer Guide*, LC18-9892

- ▶ *WebSphere Information Integration Version 9.1 Replication Installation and Customization Guide for z/OS*, SC19-1025.
- ▶ *WebSphere QualityStage User Guide*, SC18-9922
- ▶ *WebSphere QualityStage Tutorial*, SC19-9925
- ▶ Cognos 8 Business Intelligence documents available from the product installation library and Cognos Global Customer Services support Web site at the following address (license required):  
<http://support.cognos.com>
  - *Cognos 8 Framework Manager - User Guide*
  - *Cognos 8 Business Intelligence - Installation and Configuration Guide*
  - *Cognos 8 Business Intelligence IBM Special Edition Installation and Configuration Guide*
  - *Cognos 8 Framework Manager - Guidelines for Modeling Metadata*
  - *Cognos 8 Business Intelligence - Transformer User Guide*
- ▶ Barry Devlin. *Data Warehouse: From Architecture to Implementation*. Addison-Wesley Professional, 1996. ISBN 0201964252.
- ▶ H. S. Gill and P. C. Rao. *The Official Client/Server Computing Guide to Data Warehousing*. Que 1996. ISBN 0789707144.
- ▶ Claudia Imhoff, PH.D., President and Founder of Intelligence Solutions, Inc., and Mike Schroeck, Partner, IBM Global Business Services, “Operational Intelligence: Business Process Management Meets Business Intelligence” presentation at the Information On Demand conference in Las Vegas 14-19 October 2007
- ▶ “Kimball Design Tip #15: Combining SCD Techniques” by Margy Ross  
<http://www.kimballuniversity.com/html/designtipsPDF/DesignTips2000%20KimballDT15CombiningSCD.pdf>

## Online resources

These Web sites are also relevant as further information sources:

- ▶ IBM Information Server Information Center  
<http://publib.boulder.ibm.com/infocenter/iisinfsv/v8r0/index.jsp>
- ▶ Business Intelligence and Performance Management  
<http://www.ibm.com/BI>
- ▶ Parallel Sysplex technology and benefits  
<http://www.ibm.com/systems/z/resiliency/parsys.html>
- ▶ GDPS
  - Geographically Dispersed Parallel Sysplex: the e-business Availability Solution  
<http://www.ibm.com/servers/eserver/zseries/library/whitepapers/gf225114.html>
  - Implementation Services for GDPS  
<http://www.ibm.com/services/us/index.wss/so/its/a1000189>
- ▶ FICON  
<http://www.ibm.com/systems/z/hardware/connectivity/news.html>

- ▶ DB2 Value Unit Edition  
<http://www.ibm.com/software/data/db2/zos/edition-vue.html>
- ▶ System z New Application License Charges  
<http://www.ibm.com/servers/eserver/zseries/swprice/zna1c.html>
- ▶ System z
  - IBM System z9 Business Class  
<http://www.ibm.com/systems/z/hardware/z9bc/features.html>
  - IBM System z9 Enterprise Class  
<http://www.ibm.com/systems/z/hardware/z9ec/features.html>
  - IBM System z10 Enterprise Class  
<http://www.ibm.com/systems/z/hardware/z10ec/features.html>
- ▶ IBM industry models  
<http://www.ibm.com/software/data/ips/products/industrymodels/>
- ▶ WebSphere Classic Federation client on Linux on System z  
<ftp://ftp.software.ibm.com/software/data/integration/iicf/>
- ▶ Cognos Global Customer Services Web site (for documentation such as Cognos Proven Practice and Guideline white papers)  
<http://support.cognos.com>
- ▶ Cognos 8 BI for System z  
<http://www.ibm.com/software/data/info/new-systemz-software/>
- ▶ Corporate Information Factory  
<http://www.inmoncif.com/home/>
- ▶ Kimball Group  
<http://www.rkimball.com/>

## How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

[ibm.com/redbooks](http://ibm.com/redbooks)

## Help from IBM

IBM Support and downloads

[ibm.com/support](http://ibm.com/support)

IBM Global Services

[ibm.com/services](http://ibm.com/services)

Archived

# Index

## Numerics

3NF (third normal form) 347  
64-bit DB2 246

## A

access path 50, 114, 131, 134  
ad hoc query 19, 71, 374  
ad hoc report 8  
ad hoc users 23  
ADABAS 53  
ADD CLONE 143  
ADD CLONE SQL command 143  
administration 87  
aggregation 12, 61, 153, 342  
AIX 93  
allegiance, implicit or multiple 43  
AlphaBlox 411  
    functions 411  
    small sample Web application 412  
ALTER TABLE  
    EXAMPLE.CUSTOMER 145  
    EXAMPLE.LINEITEM 145  
    EXAMPLE.ORDERS 145  
    SQL statement 50  
ALTER TABLESPACE  
    COMPRESS.LINE Item 119  
    SQL statement 119  
    statement 152  
analysis  
    index compression 108  
    tasks 71  
    techniques 13  
Analysis Studio 305–306, 385  
    further use 365  
    stand-alone calculations 349  
Apache Derby database 314  
APARs 207, 263  
application layer, BI 4  
application server 229, 237  
    new instance 237  
Application Server ID 231  
architecture, scenario 92  
ARM (automatic restart manager) 34–35, 65  
Asynchronous PPRC 38  
attribute-level transformation 15  
automatic restart manager (ARM) 34–35, 65  
automation packages 34  
availability  
    BI solution 32  
    data warehouse 295

## B

backup

    ETL subsystem 7  
    policy 122  
BACKUP SYSTEM utility 37, 40  
    enhancement 42  
Barry Devlin 5  
base table 127  
    statistical information 145  
batch 273  
    massive updates 121  
    update 283  
    window 283  
    window, reduction 283  
batch pipe, opened 209  
BatchPipes 93, 206, 263, 273  
    DB2 LOAD utility 207  
    parallel jobs 263  
    setup 206  
    subsystem 202  
    subsystem file 207  
    support APARs 209, 263  
    troubleshooting 208  
BI (business intelligence) 3, 7–8, 17, 31–32, 75, 269,  
305, 364, 397–398  
    analysis 63  
    application cost 45  
    application layer 4  
    architecture 7, 67  
    modeling tools 308  
    operational 9  
    solution 32, 60, 400  
        data warehouse 32  
    solution architecture 59  
    strategic or informational 9  
    tactical or operational 9  
    trends 20  
bidirectional replication 160  
Bill Inmon 5  
Blox components 411  
Boolean term predicate 126  
BRANCH.BRANCHKEY\_DW 126  
buffer pool 36, 94, 101, 119  
    compressed index 101  
    needed pages 110  
    reduced contention 138  
    single page 104  
BUFFERPOOL BP2 107  
bulk load 87  
business  
    challenges 18, 36  
    metadata 15  
    process 19, 48, 160, 280, 411  
    processes 18  
    questions 6, 26, 282, 398  
    reports 23  
    requirement 17, 24, 32, 59, 68, 72, 283, 289, 350

- for a data warehouse environment 60
- scenario 60, 71, 257–258, 283–284
- users 17, 60, 281, 295, 398
- business intelligence (BI) 3, 7–8, 17, 31–32, 75, 269, 305, 364, 397–398
  - analysis 63
  - application cost 45
  - application layer 4
  - architecture 7, 67
  - modeling tools 308
  - operational 9
  - solution 32, 60, 400
    - data warehouse 32
  - solution architecture 59
  - strategic or informational 9
  - tactical or operational 9
  - trends 20

## C

- C\_CUSTKEY ASC 107
- CAC.SUPPLIER\_SEQ 271–272
- CAC00102I Controller 214
- call center 32, 296
- canned report 8
- cardinality 127
- catalog database
  - dw911 248
  - dw912 248
- central data warehouse 284
- Change Capture Agent 54, 61
- CICS 53, 211
- Classic Data Architect 53–54, 210, 212
  - flat file 215
- Classic Federation 62, 201
  - Data server 215
  - relational engine 210
  - sequential files 215
  - Server 72, 210
- Claudia Imhoff 9
- CLI client 212
- client tier 229
- clients (ODBC, JDBC, and CLI) 212
- clone table 50, 143
- cogconfig tool 318–320
- Cognos 8 BI 73, 306, 368
  - Analysis Studio 306
  - components 306
  - ports 310
  - Report Studio 306
  - reporting 306
- Cognos 8 BI Transformer 335
- Cognos 8 Framework Manager 335
- Cognos Administration 371
- Cognos Connection 335
- column mapping 180
- columns 100, 161, 212, 262, 283, 306, 400
- COMMAND DDNAME 166
- complex queries 94
- concurrency
  - considerations 284, 292

- issues 143
- level 95
- connection handler 210
- consistency of data 26
- consolidated view 261
  - of data sources 85
- continuous temporal model 11
- control interval 94
- COPY 122–123
- COPY NO attribute 41
- Copy Services 37
- COPY YES attribute 41
- copybook 216
- Corporate performance management (CPM) 18
- corresponding key value 87
- cost pressure 18
- coupling facility (CF) 33
  - structures 41
- CPU consumption 100
  - possible increase 100
- CPU cost estimate 304
- cumulative snapshot 11
  - with rolling summarization 11
- current data 5, 18, 146, 162, 281, 358
- customer expectations 18
- customer service 282

## D

- dashboards 307
- data
  - access 19, 32, 60, 202, 210, 299
  - aggregation 347
  - analysis techniques 13
  - caching for star schema queries 138
  - characteristics 7, 20, 51, 149, 161
  - cleansing 15
  - compression 36, 100
  - connector 210–211
  - consolidation 56
  - currency 342, 364
  - current 5, 18, 146, 162, 281, 358
  - elements 4, 7, 288
  - filter 298
  - formats 20, 56, 60, 410
  - frequency 7
  - immediate availability 138
  - integration 5, 55, 60, 226, 279
  - integrity 24, 38, 129
  - modeling options 88
  - moving 20, 32, 260, 284
  - non-relational 78
  - operational 3, 18, 71, 73, 79, 143, 159, 212, 257, 282, 295, 306
  - processing 33
  - quality 21, 56, 257, 280, 293, 295
  - real-time 67, 75, 160
  - requirements 20, 31, 60, 95, 100, 162, 258, 267, 337
  - source types 53
  - striping 121
  - structures 54, 78, 258, 271, 386

- subject area 77
- transfer 208
- transformation 15, 53, 56, 62, 162
- types 4, 24, 32, 105, 180, 272
- volatile 48
- Data Definition Language (DDL) 80, 123
  - script 89, 212
  - statement 81
- data element mapping 6
- data mart 3–6, 20, 55, 67, 77, 267, 281, 284, 289, 307
  - multidimensional analysis 6
- data model 6, 56, 71, 76–78, 88, 109, 126, 258, 335, 337
  - dimensional 85
  - dimensions 129
  - enterprise 71
  - multidimensional 11
  - operational 85
  - relationships 56
  - star schema representation 129
- data server 53–54, 210, 213
  - classic data architect 215
  - main service 210
  - query processor task 228
  - relational structure 53
  - sequential table 216
- data set 143, 273
- data sharing 33–34
  - group 34
- data source 36, 53–54, 60, 62, 72, 78, 206, 211, 258, 271, 295, 299–300, 337, 340, 364, 371, 398
  - consolidated view 85
  - DB2 53, 60, 68, 299, 410
  - internal 412
  - legacy data 81
  - multiple 56, 60, 299, 337, 398
  - non-DB2 68
  - operational 27, 85
  - queues 283
  - relational 53, 385, 410
- data warehouse 3, 6, 18–19, 31–32, 51, 60–61, 67, 71, 73, 75, 78, 99–100, 123, 159–160, 162, 201, 212, 257–258, 260, 281, 295, 307, 397, 431, 437, 448
  - availability 295
  - characteristics 59
  - data mart part 3
  - data modeling styles 10
  - database 60, 67, 87, 92, 95
    - derived information 63
    - initial load 61
    - load data 61
    - real-time data 95
    - resource 67
  - environment 4–5, 31, 37, 42, 60, 94, 118, 148, 261
    - business requirements 60
    - growing need 49
  - function 6
  - functionality 71
  - growing size 32
  - historical data 31
  - load 279
  - queries 300
  - referential integrity 87
  - solution 32, 48
    - challenges 32
  - T layer 5
  - type query 36
  - updates 283
- database
  - design 124
  - recovery log 160
  - referential integrity 87
- database-centric view 400
- DataQuant 397–398
  - big advantage 409
  - multiplatform support 398
  - rich security infrastructure 400
- DataQuant for z/OS 398
- DataStage 53, 61, 63, 75, 88, 92, 162, 168, 189, 202, 206–207, 257, 281, 341
  - engine tier 229
  - interaction with DB2 for z/OS 208
  - job 207–208, 257, 265, 279, 284–285
  - links 260
  - parallel job 260
  - performance analysis feature 276
  - repository tier 229
- DB2 1, 20, 31, 60–61, 71, 92–93, 99, 160, 257, 283, 296, 318, 397, 425–426, 432, 437
  - 64-bit 246
  - buffer pools 102
  - data compression 100
  - data sharing 33–34
  - data source 53, 60, 68, 299, 410
  - functionality 74
  - instance 231
    - fenced user 231
  - LOAD utility with BatchPipes 207
  - log file 55
  - member 63
    - data sharing group 63
    - non-DB2 data source 68
  - optimizer 136
  - replication environment 169
  - tools 51, 169
  - utilities to maintain index structures 45
- DB2 9 32, 42, 45, 62, 95, 99–100, 209
  - data 62
  - exploitation of zIIP 96
  - new features 137
  - New Function Mode subsystem 174
  - pureXML 51
  - SQL language 107
  - use 100
- DB2 9 for z/OS
  - BACKUP and RESTORE SYSTEM utilities
    - enhancement 42
  - software license cost 97
- DB2 Administration
  - Menu 7.2.0 146
  - Server 170

- DB2 AlphaBlox
  - Admin Page 413
  - home page 412
- DB2 Connect 247
- DB2 Control Center 146, 169
- DB2 for Linux, UNIX, and Windows 162
  - configuration settings 319
  - content store preparation 318
  - Enterprise Server Edition 309
  - relational tables 162
  - version check 310
- DB2 for z/OS
  - BACKUP and RESTORE SYSTEM utilities 37, 40
  - data sharing 35, 68
  - data warehouse environment 75
  - functions for a data warehouse 99
  - interaction with DataStage 208
  - subsystem for accessing and loading data 201
  - V8 45
  - WebSphere Classic Federation 201
- DB2 optimizer 50
- DB2 QMF for TSO/CICS 409
- DB2 QMF for WebSphere 410
- DB2 QMF High Performance Option for TSO/CICS 410
- DB2 Replication Center 169
  - event publishing 183
- DB2 subsystem 34, 94, 105, 164, 248, 283
  - single user 114
- DB2 Universal Database 202, 240
  - 9.5 235
  - instance 235
- DB2z stage 261–262, 290
- DD DISP 164, 213
- DD DSN 166
- DD name 219
- DDF workload 97
- DDL (Data Definition Language) 80, 123
  - script 89, 212
  - statement 81
- DDS (dimensional data store) 11, 77, 259, 284
  - table 262
- ddtestlib tool 250
- debugging load jobs 274
- delimited format 162
- DENSE\_RANK 154
- departmental data warehouse 6, 14
- Devlin, Barry 5
- dimension 12
  - hierarchy 12
  - table 124, 259, 289
    - corresponding key value 87
    - history rows 290
    - primary key 87
    - target column names 290
- dimensional data model 85
- dimensional data store (DDS) 11, 77, 259, 284
  - table 262
- dimensional model 14
- dimensional star schema model 77
- disaster recovery 37

- disk partitioning utilities 242
- disk space requirement 100
- disk storage
  - saving 103
  - subsystem 39
- dispatcher port 310
- Distributed Relational Database Architecture (DRDA) 45
- documentation tier 230
- drag-and-drop data analysis 399
- DRDA (Distributed Relational Database Architecture) 45
- DSN prefix 273
- DSNT408I SQLCODE 107
- DSS 4
- dssh tool 250
- dump-to-tape utility 42
- DWHODS.LINE Item 267, 368
- DWHODS.ORDER 153, 265, 368
- dynamic index ANDing 49, 137
- dynamic metadata 15
- dynamic PAV 43–44
- dynamic warehouse 19
- dynamic warehousing 19, 29
  - requirements 32

## E

- EBCDIC 123, 208, 273
- EDM pool 62
- employee productivity 18
- end date 289, 292
- engine tier 229
- enterprise data warehouse 4, 6
- Enterprise Storage Server (ESS) 37
- environment, heterogeneous 25
- ESS (Enterprise Storage Server) 37
- ETL (extract, transform, and load) 4, 51, 53, 62, 257
  - component 63, 201
  - job 57, 201, 258–259, 271, 273
    - parallel execution 254
  - process 51, 62
    - data warehouse 62
  - process expensive component 51
  - subsystem 7
  - surrogate keys 88
  - tool 63, 88, 257
    - data check 88
- ETL task 14
- event 11
- Event Publisher 75, 92, 202, 261, 281
  - implementation 162
- event publishing 54–55, 159, 192
  - DB2 Replication Center 183
  - Q Capture program 163
  - queue map 185
  - replication 162
  - schematic representation 162
  - source tables 187
  - table rows 188
- EXAMPLE.CUSTOMER 145
- EXAMPLE.CUSTOMER\_CLONE 145
- EXAMPLE.LINEITEM 145



- EXAMPLE.LINEITEM\_CLONE 145
- EXAMPLE.ORDERS 145
- EXAMPLE.ORDERS\_CLONE 145
- EXCHANGE DATA
  - operation 146–147
  - SQL command execution 143
- existing information, use of 18
- Extended Remote Copy (XRC) 38
- extract, transform, and load (ETL) 4, 51, 53, 62, 257–258
  - component 63, 201
  - job 57, 201, 258–259, 271, 273
  - parallel execution 254
  - process 51, 62
    - data warehouse database 62
    - expensive component 51
  - surrogate keys 88
  - tool 63, 88, 257
    - data check 88

## F

- fact 11–12
- fact table 77, 124, 259, 261, 269, 303
  - available indexes 128
  - dimension key column 131
  - foreign key 87
  - foreign key column 87
  - key column 131
  - key columns 128
  - legitimate relationship 87
  - LOAD 269–270
  - matching index scan 136
  - order\_transaction\_fact table 269
  - single multicolumn index 125
- FACT.BRANCHKEY\_DW 126
- FACT.QUANTITY 126
- failure 34, 65, 137, 164
- Fast Log Apply 41
- FCP (Fibre Channel Protocol) 42
- federated system 5
- Fibre Channel Protocol (FCP) 42
- FICON Express4 42
- FlashCopy 37
  - incremental 42
- flat files 53, 62, 78, 201, 258, 271
- foreign key column 87
- Framework Manager 335, 337
- FTP command 207
- full load description 261

## G

- GDPS (Geographically Dispersed Parallel Sysplex) 37
- GDPS/PPRC 39
- GDPS/PPRC HyperSwap 37, 40
- GDPS/XRC 38
- Geographically Dispersed Parallel Sysplex (GDPS) 37
- given time period, criteria 26
- Global Copy 38
- Global Mirror 38
- grain 12

- granularity 12, 180, 266, 269

## H

- hardware 33, 68, 94, 101, 203, 283
- high availability 24, 60, 64
- high data volumes 60, 68
- HiperSocket connection 203
- history 26, 73, 148, 178, 259, 282, 406
- HyperSwap function 40

## I

- I/O 36, 94, 96, 102, 264
  - device 263
- IBM Industry Models 88
- IBM QMF 409
- IBM WebSphere Classic
  - Federation Server 210
  - tool 226
- ICF catalog 40
- ICOPY status 122
- identity columns 80
- IFL (Integrated Facility for Linux) 46, 62, 65
- Imhoff, Claudia 9
- implicit allegiance 43
- IMS 20, 53, 60, 62, 81, 211
- incremental FlashCopy 42
- incremental load description 261
- incremental update 212
- index 77, 168, 284, 302, 348, 417
  - design for star schema processing 128
  - skipping 137
  - use tracking by using real-time statistics 121
- index compression 50, 77, 100
  - analysis 108
  - CPU impact 113
  - good candidate 100
  - jobs 425
- index on expression 50, 139
- index-only access query 100
- Information On Demand 25
  - dynamic warehousing 29
- Information Server 56, 62–63, 71, 93, 201–202, 233
  - DataQuality component 56
  - DataStage component 56
  - ddtestlib tool 250
  - dssh tool 250
  - installation 228
  - installation directory 244
  - user ID and registry considerations 231
- Information Server for System z 56
- informational business intelligence 9
- InfoSphere DataStage administrator 231
- infrastructure 1, 28, 32, 62, 68, 121, 163, 306
- initialization service 211
- Inmon, Bill 5
- installation directory 233
- INTEGER NULLIF 208, 274
- Integrated Facility for Linux (IFL) 46, 62, 65
- integration of rows and columns of data 306

- Intelligent Resource Director 52
- internal data source 412
- Internet 8, 78, 370, 400
- Internet service provider (ISP) 164
- intranet 48
- IP address 205
- ISP (Internet service provider) 164

## J

- Java APIs 399
- JavaServer Pages (JSP) 299, 412
- JCL (job control language) 106, 164, 213
  - sample 109, 166
- JDBC client 212
- job control language (JCL) 106, 164, 213
  - sample 109, 166
- JSP (JavaServer Pages) 299, 412

## K

- key column 125, 285
  - multicolumn index 128
- key expressions 139
- Kimball Group 5, 12

## L

- latency 19, 53, 60, 75, 281, 293
  - movement 261
- layer
  - external 5
  - transformation 5
- leaf page 100
- legacy 54, 60, 62, 78, 212, 271
- level of detail 198
- limited workload management, parallel processing 35
- line item 74–75, 79, 129, 177, 258, 266, 269
- lineitem table 84, 259
- Linux on System z 62, 281
  - BatchPipes 93
  - Cognos 8 BI 75
  - ETL components 202
  - Information Server 201
  - J2EE application 296
  - LPAR 202
  - WebSphere Information Server 260
- LOAD 50, 121, 123, 207, 425
- load description
  - full 261
  - incremental 261
- local queue 165
- locks 85
- LOCKSIZE Row 80
- log 36, 101, 164, 207, 257, 283, 400
  - RBA 145
- log reader 54
- logger service 211
- logging port 310
- lookup 77
  - stages 261

- tables 261
- LPAR 63, 202–203
  - failure 65
  - z/OS 92

## M

- maintenance 19, 209, 338
- manageprofile tool 327
- mapping
  - columns 180
  - user role 415
- market challenges 18
- market drivers 18
- massively parallel processing (MPP) 277
- master data management 24
- materialized query table (MQT) 50, 75, 77, 144, 152, 300
- maximum number 140
- measure 11
- message 19, 63, 162, 227, 250, 257, 263, 339, 370
  - queue 283–284
    - payload column 287
  - XML 166
- metadata 6, 21, 56, 76, 202, 212, 284, 337, 369
  - business 15
  - catalog 212–213
  - dynamic 15
  - repository
    - System z server 230
  - static technical 15
  - technical 15
  - usage 342
- Metro Mirror 37
- Metro/Global Mirror 38
- MIDAW (Modified Indirect Data Address Word) 43
- Mike Schroeck 9
- mixed workload 281
  - environment 32
- model 337
- modeling
  - multidimensional 11
- Modified Indirect Data Address Word (MIDAW) 43
- movement of data 20, 32, 260, 284
- MPP (massively parallel processing) 277
- MQ
  - code 197
  - library 245
  - message 54
- MQSeries 20, 160, 168
  - log 167
  - queue 162
  - Queue Manager 171
  - requirement 168
- MQT (materialized query table) 50, 75, 77, 144, 152, 300
- multicolumn index 130
- multidimensional analysis, data mart 6
- multidimensional data model 11
- multidimensional data modeling
  - snowflake model 12
  - star model 12
- multidimensional modeling 11

- multiple allegiance 43
- multiple data sources 56, 60, 299, 337, 398
- multiple platforms 399
- MXQBCE 127

## N

- naming standards 279
- near real time 75
- near real-time data loads 19
- near real-time ODS 32
- near real-time operational data 19
- near real-time warehouse 53
- nodes 254
  - multiple for parallel execution 254
- non-DB2 data source 68
- non-relational data 78
- not logged table space 121

## O

- O\_ORDERSTATUS column 275
- object-level recovery 42
- ODBC 53, 210, 285
  - client 212
  - data source configuration 228, 248
  - driver 226, 230
  - Enterprise stage 286
- ODS (operational data store) 5–7, 18, 20, 32, 55, 67, 75, 77, 85, 257–259, 283, 335, 367
  - environment 288, 338
  - near real-time 32
  - population 261, 269
  - real time 22, 77
- oodit command 199
- OLAP 4, 8, 21, 50, 74, 77, 139, 154, 335, 346, 363, 398
  - Analysis Studio 306
  - function 154
  - functions 154
  - query editor 398
- OLTP 17, 30–32, 60, 72, 78, 86, 95, 166, 247–248, 257–258, 281, 295
  - data source 259
  - database log 257
  - database model 78
  - database schema 259
  - database table 84
  - model schema 80
  - performance 60, 68
    - data warehouse processing 68
  - system 32, 51, 73, 77, 258
  - transaction 62–63, 95
    - data sharing group 65
    - DB2 resources 64
    - lock contention 64
- OLTP.CUSTOMER 195
- OMEGAMON XE 115, 138
- Open Solaris 47
- operational application 19–20
- operational BI 9, 17, 26, 75
  - application 27, 67

- solution 29
- system 28
- operational data 3, 19, 36, 67, 71, 73, 79, 143, 159, 212, 257–258, 282, 295, 306
  - model 85
  - real-time information 20
  - source 27, 85
- operational data store (ODS) 5–7, 18, 20, 32, 55, 67, 75, 77, 85, 257–259, 283, 335, 367
  - environment 288, 338
  - near real time 32
  - population 261, 269
  - real time 22, 77
- operational data updates 18
- operational efficiency 18
- operational intelligence 26
- operational reporting requirements 21
- operational sources 28
- Optimization Service Center
  - parameter browser 127
  - report 127
- optimizer 95, 114, 125, 128, 303
- Oracle 241
- order key 374
- order processing
  - application 297
  - Web application 75, 296
- order record 79
- ORIG.CUSTOMER 107
- outages unplanned 38

## P

- page size 100
  - buffer pool 110
- pair-wise join 125, 137
- parallel access volume (PAV) 43
- parallel engine 230
- parallel execution of multiple nodes 254
- parallel queries 49
- Parallel Sysplex 33, 62, 68
  - architecture 34
  - capability 51
  - cluster continuous availability benefit 40
  - clustering technology 33
  - GDPS/PPRC 39
  - operational task 38
  - scenario 68
  - technology 33, 51, 450
- parallelism 94, 138, 183, 264
  - I/O 43
  - partitioning 278
  - partitioning and pipeline 278
  - pipeline 277
  - query requests 45
- PART
  - data 81
  - table record 82
- PART SUPPLIER
  - data 83
  - table record 83

- partition 49, 148
  - adding 151
  - by growth 77, 149
  - by range 150
  - parallelism 278
  - rotating 152
- partitioning parallel processing 277
- PAV (parallel access volume) 43
- Peer-to-Peer Remote Copy Extended Distance (PPRC-XD) 38
- peer-to-peer replication 160
- PeopleSoft 67
- performance
  - analysis feature 276
  - bottleneck, MQSeries log 167
  - considerations for DataStage 276
  - impact on OLTP 60
  - improvement with DataStage jobs 257
  - metrics in Report Studio 306
- performance management 18, 27
  - cycle 27
  - framework 27
- pipe 264
- pipeline processing 277
- pivot table 398
- point-in-time (PIT) copy 37
- port number 215, 241
- power users 24
- PowerCube 308, 363, 365
  - new data source 365
- PowerCubes 346, 353
- PPRC (Peer-to-Peer Remote Copy) 37
- PPRC-XD (Peer-to-Peer Remote Copy Extended Distance) 38
- prefix compression 100
- PRIMARY Key 199, 212, 259, 267, 274
- primary key 87
- product layer 229
- production 24, 64, 162
  - report 22
  - sysplex 38
- publisher 24
- publishing 160
- pureXML 50

## Q

- Q Apply program 160
- Q Capture 196
  - control table 284
  - program 55, 160–161, 283
  - recovery logs 283
- Q replication 160
  - implementation 162
  - MQSeries requirements 167
  - Q Capture program 163
  - subscription configuration 171
  - types 160
- QMF 409
  - HPO/Compiler 410
  - HPO/Manager 410
- QMF for Workstation 410
- QualityStage 279
- query 5, 8, 32, 108, 210, 337, 398
  - columns 372
  - index-only access 100
  - loads 261
  - parallelism 49, 150
  - performance 77
  - processor 210–211
  - response time 35, 300
  - table 152–153, 300
- Query Studio
  - ad hoc query 374
- queue
  - anatomy 284
  - data source 283
- quiesce 143

## R

- RANK 154
- RBLP (recover based log point) 40–41
- RCMF (Remote Copy Management Facility) 39
- read and write 257
- real time 32, 73, 75
  - BI 257
  - data 60, 283
  - statistic 144
- real-time data 67, 75, 160
- real-time statistics for index-use tracking 121
- record-level transformation 15
- RECOVER 42
- recover based log point (RBLP) 40–41
- recovery 34, 60, 68, 123, 283
  - extract, transform, and load (ETL) subsystem 7
- recovery point objective (RPO) 39
- recovery time objective (RTO) 39
- Redbooks Web site 451
  - contact us xxvi
- REDDWH.ADMI NQ 166
- REDDWH.REST ARTQ 166
- REDDWH.SPIL LQ 166
- redirect percentage 97
- RedParts Distribution 26, 72, 400
  - JSP 420
  - small AlphaBlox Web application 412
  - small report 409
- redundancy 33, 68
- referential integrity
  - data warehouse 87
  - database 87
  - ETL tool 88
  - sanity checks 88
- region controller 210
- relational data source 53, 385, 410
- relational database 72
- Relational OLAP (ROLAP) 8
- Remote Copy Management Facility (RCMF) 39
- REORG utility 119
  - Compression report 119

- replication 26, 38, 62–63, 75, 159–160
  - control tables 164
  - execution 169
  - subscription 161
  - techniques 14
- Replication Center Launchpad 172
- Report Studio 305
  - data items 372
  - performance metrics 306
  - reports 306
  - SQL query 371
- reporting requirements 338
- reports 8
  - Cognos 8 BI 306
- repository tier 229, 231
- requirement
  - business 72
  - disk space 100
  - dynamic warehousing 32
  - MQSeries for Q replication 167
  - operational reporting 21
  - reporting 338
  - response time 95
  - storage 60
  - strategic reporting 72
- response time 35, 95, 304, 421
  - requirements 95
- RESTORE SYSTEM utility 37, 40–41
  - enhancement 42
- return rate 72, 352
  - % measure 395
  - analysis 392
- REXX code 193
  - JCL sample 197
- risk and compliance 18
- RMF Workload Activity Report 96
- rollback operation 147
- rolling updates 66
- row inserted 274
- ROW\_NUMBER 50, 154, 157
- RPO (recovery point objective) 39
- RTO (recovery time objective) 39
- RUNSTATS utility 141, 144

## S

- SAF Exit 214
- same LPAR 52, 65, 68, 95, 174
- same system 35, 53, 231
  - different DB2 subsystems 35
  - other software 231
  - other software components 231
- SAP 67
- scalability 29, 32–33, 35, 64–65, 67, 149, 306
  - data 33, 35
- SCD (slowly changing dimension) 12
  - stage 257
- SCD Type 2 pattern 289
- SCD\_RECENTFLAG 289
- scenario model 347
- schema, OLTP model 80
- Schroeck, Mike 9
- SELECT Count 146, 251
- selected customer
  - Frank N Q 298
  - life 297
- separate LPARs 67
- sequential file 62–63, 76, 81, 211, 216, 222, 267, 269
  - copybook 216
  - mapping 215
  - sample data 81
- Server Time Protocol (STP) 33
- service level 32, 283
- service time
  - CPU 97
  - IIP 97
- serviceability 67
- service-level agreement (SLA) 143
- service-oriented architecture (SOA) 24, 399
  - capability 400
- services tier 229
- shutdown port 310
- single-phase commit query processor 211
- SJMISSKY 127
- SJTABLES 95, 127
- SLA (service-level agreement) 143
- slowly changing dimension (SCD) 12
  - stage 257
- SMP/E (System Modification Program/Extended) 163, 202, 212, 277
- snowflake model 12
- SOA (service-oriented architecture) 24, 399
  - capability 400
- source data 21, 53, 77, 174, 248, 278, 338, 352, 401
  - authenticity 87
  - Data Quality 21
  - relational aware analysis 338
- source table 160, 283
  - data value 190
  - event publishing change capture 162
  - first column 180
  - second column 180
- SQL command, ADD CLONE 143
- SQL icon 370
- SQL queries 210
- SQL request 198, 211
- SQL restriction 126
- SQL statement 85, 118, 123, 302, 371, 402
  - DSNE620I NUMBER 154
- staging table 283
- standard CP 97
- standards 25, 36, 295, 411
- star join 125
  - SQL restrictions 126
- star model 12
- star schema 49, 86, 124–125, 259
  - access methods 125
  - index design 128
  - model 77
  - previous index design challenges 137
  - processing 126

- queries 138
- STARJOIN 127
- state 11
- static PAV 43
- static technical metadata 15
- STOGROUP TENGB 150
- storage requirements 60
- stored procedures 161, 176, 212
- strategic business intelligence 9
- strategic reporting requirements 72
- subject area 77
- subsystem name 164
- summarization 143
- summary table 77, 121
- SUPPLIER data 82
- SUPPLIER table record 83
- suppliers 79, 154, 258, 271
- surrogate key 261, 267, 290
  - stage 257
- symmetric multiprocessing (SMP) 277
- synchronization 26
- Synchronous Peer-to-Peer Remote Copy (PPRC) 37
- SYS1.DSN.V910.SDSN Load 106, 164
- SYS1.MQM.Q801.CSQL OAD 166
- SYS1.MQM.V600.SCSQ AUTH 166
- SYS1.REP.V910.SASN Load 164
- SYSINDEXSPACESTATS.LASTUSED 121
- SYSPITR CRCR 41
- sysplex 33
- Sysplex Timer 33, 39
- SYSPRINT DD SYSOUT 106
- system
  - exit 211
  - parameters for star schema processing 127
  - updates 261
- System Data Mover (SDM) 38
- System Modification Program/Extended (SMP/E) 163, 202, 212, 277
- System z 31–32, 46, 57, 93, 118, 121
  - architecture for BI solution 59
  - business intelligence architecture 61
  - data warehouse 20, 31–32
  - hardware 33
  - hardware compression feature 50
  - Information Server 56–57, 281
  - Open Solaris 47
  - operation BI for data warehousing 295
  - three-site, multipurpose, replication solution 38
  - Web application 296
- system-level backup 42

## T

- TABLE SPACE
  - COMPRESS.LINE Item 120
  - COMPRESSION Report 120
- table space 94, 108, 119, 179
  - compression 118
  - data sets 149
  - hardware compression 119
  - not logged 121

- operational parameters 179
- page size 151
- page size value 149
- partitioning 49, 148–149
- transactional environment 80
- tables 61–62, 77–78, 121, 160, 211–212, 257, 283, 342
- TABLESPACE statement 149
- tactical BI 9
- target queue 166
- target table 53, 160, 199, 263, 283
  - first column 180
  - Profile settings 178
  - straight insert 283
- TCO (total cost of ownership) 45–46, 97
- TCP/IP 203, 210
  - connection handler service information entry 214
  - distributed requests 45
- technical metadata 15
- third normal form (3NF) 347
- three-site solution 37
  - high availability and disaster recovery 38
  - multipurpose replication 38
- time dimension 354
  - general properties 357
  - time properties 358
- time period 74
- tools
  - data warehouse 4, 56, 292
  - user 23
- Total Charge Amt
  - field 384
  - header 380
- total cost of ownership (TCO) 45–46, 97
- TPC-H benchmark 71, 78, 368
  - model 11
- TPC-H data model 85
- TPC-H queries 367
- traditional BI analysis 26
- transaction 22, 67, 71, 77, 162, 258, 281, 293, 299
- transactional data model 78, 80
- transactional environment
  - OLTP workload 84
  - simulation 84
- transformation 53, 64, 265
- Transformer 335, 353
- Transformer model 352, 355
- transformer stage 265, 267, 284
- two-phase commit query processor 211

## U

- uncompressed data 104
- unidirectional replication 160–161
- UNIQUE INDEX
  - ORIG.CUST OMER\_UQ 109
- universal table space 144, 149
  - different types 149
  - partition by growth 149
  - partition by range 150
- UNIX 411
- UNIX System Service (USS) 198

- profile 163
- upsert mode 285–286
- user ID 178, 198, 228, 231, 272
  - home directory 198
- USER QREP 164
- User Role mapping 415
- user-defined function 139
- USS profile 163

## V

- VARCHAR NULLIF 208
- varying-length row 150
- very large database 49
- viewers 24
- virtual data mart 20
- visual report 398, 400
  - key components 407
- volatile data 48
- VSAM 20, 53, 60, 62, 81, 94, 105, 167, 211
  - data set 143
  - data striping 121
  - file 211

## W

- warehouse catalog 6
- warehouse management subsystem 7
- warehouse business intelligence 5
- WCF.CACDS01.SCAC CONF 213
- Web application 28, 74, 162, 296, 399
- WebSphere Administration 414
- WebSphere Application Server 65, 84, 202, 231, 296, 411
  - existing and compatible installation 237
  - high availability 65
  - new version 237
- WebSphere Classic Data Event Publisher 54
- WebSphere Classic Federation 53, 62–63, 76, 81, 93, 202, 258, 271
  - CLI library 250
  - client 226, 271, 451
  - client configuration 228
  - client installation 226
  - configuration file 246
  - data server 53
  - data source 228, 248
  - dedicated client 226
  - Server 54, 77, 210, 226
  - stage 271–272
- WebSphere DataStage 63, 229, 260
  - administrator 252
- WebSphere Federation Server 281
- WebSphere MQ 54, 63, 161, 202, 210, 241, 245, 283
  - client 245, 247
  - communication protocol 212
  - Connector 284
  - Connector stage 284
  - library 245–246
  - object 165
  - object creation 165

- privilege 163
- queue 63, 163, 283
- queue manager 66
- WebSphere Portal 300
- WebSphere Quality Stage 63
- WLM (Workload Manager) 35, 64–65, 96, 209, 263, 281
  - Dynamic Alias Assignment function 43
- work file 133, 364
  - DB2 caches data 138
- WORK.LINE Item 142
- WORK.LINE ITEM\_X1 141
  - SYSCOLDIST CATALOG UPDATE 141
  - SYSCOLUMNS CATALOG UPDATE 141
  - SYSINDEXES CATALOG UPDATE 141
  - SYSINDEXPART CATALOG UPDATE 141
- WORK.LINE ITEM\_XEXP\_01
  - SYSINDEXES CATALOG UPDATE 141
  - SYSINDEXPART CATALOG UPDATE 141
  - SYSKEYTARGETS CATALOG UPDATE 141
  - SYSKEYTGTDIST CATALOG UPDATE 141
- WORK.ORDR\_TRANSACTION\_FACT 126
- workload 20, 64, 84, 102, 211, 281
  - balancing 64
  - growth 46
  - performance 32
  - priorities 95
- workload management 32, 35, 65–67
- Workload Manager (WLM) 35, 64–65, 96, 209, 263, 281
  - Dynamic Alias Assignment function 43

## X

- XML for Analysis (XMLA) 398
- XML support 50
- XMLA (XML for Analysis) 398
- XRC (Extended Remote Copy) 38

## Z

- z/OS 20, 31, 62, 71, 78, 257, 283, 397
  - data source 299
  - DataQuant 398
  - Global Mirror 38
  - LPAR 92
  - Metro/Global Mirror 38
  - system 33, 170, 203
    - remote DB2 203, 248
  - system architecture 409
  - V8 137
- z/VM 67
- z9 Integrated Information Processor (zIIP) 45, 93
  - utilization 96
- zIIP (z9 Integrated Information Processor) 45, 93
  - utilization 96
- zNALC 46

Archived





**Redbooks**

# Enterprise Data Warehousing with DB2 9 for z/OS

(1.0" spine)

0.875" x 1.498"

460 <-> 788 pages







# Enterprise Data Warehousing with DB2 9 for z/OS

**Understand the evolution of data warehousing**

**Evaluate infrastructure tools on IBM System z and Linux on System z**

**See how the DB2 for z/OS engine is suitable for data warehousing**

Enterprises look more and more to business intelligence (BI) to gain a competitive edge. Today's BI systems incorporate large data warehouses that are consolidated with near real-time operational data stores (ODS) and continuously updated from multiple sources. An increasing number of users in the enterprise want to access the data warehouse with BI applications with real-time needs.

There is a renewed interest in the ability to implement a data warehouse solution on DB2 for z/OS and System z. This is due to the inherent characteristics of security, availability, performance, mixed workload management, and the growing portfolio of data warehousing tools and functions provided by IBM.

In this IBM Redbooks publication, we focus on today's software components on System z and show how you can use them to realize the infrastructure for a full data warehouse solution. By using a retail business scenario loosely based on the TPC-H benchmark, we guide you through the warehouse implementation steps. In addition, we highlight the available methods, techniques, and technologies for the deployment of this solution.

This book provides an opportunity for you to look at satisfying the operational needs of corporate users in addition to the longer term needs. In addition, business decision makers, architects, implementors, DBAs, and data acquisition specialists can use this book to gain a better understanding of how a data warehouse can be designed, implemented, and used.

## INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

### BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**  
[ibm.com/redbooks](http://ibm.com/redbooks)

SG24-7637-00

ISBN 0738431400