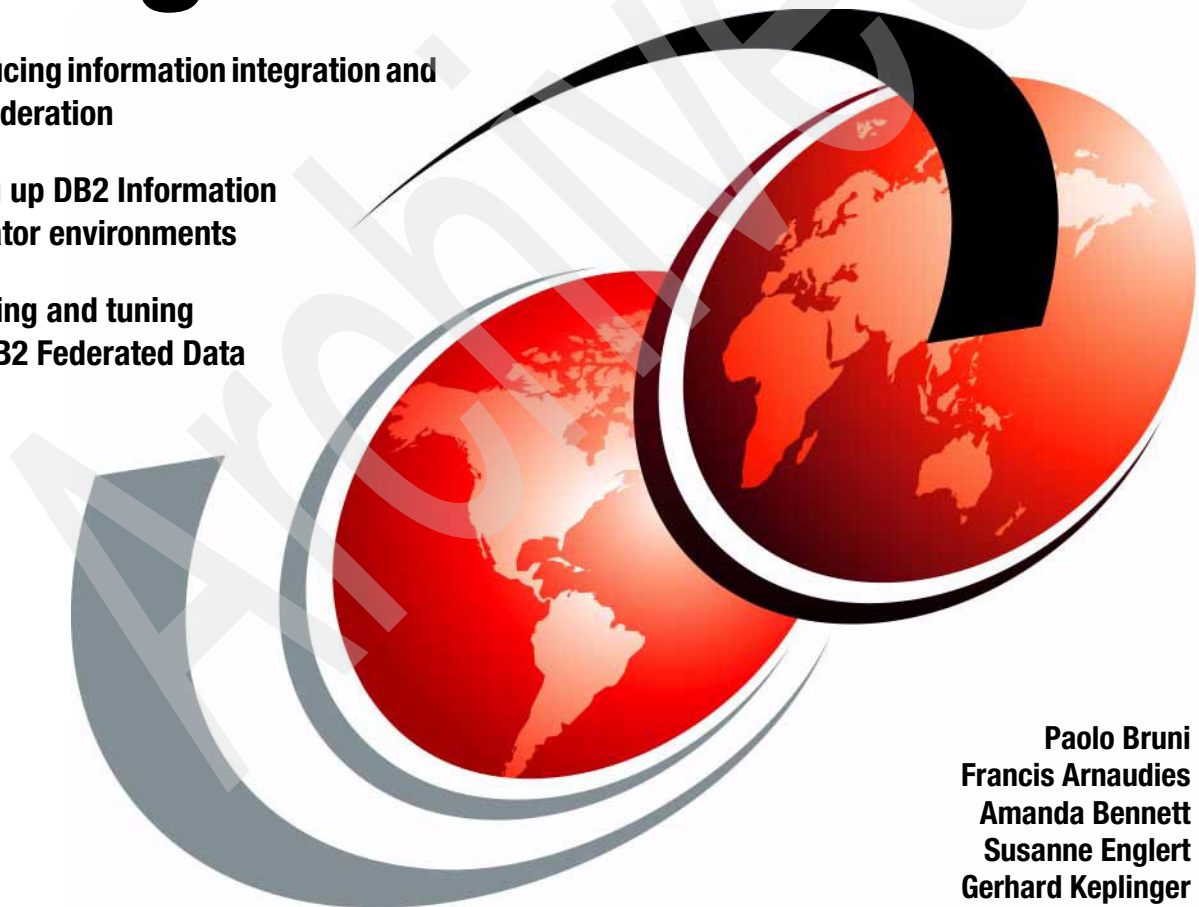


# Data Federation with IBM DB2 Information Integrator V8.1

Introducing information integration and  
data federation

Setting up DB2 Information  
Integrator environments

Operating and tuning  
with DB2 Federated Data



Paolo Bruni  
Francis Arnaudies  
Amanda Bennett  
Susanne Englert  
Gerhard Keplinger





International Technical Support Organization

**Data Federation with IBM DB2 Information  
Integrator V8.1**

October 2003

Archived

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

**First Edition (October 2003)**

This edition applies to Version 8, Release 1 of IBM DB2 Information Integrator (program number 5724-C74).

© Copyright International Business Machines Corporation 2003. 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> .....	xi
<b>Tables</b> .....	xv
<b>Examples</b> .....	xvii
<b>Notices</b> .....	xxi
Trademarks .....	xxii
<b>Preface</b> .....	xxiii
What this redbook is about. ....	xxiii
The contents .....	xxiii
The audience .....	xxv
The team that wrote this redbook. ....	xxvi
Become a published author .....	xxviii
Comments welcome. ....	xxviii
<b>Part 1. Introduction</b> .....	1
<b>Chapter 1. Information integration concepts</b> .....	3
1.1 The business need .....	4
1.1.1 Business challenges .....	4
1.1.2 Business on demand. ....	5
1.1.3 From on demand to grid computing .....	6
1.1.4 From grid to federated data. ....	8
1.2 IBM's information integration. ....	9
1.2.1 Consolidation and federation .....	10
1.2.2 Transparency, transparency .....	11
1.3 The DB2 Information Integrator family of products .....	13
1.3.1 DB2 Information Integrator .....	14
1.3.2 DB2 Information Integrator for Content. ....	15
1.3.3 Extension through partnership .....	16
1.3.4 DB2 Information Integrator and Life Sciences .....	17
1.3.5 The evolution. ....	18
<b>Chapter 2. IBM DB2 Information Integrator V8.1</b> .....	21
2.1 Overview .....	22
2.2 DB2 Information Integrator functions and objects .....	27
2.2.1 Database server .....	29

2.2.2 Federated database .....	29
2.2.3 Configuring the federated system .....	32
2.2.4 Replication .....	45
2.2.5 Wrapper development kit .....	47
<b>Part 2. Configuring the federated data solution .....</b>	<b>51</b>
<b>Chapter 3. The case study .....</b>	<b>53</b>
3.1 The federated system .....	54
3.1.1 The base system .....	54
3.1.2 The DB2 Federated Data .....	55
3.1.3 Information Integrator for relational data sources .....	56
3.1.4 The final picture .....	57
3.2 Setting up the environment .....	58
3.2.1 The starting point .....	58
3.2.2 Connecting DB2 for z/OS and Informix data sources .....	59
3.2.3 Installing Information Integrator .....	60
3.2.4 Connecting Oracle data source .....	61
3.2.5 Connecting Microsoft SQL Server data source .....	62
3.2.6 Connecting XML data source .....	63
3.2.7 Connecting flat file data source .....	64
3.2.8 Connecting Microsoft Excel data source .....	65
3.2.9 System environment: Summary .....	66
3.3 The data and the application .....	68
3.3.1 Data model .....	68
3.3.2 Data distribution .....	70
3.3.3 Data access .....	71
<b>Chapter 4. Configuring DB2 Federated Data support .....</b>	<b>73</b>
4.1 Background information .....	74
4.2 Setting up the DB2 UDB instance on AIX .....	74
4.2.1 Pre-installation requirements .....	75
4.2.2 32-bit or 64-bit DB2 ESE installation and instance .....	76
4.2.3 Installation instructions .....	76
4.2.4 DB2 database configuration for federation .....	87
4.3 Integrating DB2 UDB for z/OS .....	91
4.3.1 Cataloging DB2 for z/OS .....	92
4.3.2 Creating the DB2 for z/OS wrapper .....	93
4.3.3 Creating the DB2 for z/OS server .....	94
4.3.4 Altering DB2 for z/OS server definition and server options .....	98
4.3.5 Creating DB2 for z/OS user mappings .....	99
4.3.6 Altering DB2 for z/OS user mappings .....	101
4.3.7 Creating DB2 for z/OS nicknames .....	102
4.3.8 Altering DB2 for z/OS nicknames .....	105

4.4 Integrating Informix Dynamic Server . . . . .	105
4.4.1 Informix client configuration . . . . .	106
4.4.2 Informix wrapper libraries . . . . .	106
4.4.3 Creating the Informix wrapper . . . . .	108
4.4.4 Creating the Informix server . . . . .	109
4.4.5 Altering Informix server definition and server options . . . . .	112
4.4.6 Creating Informix user mappings . . . . .	113
4.4.7 Altering Informix user mappings . . . . .	115
4.4.8 Creating Informix nicknames . . . . .	115
4.4.9 Altering Informix nicknames . . . . .	118
4.5 Integrating DB2 UDB for iSeries . . . . .	118
<b>Chapter 5. Installing and configuring DB2 Information Integrator . . . . .</b>	<b>121</b>
5.1 General prerequisites . . . . .	122
5.2 Installing DB2 Information Integrator . . . . .	125
5.2.1 Start the installer . . . . .	126
5.2.2 Installation response file examples . . . . .	157
5.2.3 Installation hints, tips, and techniques . . . . .	159
5.3 Applying installed wrappers to instances . . . . .	162
5.4 Integrating Oracle 9i . . . . .	168
5.4.1 Configuration information for Oracle 9i wrapper . . . . .	168
5.4.2 Creating the Oracle wrapper . . . . .	171
5.4.3 Creating the Oracle server . . . . .	172
5.4.4 Altering Oracle server definition and server options . . . . .	176
5.4.5 Creating Oracle user mappings . . . . .	176
5.4.6 Altering Oracle user mappings . . . . .	178
5.4.7 Creating Oracle nicknames . . . . .	179
5.4.8 Altering Oracle nicknames . . . . .	181
5.5 Integrating Microsoft SQL Server 2000 . . . . .	181
5.5.1 Microsoft SQL Server client configuration . . . . .	182
5.5.2 Creating the Microsoft SQL Server wrapper . . . . .	185
5.5.3 Creating the Microsoft SQL Server server . . . . .	186
5.5.4 Altering Microsoft SQL Server definition and server options . . . . .	189
5.5.5 Creating Microsoft SQL Server user mappings . . . . .	190
5.5.6 Altering Microsoft SQL Server user mappings . . . . .	192
5.5.7 Creating Microsoft SQL Server nicknames . . . . .	192
5.5.8 Altering Microsoft SQL Server nicknames . . . . .	195
5.6 Integrating XML data source . . . . .	195
5.6.1 Introduction . . . . .	195
5.6.2 Configuration information for XML wrapper . . . . .	197
5.6.3 Creating the XML wrapper . . . . .	197
5.6.4 Creating the XML server . . . . .	198
5.6.5 Creating XML nicknames . . . . .	199

5.6.6 Altering XML nicknames . . . . .	209
5.7 Integrating table-structured files . . . . .	209
5.7.1 Introduction . . . . .	209
5.7.2 Table-structured file configuration information . . . . .	211
5.7.3 Creating the table-structured file wrapper . . . . .	212
5.7.4 Creating the table-structured file server . . . . .	212
5.7.5 Creating table-structured file nicknames . . . . .	213
5.7.6 Altering table-structured file nicknames . . . . .	217
5.8 Integrating Microsoft Excel . . . . .	218
5.8.1 Introduction . . . . .	218
5.8.2 Configuration information for ODBC wrapper . . . . .	221
5.8.3 Setting Excel ODBC on Windows . . . . .	222
5.8.4 Setting OpenLink client on AIX . . . . .	223
5.8.5 Creating ODBC wrappers . . . . .	223
5.8.6 Creating ODBC servers . . . . .	225
5.8.7 Altering the ODBC server . . . . .	227
5.8.8 Creating the ODBC nickname . . . . .	228
5.8.9 Altering ODBC nicknames . . . . .	232
5.9 Maintaining . . . . .	232
5.9.1 Applying FixPaks . . . . .	232
5.9.2 Updating nickname statistics . . . . .	233
5.9.3 Schema changes at data sources . . . . .	234
5.9.4 Nicknames used in views and packages . . . . .	234
5.10 Troubleshooting . . . . .	235
5.10.1 Errors linking with data source clients . . . . .	235
5.10.2 Errors when defining and using federated objects . . . . .	237
5.10.3 Information to gather . . . . .	238
<b>Part 3. Performance concepts with DB2 Information Integrator . . . . .</b>	<b>239</b>
<b>Chapter 6. A performance tutorial . . . . .</b>	<b>241</b>
6.1 Federated query performance . . . . .	242
6.1.1 Performance factors . . . . .	242
6.1.2 The pushdown concept . . . . .	243
6.1.3 Optimizing a federated query . . . . .	246
6.1.4 Pushdown analysis and cost optimization . . . . .	249
6.1.5 Importance of pushdown . . . . .	250
6.1.6 Interpreting federated query execution plans . . . . .	251
6.2 Tuning a query on a single data source . . . . .	253
6.2.1 What is a good query? . . . . .	253
6.2.2 Evaluating the execution plan with Explain . . . . .	254
6.2.3 Federated server options for best performance . . . . .	260
6.2.4 Analyzing performance of a single remote source query . . . . .	263



6.3 Tuning a query on multiple data sources . . . . .	264
6.3.1 Multiple source queries . . . . .	264
6.3.2 A simple distributed two-source query . . . . .	264
6.3.3 Execution plan for the simple distributed query . . . . .	265
6.3.4 Performance of distributed queries . . . . .	266
6.3.5 Federated queries with nicknames and partitioned tables . . . . .	271
6.3.6 Summary of distributed query performance . . . . .	272
6.4 Performance and availability with MQTs . . . . .	273
6.5 Federated query performance checklist . . . . .	275
6.5.1 Phase 1 . . . . .	276
6.5.2 Phase 2 . . . . .	278
6.5.3 Phase 3 . . . . .	280
<b>Chapter 7. Our performance tools . . . . .</b>	<b>283</b>
7.1 DB2 Explain . . . . .	284
7.1.1 Overview . . . . .	284
7.1.2 DB2 Explain facilities . . . . .	285
7.1.3 Explain tables . . . . .	286
7.1.4 Explaining the queries . . . . .	287
7.2 db2batch . . . . .	288
7.3 Get Statistics utility: get_stats . . . . .	289
<b>Part 4. Exploiting performance options . . . . .</b>	<b>291</b>
<b>Chapter 8. National language support . . . . .</b>	<b>293</b>
8.1 Introduction to code page settings . . . . .	294
8.1.1 Remote relational data source . . . . .	295
8.1.2 DB2 Information Integrator . . . . .	296
8.1.3 DB2 Client . . . . .	297
8.2 Supporting a non-default language . . . . .	299
8.2.1 DB2 Information Integrator setting . . . . .	299
8.2.2 DB2 Client setting . . . . .	299
8.2.3 Conclusion . . . . .	300
8.3 Adding a new language to an existing system . . . . .	300
8.3.1 Remote data source code page definition . . . . .	300
8.3.2 DB2 Information Integrator setting . . . . .	301
8.3.3 DB2 Client setting . . . . .	302
8.3.4 Conclusion . . . . .	303
<b>Chapter 9. Nickname statistics . . . . .</b>	<b>305</b>
9.1 Overview . . . . .	306
9.2 Nicknames with missing statistics . . . . .	308
9.2.1 Check statistics for nicknames . . . . .	308
9.2.2 Create the access plan . . . . .	309

9.2.3 Perform the query .....	310
9.3 Nicknames with statistics .....	311
9.3.1 Option 1: Update stats and recreate nickname .....	311
9.3.2 Option 2: Run get_stats utility .....	312
9.3.3 Check statistics for nicknames again .....	312
9.3.4 Create the access plan .....	313
9.3.5 Perform the query .....	313
9.4 Conclusion .....	313
9.4.1 Comparing the access plans .....	313
9.4.2 Summary .....	315
<b>Chapter 10. Major server options with relational data sources .....</b>	<b>317</b>
10.1 Server option db2_maximal_pushdown .....	318
10.1.1 Overview .....	318
10.1.2 DB2_MAXIMAL_PUSHDOWN set to N (default) .....	319
10.1.3 DB2_MAXIMAL_PUSHDOWN set to Y .....	320
10.1.4 Conclusion on DB2_maximal_pushdown .....	322
10.1.5 DB2_MAXIMAL_PUSHDOWN vs. PUSHDOWN .....	323
10.2 Server option collating_sequence .....	323
10.2.1 Overview .....	323
10.2.2 COLLATING_SEQUENCE set to N (default) .....	324
10.2.3 COLLATING_SEQUENCE set to Y .....	325
10.2.4 Conclusion on collating_sequence .....	327
10.3 Oracle server option varchar_no_trailing_blanks .....	327
10.3.1 Trailing blanks with DB2, Informix, and SQL Server .....	328
10.3.2 Trailing blanks with Oracle .....	329
10.3.3 VARCHAR_NO_TRAILING_BLANKS set to N .....	329
10.3.4 VARCHAR_NO_TRAILING_BLANKS set to Y .....	330
<b>Chapter 11. Using data type mappings .....</b>	<b>333</b>
11.1 Overview .....	334
11.2 Step 1: Explicit cast on nickname column .....	335
11.3 Step 2: Accommodate the default mapping .....	337
11.4 Step 3: Altering local column data type .....	338
<b>Chapter 12. Using function mappings .....</b>	<b>341</b>
12.1 Overview .....	342
12.2 Mapping user defined functions .....	343
<b>Chapter 13. Major server options with non relational data sources. . .</b>	<b>347</b>
13.1 Overview .....	348
13.2 Pushdown with Excel data sources .....	349
13.2.1 PUSHDOWN set to N (default) .....	349
13.2.2 PUSHDOWN set to Y .....	350

13.3 Table-structured files parameter - Sorted .....	351
13.3.1 Nickname parameter SORTED set to N (default) .....	351
13.3.2 Nickname parameter SORTED set to Y .....	352
13.3.3 Conclusion .....	353
<b>Chapter 14. Using index specifications.</b> .....	355
14.1 Overview .....	356
14.2 Table acquires new index after nickname creation .....	357
14.3 Nicknames over remote views .....	357
14.3.1 What about utilizing an existing index? .....	361
14.3.2 Adding index statistics to nicknames for views .....	364
<b>Chapter 15. Using materialized query tables</b> .....	365
15.1 Overview .....	366
15.2 Accessing data without MQT .....	367
15.3 Accessing data with MQT .....	368
15.3.1 Refreshing data in MQTs with nicknames .....	371
<b>Part 5. Appendices</b> .....	373
<b>Appendix A. Case study</b> .....	375
A.1 DDL for DB2 for AIX .....	377
A.2 Script for building the case study .....	382
A.3 Data sources/wrapper/server/schema names .....	388
A.4 DB2 list db directory .....	388
A.5 DB2 list dcs directory .....	389
A.6 DB2 list Node Directory .....	390
A.7 /home/informix/ids92/etc/sqlhosts .....	391
A.8 /oracle9ir2/network/admin/tnsnames.ora .....	391
A.9 /home/db2fed32/odbc.ini .....	391
A.10 /home/db2fed32/sqllib/cfg/db2dj.ini .....	392
A.11 ldb2set -all .....	393
A.12 env .....	393
<b>Appendix B. Unicode tutorial</b> .....	395
B.1 Unicode .....	396
B.2 Locale .....	401
B.2.1 Understanding locale .....	403
<b>Glossary</b> .....	407
<b>Abbreviations and acronyms</b> .....	409
<b>Related publications</b> .....	413
IBM Redbooks .....	413

Other publications ..... 413

Online resources ..... 414

How to get IBM Redbooks ..... 415

Help from IBM ..... 415

**Index** ..... 417

Archived

# Figures

1-1	Business transformation	6
1-2	Grid virtualization	7
1-3	Data federation	9
1-4	Using wrappers	12
1-5	The DB2 Information Integrator products	14
1-6	The Life Sciences integration environment	18
1-7	The history of DB2 federated products	20
2-1	Data federation technology	22
2-2	Federated database with DB2 Information Integrator V8.1	24
2-3	The basic federated components	28
2-4	Replication architecture	47
3-1	DB2 local system	54
3-2	DB2 Federated Data	55
3-3	DB2 Federated Data and relational data sources	56
3-4	DB2 Information Integrator federated data support	57
3-5	Our environment - The starting point	58
3-6	Our environment - Adding DB2 for z/OS and Informix data sources	60
3-7	Our environment - Installing Information Integrator	61
3-8	Our environment - Connecting Oracle data source	62
3-9	Our environment - Connecting Microsoft SQL Server data source	63
3-10	Our environment - Connecting XML files	64
3-11	Our environment - Connecting flat files	65
3-12	Our environment - Connecting Excel files	66
3-13	Project data sources	67
3-14	Data model	69
4-1	DB2 Instance setup - Introduction	77
4-2	DB2 Instance setup - Instance setup	78
4-3	DB2 Instance setup - Instance usage	79
4-4	DB2 Instance setup - Instance owning user	80
4-5	DB2 Instance setup - Fenced user	81
4-6	DB2 Instance setup - Instance TCP/IP	82
4-7	DB2 Instance setup - Instance properties	83
4-8	DB2 Instance setup - DB2 tools catalog	84
4-9	DB2 Instance setup - Monitor notification settings	85
4-10	DB2 Instance setup - Informix data source support settings	86
4-11	DB2 Instance setup - Summary information	87
4-12	DB2 for z/OS - Create wrapper	93
4-13	DB2 for z/OS - Create wrapper dialog	94

4-14	DB2 for z/OS - Create server . . . . .	95
4-15	DB2 for z/OS - Create server dialog . . . . .	96
4-16	DB2 for z/OS - Create server dialog - Settings . . . . .	96
4-17	DB2 for z/OS - Create user mapping . . . . .	99
4-18	DB2 for z/OS - Create user mapping dialog . . . . .	100
4-19	DB2 for z/OS - Create user mapping settings . . . . .	101
4-20	DB2 for z/OS - Create nicknames . . . . .	102
4-21	DB2 for z/OS - Nickname created . . . . .	103
4-22	DB2 for z/OS - Change schema . . . . .	104
4-23	IDS - Create wrapper dialog . . . . .	108
4-24	IDS - Create server dialog . . . . .	109
4-25	IDS - Create server dialog - Settings . . . . .	110
4-26	IDS - Create user mapping dialog . . . . .	114
4-27	IDS - Create user mapping settings . . . . .	114
4-28	IDS - Discover . . . . .	116
4-29	IDS - Change schema . . . . .	117
5-1	DB2 Information Integrator Installation - Launch screen . . . . .	126
5-2	SW license agreement . . . . .	127
5-3	Product selection - Relational, non relational wrappers . . . . .	128
5-4	Source of relational wrapper installation . . . . .	129
5-5	Start the relational wrappers installer . . . . .	130
5-6	Relational wrappers setup . . . . .	131
5-7	Relational wrappers setup - Select products . . . . .	132
5-8	Relational wrappers setup - Introduction . . . . .	133
5-9	Relational wrappers setup - SW License Agreement . . . . .	134
5-10	Relational wrappers setup - Select installation action . . . . .	135
5-11	Relational wrappers setup - Features to install . . . . .	136
5-12	Relational wrappers setup - Languages . . . . .	137
5-13	Relational wrappers setup - Setup DB2 instance . . . . .	138
5-14	Relational wrappers setup - Summary . . . . .	139
5-15	Relational wrappers setup - Installation progress . . . . .	140
5-16	Relational wrappers setup - Complete . . . . .	141
5-17	Relational wrappers setup - Status . . . . .	142
5-18	Source of non relational wrapper installation . . . . .	143
5-19	Start the non relational wrappers installer . . . . .	144
5-20	Non relational wrappers setup . . . . .	145
5-21	Non relational wrappers setup - Select products . . . . .	146
5-22	Non relational wrappers setup - Introduction . . . . .	147
5-23	Non relational wrappers setup - SW License Agreement . . . . .	148
5-24	Non relational wrappers setup - Select install actions . . . . .	149
5-25	Non relational wrappers setup - Features to install . . . . .	150
5-26	Non relational wrappers setup - Languages . . . . .	151
5-27	Non relational wrappers setup - Setup DB2 instance . . . . .	152

5-28	Non relational wrappers setup - Summary . . . . .	153
5-29	Non relational wrappers setup - Installation progress . . . . .	154
5-30	Non relational wrappers setup - Status . . . . .	155
5-31	Non relational wrappers setup - Complete. . . . .	156
5-32	DB2 Information Integrator installation - Complete . . . . .	157
5-33	Configure the wrappers for the DB2 instance db2fed32 . . . . .	163
5-34	Oracle data source support . . . . .	164
5-35	Oracle - Warning . . . . .	165
5-36	Microsoft SQL Server - Data source . . . . .	165
5-37	DB2 - Setup wizard . . . . .	166
5-38	DB2 - Setup complete . . . . .	167
5-39	Oracle - Create Wrapper . . . . .	172
5-40	Oracle - Create Server dialog . . . . .	173
5-41	Oracle - Create Server - Settings. . . . .	174
5-42	Oracle - Create User Mappings . . . . .	177
5-43	Oracle - Create User Mappings - Settings. . . . .	178
5-44	Oracle - Create Nicknames . . . . .	179
5-45	Oracle - Create Nicknames - Change Schema . . . . .	180
5-46	Microsoft SQL Server - Create Wrapper dialog . . . . .	185
5-47	Microsoft SQL Server - Create Server . . . . .	186
5-48	Microsoft SQL Server - Create Server - Settings. . . . .	187
5-49	Microsoft SQL Server - Create user mappings . . . . .	191
5-50	Microsoft SQL Server - Create user mappings - Settings . . . . .	191
5-51	Microsoft SQL Server - Discover . . . . .	193
5-52	Microsoft SQL Server - Create nicknames - Change schema . . . . .	194
5-53	The XML data source. . . . .	196
5-54	XML - Create Wrapper. . . . .	198
5-55	XML - Create Server . . . . .	199
5-56	XML - Create Server - Discover. . . . .	200
5-57	XML - File Browser . . . . .	203
5-58	XML - Discover . . . . .	204
5-59	XML - Change column . . . . .	205
5-60	XML - Properties . . . . .	205
5-61	XML - Sample contents . . . . .	206
5-62	Table-structured file data source . . . . .	210
5-63	Table-structured file - Create Wrapper . . . . .	212
5-64	Table-structured file - Create Server . . . . .	213
5-65	Table-structured file - Create nicknames. . . . .	214
5-66	Table-structured file - Add column . . . . .	215
5-67	Table-structured file - Properties . . . . .	216
5-68	Table-structured file - Select objects for nicknames . . . . .	216
5-69	Control Center - Again nicknames. . . . .	217
5-70	Excel files through OpenLink . . . . .	221

5-71	ODBC - Create wrapper . . . . .	224
5-72	ODBC - Create wrapper - Settings . . . . .	224
5-73	ODBC - Create server . . . . .	225
5-74	ODBC - Create server - Settings . . . . .	226
5-75	ODBC - Add nickname . . . . .	228
5-76	ODBC - Nickname created . . . . .	229
5-77	ODBC - Nickname contents . . . . .	229
5-78	Microsoft Excel - Workbook definition . . . . .	231
6-1	Pushdown example 1 . . . . .	244
6-2	Pushdown example 2 . . . . .	245
6-3	Query compiler flow . . . . .	248
6-4	Federated vs. native - Single query . . . . .	254
6-5	Query 4 Explain plan . . . . .	256
6-6	Query 4 rewrite . . . . .	257
6-7	Query 14 rewrite and Explain . . . . .	258
6-8	Query 20 Explain plan . . . . .	260
6-9	Improving nickname's statistics . . . . .	263
6-10	Distributed query . . . . .	265
6-11	Distributed query - Execution plan . . . . .	266
6-12	Back to the query . . . . .	268
6-13	Comparison results . . . . .	270
6-14	Materialized views . . . . .	274
6-15	Materialized views on nicknames . . . . .	275
6-16	Performance tuning - Phase 1 . . . . .	277
6-17	Performance tuning - Phase 2 . . . . .	279
6-18	Performance tuning - Phase 3 . . . . .	281
8-1	The three layers of code page setting . . . . .	295
8-2	Windows 2000 - Regional options . . . . .	298
8-3	Setting the code page for no conversion . . . . .	300
8-4	The default code page settings . . . . .	301
8-5	The Unicode code page settings . . . . .	302
9-1	Access plan without and with statistics on the nickname . . . . .	314
10-1	Query plan for db2_maximal_pushdown to 'N' . . . . .	320
10-2	Query plan for db2_maximal_pushdown to 'Y' . . . . .	321
10-3	Visual Explain on collating_sequence example . . . . .	327
13-1	Access plan - Pushdown set to N . . . . .	349
13-2	Access plan - Pushdown set to Y . . . . .	350
13-3	SORTED = 'N' versus SORTED = 'Y' . . . . .	353
14-1	Access plan without index specification on nickname . . . . .	360
14-2	Access plan with index specification on nickname . . . . .	363
15-1	MQT - Sample query access plan . . . . .	368
15-2	MQT - Explain . . . . .	370



# Tables

2-1	DB2 Information Integrator packaging	25
2-2	Supported data sources	27
2-3	Global catalog contents for remote data sources	44
2-4	Wrapper building blocks	48
3-1	Database size for each data source	69
3-2	Distribution of tables across the data sources	70
3-3	Nickname schema for data sources	70
4-1	DB2DJ.INI variables - Required and optional	89
4-2	The DB2 for z/OS system	92
4-3	DB2 for z/OS server options	97
4-4	DB2 for z/OS additional server options	99
4-5	The Informix system	106
4-6	Informix server options	111
4-7	Informix additional server options	113
4-8	DB2 for iSeries system	118
5-1	Wrappers and supported operating systems	124
5-2	Oracle information	168
5-3	Oracle server options	174
5-4	Oracle additional server options	176
5-5	Microsoft SQL Server information	182
5-6	Microsoft SQL Server - Server options	188
5-7	Microsoft SQL Server additional server options	190
5-8	XML information	197
5-9	Table-structured file - Information	211
5-10	Excel data type mapping	219
5-11	Excel information	221
5-12	OpenLink Server ODBC driver information	222
5-13	OpenLink Client ODBC driver information	222
5-14	ODBC additional server options	227
7-1	Nickname catalog information updated by get_stats utility	289
8-1	Client settings for relational wrapper	296
9-1	Output from sysstat.tables	309
9-2	Output from sysstat.tables after update	312
A-1	Data source - Wrapper-Server-Schema name - Matrix	388
B-1	Unicode specifications and requirements	397



# Examples

2-1	Creating a wrapper	34
2-2	Creating a server	35
2-3	Creating a nickname	39
4-1	DB2 for z/OS - Create wrapper statements	94
4-2	DB2 for z/OS - Create server statements	98
4-3	DB2 for z/OS - Create user mapping statements	101
4-4	DB2 for z/OS - Create nickname statements	104
4-5	Informix - Create wrapper statements	108
4-6	Informix - Create server statements	112
4-7	Informix - Create user mapping statements	115
4-8	Informix - Create nickname statements	117
5-1	Response file - Installation of relational wrappers	157
5-2	Response file - Installation of non relational wrappers	158
5-3	The tnsnames.ora file	170
5-4	Oracle - Create wrapper statement	172
5-5	Oracle - Create server statement	175
5-6	Oracle - Create user mapping statement	178
5-7	Oracle - Create nickname statements	180
5-8	Entry .odbc.ini for SQL Server	183
5-9	db2set variables	185
5-10	SQL Server - Create wrapper statements	185
5-11	Microsoft SQL Server - Create server statement	189
5-12	Microsoft SQL Server - Create user mapping statement	192
5-13	Microsoft SQL Server - Create nickname statements	194
5-14	XML - Create wrapper statement	198
5-15	XML - Create server statement	199
5-16	XML document	201
5-17	XML - Define element	201
5-18	XML - Create nickname statements	206
5-19	XML - FILE_PATH not defined	207
5-20	XML - Create nickname statement	209
5-21	Table-structured file - Create wrapper statement	212
5-22	Table-structured file - Create server statement	213
5-23	Contents of file 'region.tbl'	213
5-24	Table-structured file - Create nickname statement	217
5-25	Activating OpenLink server	223
5-26	ODBC - Create wrapper statement	225
5-27	ODBC - Create server statement	226

5-28	ODBC - Create nickname statement . . . . .	230
5-29	Microsoft Excel - Create nickname statements . . . . .	232
5-30	Output of db2diag.log. . . . .	237
7-1	Our Explain script. . . . .	287
7-2	db2batch syntax and example . . . . .	288
8-1	Checking your code page . . . . .	297
8-2	Setting the DB2 code page . . . . .	299
8-3	Setting the Oracle code page . . . . .	299
8-4	DB2 registry setting . . . . .	300
8-5	Setting the Unicode code page for DB2. . . . .	302
8-6	Setting the Unicode code page for Oracle. . . . .	302
8-7	DB2 registry setting . . . . .	303
9-1	Revenue for orders . . . . .	307
9-2	Check statistical information . . . . .	308
9-3	Script to create the access plan for a query. . . . .	309
9-4	Perform the query without statistics. . . . .	310
9-5	db2batch output without statistics . . . . .	310
9-6	Recreating nicknames . . . . .	311
9-7	Get_stats utility on Lineitem, Orders, Customer . . . . .	312
9-8	Perform the query with statistics . . . . .	313
9-9	Shipped queries - No statistics on nicknames. . . . .	314
9-10	Shipped queries - With statistics on nicknames. . . . .	315
10-1	Query for db2_maximal_pushdown . . . . .	318
10-2	Query Explain with collating_sequence set to N . . . . .	325
10-3	Add or Modify server option collating_sequence . . . . .	326
10-4	Query Explain with collating_sequence set to Y . . . . .	326
10-5	Query statement . . . . .	327
10-6	Trailing blanks with DB2 . . . . .	328
10-7	Trailing blanks with Oracle. . . . .	329
10-8	Query shipped to Oracle . . . . .	330
10-9	Set server option . . . . .	330
10-10	Query shipped to Oracle . . . . .	330
11-1	Original query. . . . .	334
11-2	Casting the remote column l_shipdate to DATE . . . . .	335
11-3	Query pushed down to Oracle (db2exfmt output) . . . . .	336
11-4	Performance with the explicit cast . . . . .	336
11-5	Using TIMESTAMP to compare column l_shipdate. . . . .	337
11-6	Query pushed down to Oracle . . . . .	337
11-7	Performance accommodating the default mapping . . . . .	338
11-8	Using data type mapping and original query syntax . . . . .	338
11-9	Query pushed down to Oracle (db2exfmt output) . . . . .	339
11-10	Performance with data type mapping . . . . .	339
12-1	Oracle function. . . . .	343

12-2	Test Oracle function . . . . .	344
12-3	Create function mapping . . . . .	345
12-4	Function mapping test . . . . .	346
12-5	Function mapping test - Query pushed down to Oracle. . . . .	346
13-1	Simple query . . . . .	349
13-2	No pushdown - Queries shipped to the server. . . . .	350
13-3	Set PUSHDOWN to Y . . . . .	350
13-4	Pushdown - Query shipped to the server. . . . .	350
13-5	Query used for sorted table-structured file. . . . .	351
13-6	Set parameter sorted to N . . . . .	352
13-7	Performance details. . . . .	352
13-8	Set parameter sorted to Y . . . . .	352
13-9	Performance details. . . . .	352
14-1	Create new index on table part on DB2 DATDB data source . . . . .	357
14-2	Add information about new index on data source table to nickname. . . . .	357
14-3	Create view soldbypart on DB2 database datdb . . . . .	358
14-4	Create nickname for view on DB2 database feddb . . . . .	358
14-5	Information about our new nickname. . . . .	358
14-6	update cardinality for new nickname . . . . .	359
14-7	Get total quantity sold for parts with special container. . . . .	359
14-8	Result of query in Example 14-7 . . . . .	359
14-9	Shipped queries. . . . .	361
14-10	Create index specification for nickname on view. . . . .	362
14-11	Result of query in Example 14-7 . . . . .	362
14-12	Shipped queries. . . . .	363
14-13	Creating a nickname with index statistics . . . . .	364
15-1	MQT - Sample query . . . . .	367
15-2	MQT - Query response time . . . . .	367
15-3	MQT - Create table . . . . .	368
15-4	MQT - Enabling MQT use . . . . .	370
15-5	MQT - Response time with new access plan. . . . .	370
A-1	DB2 data source - Data definition language . . . . .	377
A-2	Case study - Creating federated database objects . . . . .	382
A-3	System Database Directory . . . . .	388
A-4	DCS directory. . . . .	389
A-5	Node Directory. . . . .	390
A-6	etc/sqlhosts . . . . .	391
A-7	tsnames.ora . . . . .	391
A-8	.odbc.ini . . . . .	391
A-9	db2dj.ini . . . . .	392
A-10	db2set -all . . . . .	393
A-11	env . . . . .	393

Archived

# 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 illustrates 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. 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 IBM's application programming interfaces.

# Trademarks

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

developerWorks®	DB2 Connect™	MVS™
iSeries™	DB2 Universal Database™	Notes®
z/OS®	DB2®	OS/390®
zSeries®	DRDA®	QBIC®
Approach®	Enterprise Storage Server®	QMF™
AIX 5L™	Informix®	Redbooks™
AIX®	Intelligent Miner™	RACF®
AS/400®	IBM®	RAMAC®
CICS®	IMS™	RS/6000®
Database 2™	Language Environment®	System/390®
DataJoiner®	Lotus Notes®	WebSphere®
DiscoveryLink®	Lotus®	Redbooks (logo)  ™
Domino®	MQSeries®	IBM  server™

The following terms are trademarks of other companies:

Intel and Intel Inside (logos) are trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

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



# Preface

Federation and integration are open technologies, which extend your investment on existing data and relational or non relational data by bringing it all together in a single virtual location for on demand application access. IBM® DB2® Information Integrator V8.1 is the framework supporting federated data solutions.

This IBM Redbook provides implementation guidelines for deploying a federated access solution with IBM DB2 Information Integrator V8.1. Topics include configuring, tuning, and debugging a federated system; use of federation in a business intelligence-like environment with read access to relational (DB2 family, Oracle, SQL Server) and non relational sources (flat files, XML, ODBC), and usage through queries or reporting tools.

This redbook will help you create a federated data solution with a case study, which accesses several data sources across the AIX®, Windows®, and z/OS® platforms.

## What this redbook is about

In this IBM Redbook we describe the topic of data federation as implemented with the DB2 Information Integrator Version 8.1 product. This book aims to provide database administrators and DBMS support personnel with clear examples on how to implement data federation and gain the benefits available. In order to do so, we have defined a case study and have carried several tests documenting our experience with in-depth examples and detailed screen captures.

By no means do we have the presumption of having covered all the permutations possible when considering all products, platforms, and data sources supported by the DB2 Information Integrator family of products. We have however tried in our project to cover what we considered the most common needs when getting started with federating data across relational and non relational data sources.

## The contents

Let us start with what is not in this redbook, just to make sure that the reader will not waste time.

## What you will not find here

We have not included DB2 Information Integrator for Content, formerly Enterprise Information Portal (EIP). It is a large a specialized topic, which even though it should be considered as a natural candidate for extending your federated system, we think it deserves another dedicated project and redbook. And we have not used the IBM Lotus® Extended Search (previously Domino® Extended Search).

The same for Life Sciences. Because of the very different characteristics of the data, and the several possible data sources, a separate project will be necessary.

Legacy data sources like IMS™ and VSAM have not been included. They are currently supported through vendors' products and we have not used them. Our examples on flat files have been limited to what is currently directly supported, a table like structure with fixed length records. We have not explored the capabilities of developing a home grown wrapper using the Wrapper Developer's Kit.

We have not used Net.Search or created any text related definition and search.

We have not considered the application aspects of federated data, we have ignored patterns, WebSphere®, Web Services, WebSphere MQ, but we hope to in the immediate future.

Finally, we have only mentioned data replication. The reasons for not going into details on this complementary part of an information integration solution are lack of time, and more important, the existence of an excellent, very recent and exhaustive redbook dedicated to this topic: *A Practical Guide to DB2 Data Replication V8*, SG24-6828.

So, what is left? We think there is plenty left. Let us look at contents then!

## What is in the redbook

Basically, we have installed and configured DB2 Information Integrator on AIX. We also have installed a bunch of other DBMS on other platforms. We have taken some coherent and consistent data and loaded it into all data sources. We have then defined all the remote data in our federated system and performed several queries examining their performance characteristics. We have tried to document in a meaningful way most of the setup, investigation, and findings.

Part 1, "Introduction" on page 1 has two chapters. The first one describes the concepts of grid computing, information integration, and federated data. In the second one we briefly describe the main components and functions of DB2 Information Integrator. The intent here is to provide enough background

information to be able to read the following parts without rushing immediately to the other documentation.

Part 2, “Configuring the federated data solution” on page 51 has four chapters. The first one describes our case study, and the other three describes the three steps in implementing our federated data solution. The theme is *gradually increasing complexity* when installing and configuring the environment. In the first step we assume that you will start with an existing DB2 system on AIX, and we activated the native federated data support of the DB2 engine to access other members of the DB2 family. We then added the Information Integrator capabilities for non IBM relational sources such as Oracle and Microsoft® SQL Server on Windows, and as last step, we included XML, Excel, and flat files as non relational data sources.

Part 3, “Performance concepts with DB2 Information Integrator” on page 239 provides a general tutorial on the performance concepts, and the performance tools that are the basis for understanding how your queries are performing in a data federation environment.

Part 4, “Exploiting performance options” on page 291 has several chapters dedicated to showing examples of implementing accesses to the data sources, and providing recommendations on how to improve performance. We look with some detail at nickname statistics, server options for relational and non relational data sources, data type mapping, function mapping, index specifications, and materialized query tables, and show their possible impact on your queries. We also give examples of code page specifications.

## The audience

The anticipated audience of this redbook is the DBA who wishes to understand the concepts and product abilities of IBM's third generation information management product, DB2 Information Integrator. We hope that independent from the business need, which is urging you to integrate your data, you will find enough test cases and detailed examples to quickly start with DB2 Information Integrator's federated data solution.

## The team that wrote this redbook

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



*Figure 1 Left to right: Gerhard, Amanda, Paolo, Susanne, and Francis*

**Paolo Bruni** is a certified Consultant IT Architect working as Data Management Project Leader at the International Technical Support Organization, San Jose Center since 1998. In this capacity he has authored several redbooks on DB2 and DB2 tools, and has conducted workshops and seminars worldwide. During his many years with IBM, in development and in the field, Paolo's work has been mainly related to database systems.

**Francis Arnaudies** is an IT Specialist in Paris with IBM France. He has eight years of experience in the IT Industry, fulfilling a number of positions such as five years as a Developer, to the last three years as a DB2 Technical Pre-sales Specialist. Francis holds a degree in Computer Science awarded from the University of La Rochelle, France. His areas of expertise include DB2 migration from Oracle, Sybase, and Informix®.

**Amanda Bennett** is an Early Support Program Manager, presently project managing DB2's Information Integrator Beta participants for the EMEA Product Introduction Center, Hursley, UK. She has seven years of sales experience in the IT and telecom industries. Amanda joined IBM Data Management Software Business in 2000.

**Susanne Englert** has been a Senior Software Engineer at the IBM Silicon Valley Lab since August 2000, and currently works on DB2 Information Integrator. She has 15 years of experience in software performance measurement and analysis, with particular interest in the performance and optimization of complex queries on large databases. Susanne chaired the Transaction Processing Performance Council's (TPC) Decision Support subcommittee from 1996 to 2000. She holds degrees in mathematics and computer science from the University of California and the University of Bonn.

**Gerhard Keplinger** is an Advisory IT Specialist with IBM Austria involved in DB2 pre-sales and services. Gerhard holds a master's degree in Software Engineering from the Polytechnic University Hagenberg, Austria, and joined IBM in 2001. His areas of expertise include Oracle and Informix Dynamic Server.

We would like to thank the following people for their contributions to this project, with special gratitude to Eileen Lin and Micks Purnell:

Walter Alvey  
Khalid Albarrak  
Aakash Bordia  
Marlene Coates  
Cathy Drummond  
Anjali Grover  
Arthur Kaufmann  
Jacques Labrie  
Eileen Lin  
Gil Lee  
Debbie Mayhew  
Rob Montroy  
Tina Mukai  
Randy Nakagawa  
Michael Ortega-Binderberger  
Mi Shum  
Laura Stewart  
Mark Taylor  
Ioana Ursu  
Ben Wilde  
Kevin Winterfield  
Ron Yorita  
IBM Silicon Valley Lab, San Jose

Nagraj Alur  
Emma Jacobs  
Bart Steegmans

Maritza M. Dubec  
International Technical Support Organization, San Jose Center

Hernando Bedoya  
International Technical Support Organization, Rochester Center

Micks Purnell  
IBM USA

## Become a published author

Join us for a two- to seven-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll 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 Redbooks™ to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

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

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

- ▶ Send your comments in an Internet note to:

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

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization  
Dept. Error Building 80-E2  
650 Harry Road  
San Jose, California 95120-6099



# Part 1

# Introduction

In this introductory part we outline the concepts of grid computing, information integration, and federated data. We then briefly describe the main components and functions of the DB2 Information Integrator and related products.

The chapters in this part are:

- ▶ Information integration concepts
- ▶ IBM DB2 Information Integrator V8.1





# Information integration concepts

In this chapter we briefly describe the needs for information integration, and the main concepts behind the DB2 Information Integrator products. We start with the business needs, involve grid computing, outline the main choices and functions of the information integration area, and then introduce DB2 Information Integrator and related products. The objective of this chapter is to provide business justification and background information for DB2 Information Integrator.

For more details on information integration, please refer to *Getting Started in Integrating Your Information*, SG24-6892, and the several articles included in *IBM Systems Journal Vol. 41, No. 4, 2002*, and *Information Integration*, G321-01473. See the following Web site for recent white papers and other technical documents:

<http://www.ibm.com/software/data/integration/>

## 1.1 The business need

Today's fast evolving business climate enforces increased demands for fast analysis of large amounts and disparate sources of business critical data. Businesses must access not only traditional application sources such as relational databases, but also Extensible Markup Language (XML) documents, text documents, scanned images, video clips, news feeds, Web content, e-mail, analytical cubes, and special-purpose stores. Information integration is a way of providing an end-to-end solution for transparently managing both the volumes and diversity of data. DB2's Information Integrator is the product that can quickly build the framework for such a solution by allowing applications to meet the need to adapt to business change, while maintaining a business infrastructure that keeps up with the demands of the market place.

In this redbook we look at the business challenges and environment, the on demand solution, the grid architecture, information integration, and then describe how the DB2 family of information management products addresses these information integration needs.

### 1.1.1 Business challenges

Information needs to be provided fast and efficiently, and, as breakthroughs in technology emerge, the present demands increase.

With a variety of business deliverables that provide greater visibility of the business needs, both from an internal or external perspective, we have seen a dramatic evolution in the daily functioning of business. Terms such as *Customer Relationship Management*, *Supply Chain Management*, and *Enterprise Resource Planning* have evolved to provide greater understanding of business information assets.

At the same time we see the overall landscape of business models changing and adapting with technology, as the increase of international business partnerships and relationships extend. A business needs to provide constant up to date knowledge on business critical functions.

Information integration can help in solving the challenges related to:

- ▶ Information stored on multiple disparate databases
- ▶ Information stored across different geographical locations
- ▶ The information source may come in a variety of structured or non structured formats.
- ▶ Replication or consolidation is not always a viable option, given possible technology and time constraints.

- ▶ Businesses need to reach out to Web based audiences, streamlining business processes to interact with customer requirements and company-wide information more effectively.
- ▶ Business functions need to integrate more tightly and seamlessly with external business value networks, such as suppliers and manufacturers.
- ▶ There is a need for seamlessly interlinking employees and specific areas of business information to produce efficient and effective streamlined working groups.
- ▶ Consolidated view of the truth, encompassing company-wide view of all disparate statistics

### 1.1.2 Business on demand

To keep up with the evolution of e-business computing, companies in every industry have to act and react on demand. Responding to any customer request, each market opportunity and every external competition requires direct access to computers, software, data, and other resources. Working with heterogeneous systems and increased bandwidth capabilities requires integration between people, processes, and information. And that integration must extend across the company, partners, suppliers, and customers.

Figure 1-1 represents the transformation to a fully integrated business across people, processes, and information, including suppliers and distributors, and customers and employees. Integration, automation, and virtualization are the three key elements of this on-demand operating environment:

- ▶ Integration is the efficient and flexible combination of data to optimize operations across and beyond the enterprise. It is about people, processes, and information.
- ▶ Automation is the capability to increasingly automate business processes with the ultimate goal of self-regulation, thus reducing the complexity of data management to enable better use of assets.
- ▶ Virtualization provides a single, consolidated view of and easy access to all available resources in a network, no matter where the data resides, or the type of data source.

IBM defines an on demand business, as an enterprise whose business processes integrate end-to-end across the company with key partners, suppliers, and customers in order to respond with speed to any customer demand, market opportunity, or external threat.

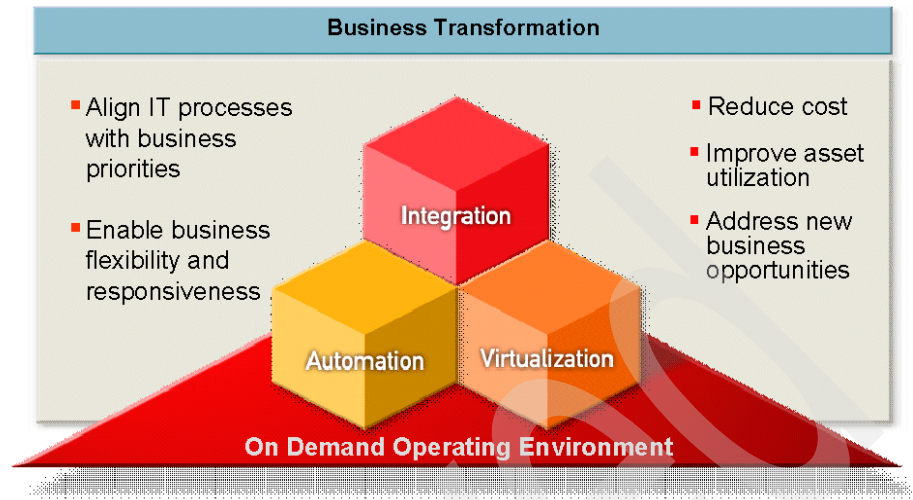


Figure 1-1 Business transformation

For a business to successfully attain and maintain data and information attributes, it must build an IT infrastructure that is designed to specifically support the business' goals. The on demand operating environment is the end-to-end enabling of IT infrastructure that allows a business to execute IT operations synchronized with its business strategy. It is an integrated platform, which must be based on *open standards*, which enable rapid deployment and integration of business applications and processes.

Combined with an environment that allows true virtualization and automation of the infrastructure, the on demand operating environment enables delivery of IT capability on demand.

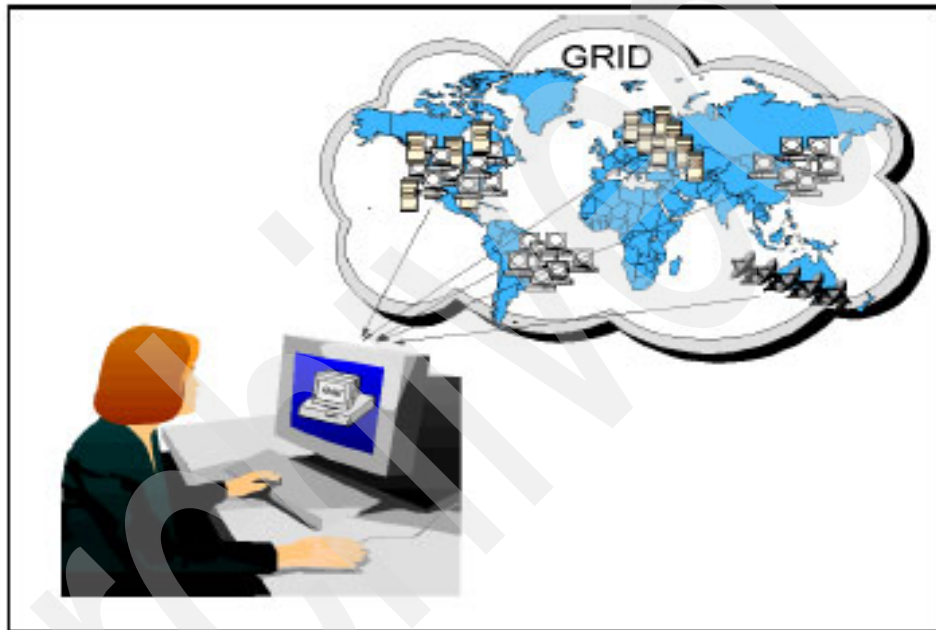
Assisting with the rapid deployment of an on demand operating environment, the federation of the information into a seamless format is also fundamental in insuring this information is fully utilized.

### 1.1.3 From on demand to grid computing

Grid computing, most simply stated, is distributed computing taken to the next evolutionary level. The grid provides an infrastructure on which to support a large collection of communication resources such as both hardware and software.

The goal is to create the illusion of a simple yet large and powerful self managing virtual computer out of a large collection of connected heterogeneous systems, sharing various combinations of resources. See Figure 1-2. The standardization

of communications between heterogeneous systems created the Internet explosion. The emerging standardization for sharing resources, along with the availability of higher bandwidth, are driving a possibly equally large evolutionary step in grid computing. Another function of the grid is to better balance resource utilization. An organization may have occasional unexpected peaks of activity that demand more resources. If the applications are grid enabled, they can be moved to under utilized machines during such peaks. In fact, some grid implementations can migrate partially completed jobs. In general, a grid can provide a consistent way to balance the loads on a wider federation of resources.



*Figure 1-2 Grid virtualization*

Grid computing in its purest form deals with direct access to computers, software, data, and other resources, as is required by a wide range of problem-solving and resource-brokering strategies emerging in industry, science, and engineering. Working with heterogeneous systems and increased bandwidth capabilities, the grid system further utilizes open standards, resulting in greater information sharing.

The simplest example of a grid is to run an existing application on a different machine. If then we consider the use of open architectures and the ways in which computers, hardware, and software can be further extended, this dramatically opens up possibilities. The grid exploits latent hardware and software by directing communication traffic to under utilized or preferred areas.

By working in sync with the grid architecture and networked information, the utilization of remote access is further extended. In turn this increases the speeds available in the network, which provides the ability for greater turnaround of information, faster more efficient networking, and a better use of resources. This in turn, opens up the ability to conduct fast problem-solving and leading edge resource-brokering work processes on a global scale.

The grid infrastructure provides an environment in which IT systems can be universally accessed and utilized to maximize the full potential usage, in some cases eradicating the need for duplicating IT hardware and software.

#### **1.1.4 From grid to federated data**

An increasing number of grid applications manage data at very large scales of both size and distribution. The complexity of data management on a grid arises from the scale, dynamism, autonomy, and distribution of data sources. These complexities should be made transparent to grid applications through a layer that enables ease of data access and processing, dynamic migration of data for workload balancing, parallel data processing, and collaboration. An initial step towards providing such a level of transparency comes from data federation.

Federation is the ability to transparently access diverse business information from a variety of sources and platforms as though it were a single resource. A federated server may access data directly (by accessing a relational database) or access an application that creates and returns data dynamically (as a Web service). The federated approach to information integration provides the ability to synchronize distributed data without requiring that it be moved to a central repository, as shown in Figure 1-3.

Based on ongoing research investments and proven data management technologies in areas such as relational data, XML, content management, federation, search, and replication, IBM has developed the integrated infrastructure shown in Figure 1-3. Data federation uses SQL as the single language to access all data sources. This enables all data sources to be accessed in a standardized format, whether they are an application program, tool, or program product.

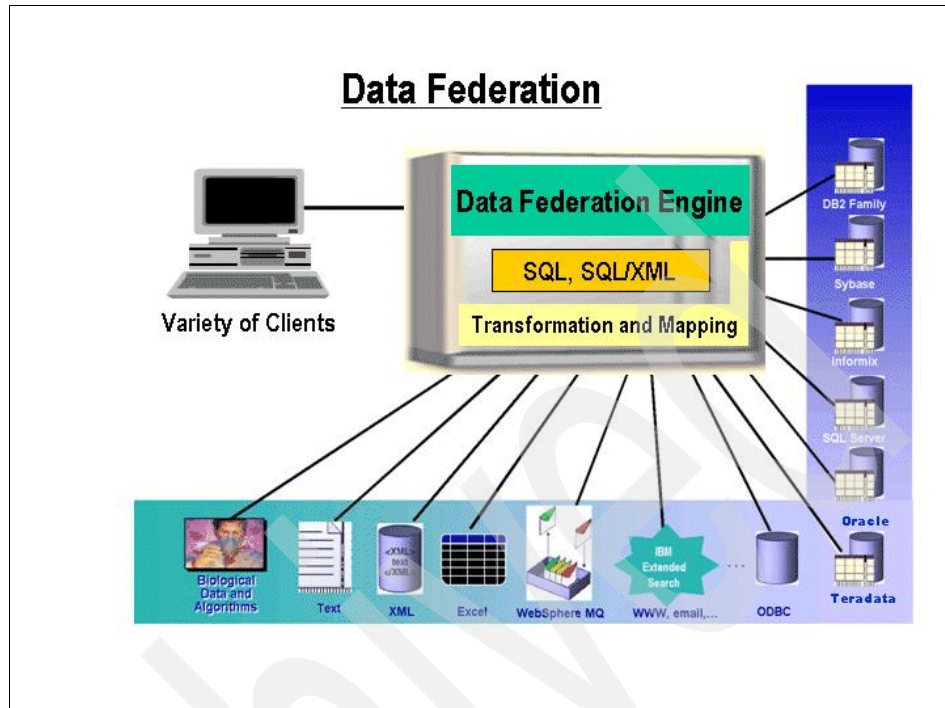


Figure 1-3 Data federation

This is fundamental to IBM's approach and focus, which centers on developing open standards-based solutions, which offer customers not only more benefits than proprietary ones, but greater return on investment as well. In fact, the capabilities of the operating environment must be based on open standards and must support heterogeneous systems. Greater demands to access complex types of information, with fast turnaround speeds, require a robust underlying infrastructure.

## 1.2 IBM's information integration

Distributed data is the reality in most modern enterprises, even without grid computing. Competition, evolving technology, mergers, acquisitions, geographic distribution, and the inevitable decentralization of growth, all contribute to create a diversity of sites and data formats in which critical data is stored and managed. But it is only by combining (integrating) the information from these systems that an enterprise can realize the full value of the data it contains. The two primary approaches to integrating information are consolidating data for local access, and accessing data in place through federation technology.

## 1.2.1 Consolidation and federation

Consolidating data into a single physical data store has been the best way to achieve fast, highly available, and integrated access to related information. Creating a single physical copy lets businesses meet access performance or availability requirements, deliver snapshots that are point-in-time consistent, and provide sophisticated transformation for semantic consistency. Consolidated data stores, which are typically managed by extract, transform, load, or replication processes, have been used extensively for data warehousing solutions and are the standard choice for information integration today. The drawbacks in this solution are generally:

- ▶ Cost  
Significant additional administration, server, and storage costs
- ▶ Latency  
Lack of currency in instances where the difference between the copy and the source is important.

Information integration is the middleware technology that lets applications access diverse and distributed data as if it were a single source, regardless of location, format, or access language. The performance is typically slower than for a consolidated data store because of the distributed locations involved. But, if the data sources are sufficiently consistent, this solution can:

- ▶ Reduce implementation and maintenance costs  
No need for the additional hardware for servers and storage, skills, and personnel costs
- ▶ Access current data from the source of record  
Allowing real-time transactions
- ▶ Combine different sources of traditional data with mixed format data  
Materializing a single image of the data by federating them at the time of access

Integrating information in heterogeneous data environments generally provides quicker return on IT investments for many applications than data centralization in single, large database systems. Centralization is justified by performance or the need to make the data consistent. However, both approaches are necessary at times. A simple example would be a warehouse for historical views (summarized and replicated) and federated access to real-time production data for business activity management. With replication and federation both being part of the integration infrastructure, both these complementary requirements can be satisfied.



## 1.2.2 Transparency, transparency

IBM developed DB2 Data Joiner as the first vehicle for federated technology. Later federated database technology was delivered with DB2's UDB Version 7.1. This product provides a unified access to diverse and distributed data belonging to IBM's DB2 family as well as Informix IDS. Information integration uses and extends the federated approach by providing the ability to synchronize distributed data without requiring that it be moved to a central repository.

Exploding the functionality, the DB2 Information Integrator products build on the federated architecture. The federated DB2 server now provides transparent access to a large and growing number of heterogeneous, distributed data sources. The federated DB2 provides two kinds of *virtualization* to users:

- ▶ Heterogeneity transparency

It is the masking of the data formats at each source; the hardware and software they run on; how data is accessed at each source (through what programming interface or language); and even about how the data stored in these sources is modeled and managed.

- ▶ Distribution transparency

It is the masking of the distributed nature of the sources and the network communication needed to access them.

A federated system looks to the application developer like a regular DB2. A user can run queries to access data from multiple sources, joining and restricting, aggregating and analyzing data with the full power of SQL. A user can also update the data, if they have the right permissions at the sources. Yet, unlike JDBC, ODBC, or ADO, the data sources in a federated system need not be DBMSs, but can be anything from flat files, to application programs, to XML.

A typical federated system is shown in Figure 1-3. Applications can use any supported interface (including ODBC, JDBC, or a Web service client) to interact with the federated DBMS. Figure 1-4 goes into a little more detail. The federated DBMS communicates with the data sources by means of software modules called *wrappers*.

## Federated access with DB2 Information Integrator

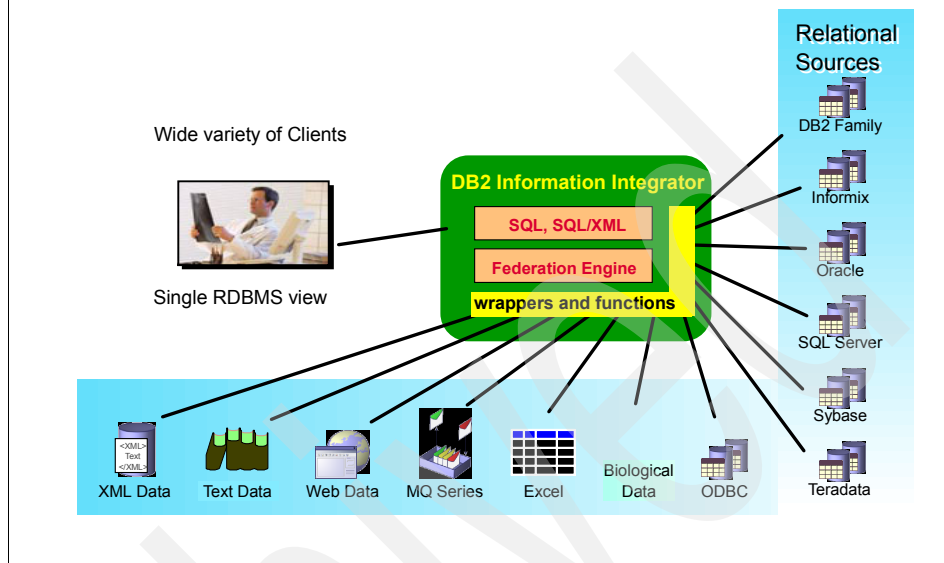


Figure 1-4 Using wrappers

When an application submits a query to the federated system, the federated DB2 identifies the relevant data sources, and develops a query execution plan for obtaining the requested data. The plan typically breaks the original query into fragments that represent work to be delegated to individual data sources, as well as additional processing to be performed by the federated DB2 to further filter, aggregate, or merge the data. The ability of the federated DB2 to further process data received from sources allows applications to take advantage of the full power of the query language, even if some of the information requested comes from data sources with little or no native query processing capability, such as simple text files.

The federated DB2 has a local data store to cache query results for further processing.

DB2 Information Integrator's federated technology enables customers to abstract a common data model across diverse and distributed data and content sources, and to access and manipulate them as though they were a single source. This product set supports the predominantly read-access scenarios common to enterprise wide reporting, knowledge management, business intelligence, portal infrastructures, and customer relationship management satisfying your specific business integration requirements.

## 1.3 The DB2 Information Integrator family of products

In this section we introduce the products that are part of the DB2 Information Integrator family. For more information, refer to the product announcement material available from the following Web sites:

<http://www.ibm.com/software/data/integration/>  
<http://www.ibm.com/common/ssi/>

The concept of federated access to diverse data sources is not new, but the DB2 Information Integrator implementation, based on recent DB2 technology and several research studies, is part of a new generation. IBM customers have known and used DB2 DataJoiner, DB2 Relational Connect, DB2 Life Sciences Data Connect, or the IBM DiscoveryLink® service offer; DB2 Information Integrator makes significant advances over earlier technologies in the areas of:

- ▶ **Performance**  
Materialized query tables (MQTs) allow you to cache query results to answer other similar queries without re-executing access and data transmission. Query compiler enhancements add more intelligence to the optimizer's choices.
- ▶ **Usability**  
Federation and replication administration is integrated into the Control Center, and wizards are on hand for the configuration steps.
- ▶ **Supported sources**  
DB2 Information Integrator extends source access to rich document or content repositories, e-mail systems, and the Web.
- ▶ **Query capabilities**  
DB2 DataJoiner uses a restricted DB2 UDB Version 2 SQL capability, while DB2 Information Integrator uses the DB2 UDB Version 8 SQL with its extended function in the area of business intelligence.

IBM DB2 Information Integrator software, shown in Figure 1-5, provides the foundation for a strategic information integration framework. Such a framework helps customers to access, manipulate, and integrate diverse and distributed data in real time.

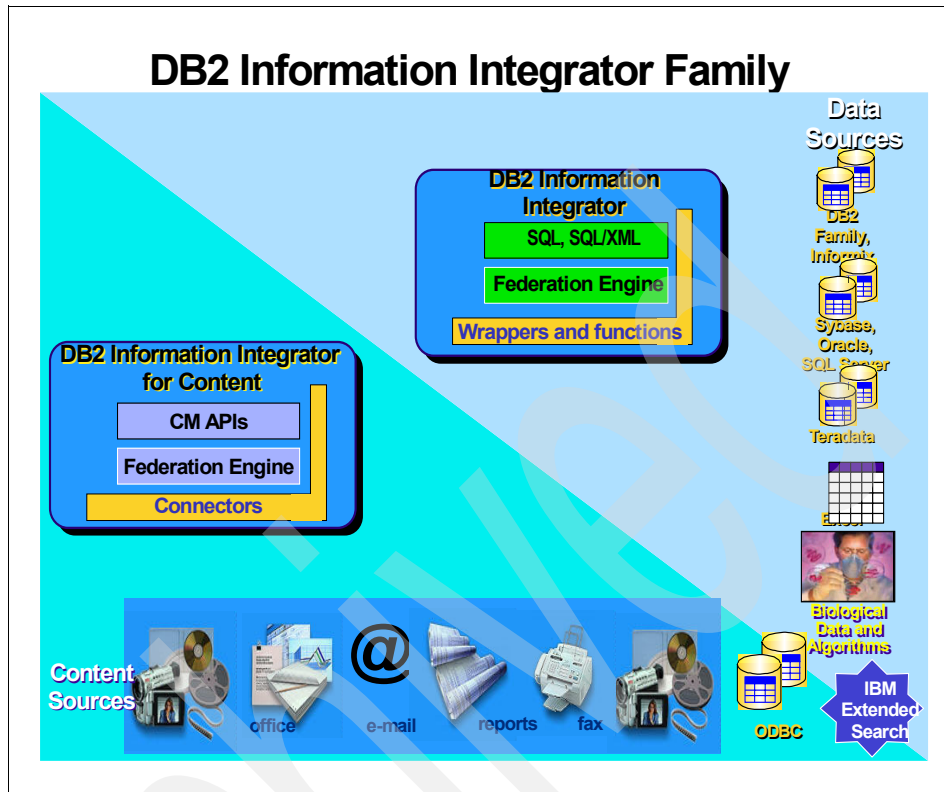


Figure 1-5 The DB2 Information Integrator products

The products are:

- ▶ IBM DB2 Information Integrator, V8.1, a new product based on DB2 information management technology
- ▶ IBM DB2 Information Integrator for Content, V8.2, formerly IBM Enterprise Information Portal

Each product is aimed at the user community defined primarily by the type of data accessed.

### 1.3.1 DB2 Information Integrator

DB2 Information Integrator is targeted at the application development community familiar with relational database application development. Applications that use SQL, or tools that generate SQL (such as integrated development environments, reporting and analytical tools, and so on), can now access and manipulate distributed and diverse data through a federated data server.

DB2 Information Integrator is most appropriate for projects whose primary data sources are relational data augmented by other XML, Web, or content sources. DB2 Information Integrator is based on the DB2 technology infrastructure, leveraging IBM's prior investments in products such as IBM DB2 DataJoiner, IBM DB2 Relational Connect, Life Sciences Data Connect, and IBM DiscoveryLink. DB2 Information Integrator is built on DB2 UDB.

DB2 Information Integrator includes the ability to federate, search, cache, transform, and replicate data. As a federated data server, it provides out-of-the box access to DB2 Universal Database™, IBM Informix products, as well as databases from Microsoft, Oracle, Sybase, and Teradata. In addition, it can also access semi-structured data from WebSphere MQ messages, XML documents, Web services, Microsoft Excel, flat files, ODBC or OLE DB sources, plus a variety of formats unique to the life sciences industry. Integrated support for IBM Lotus Extended Search provides the solution's broad content access to a variety of content repositories, including DB2 Content Manager, as well as e-mail databases, document repositories, third-party Internet search engines, and LDAP directories. In addition, a developer's kit allows one to extend the federation capability to virtually any data source.

DB2 Information Integrator also includes a replication server for non DB2 relational databases. Customers can replicate data between IBM (DB2, including IBM Informix), Microsoft, Oracle, Sybase, and Teradata (target only) databases. You can configure a variety of topologies, latencies, and consistency characteristics.

DB2 Information Integrator represents IBM's first giant step toward information on demand. An initial approach to the unified view of the data regardless of differences in data format, location, and access interfaces with dynamically managed data placement to match availability, currency, and performance requirements; and autonomic features reduce the burden on IT staffs for managing complex data architectures.

For more information on the product architecture, please refer to *Using the federated database technology of IBM DB2 Information Integrator*, white paper by Anjali Grover, Eileen Lin, and Ioana Ursu, available from the Web site:

<http://www-3.ibm.com/software/data/pubs/papers/#iipapers>

### 1.3.2 DB2 Information Integrator for Content

Both DB2 Information Integrator for Content and DB2 Information Integrator enable customers to abstract a common data model across diverse and distributed data and content sources, as well as to access and manipulate them as though they were a single source. The two products are differentiated however by the data that users access, and the developers who use it. DB2 Information

Integrator V.8.1 is targeted at the application developers who are familiar with relational database application development.

DB2 Information Integrator for Content is targeted at the content application developer who needs to search for and access text and non-text information across a wide range of content sources. Providing seamless reach into diverse data environments, DB2 Information Integrator for Content represents a renaming and repositioning of the Enterprise Information Portal offering.

DB2 Information Integrator for Content offers a rich set of integration features, such as connectors to diverse content sources, sophisticated information mining, and advanced workflow. To speed implementation time of content integration projects, DB2 Information Integrator for Content provides access to a variety of data sources out-of-the-box, all of which can be federated into a single search. These connectors include access to the DB2 Content Manager family and other content repositories, Lotus databases, relational databases, and wide-ranging content available with IBM Lotus Extended Search.

Additionally, DB2 Information Integrator for Content includes a sophisticated information mining capability that uses Web crawling and text-mining algorithms to provide structure to unstructured content. The mining algorithms include the ability to identify the language in which a document is written, identify such features within documents as names, classify documents according to a defined taxonomy, groups documents by category, and summarizes documents. By building additional knowledge about enterprise-wide information, businesses can reap additional return from existing content assets.

Finally, DB2 Information Integrator for Content provides an advanced workflow application to enable businesses to increase productivity, reduce production times, and improve communication and collaboration. Using a graphical workflow builder, developers easily define workflow processes that incorporate the results of queries to DB2 Information Integrator for Content for use across the enterprise.

### **1.3.3 Extension through partnership**

DB2 Information Integrator is middleware that needs to be placed in a solution context. IBM worked extensively with complementary IBM software group offerings, partner ISVs, and systems integrators to enhance analysis and reporting capabilities, enable rapid application development, enrich function delivery, extend access to additional data sources, and expedite solution delivery.

DB2 Information Integrator can be used with any industry leading analytical and reporting tools that work with DB2 UDB V8.1, such as Business Objects, Brio, Cognos, Crystal Decisions, MicroStrategy, SAS, and so on. Ascential and

Informatica provide the complementary ETL technology for data profiling, transformation, integration, and consolidation capabilities.

Other ISV partners enrich feature capabilities for XML exploitation or customer data integration. Nimble Technology has packaged an offering for DB2 Information Integrator that brings the full expressiveness of an XML query language to the middleware. Cross Access, Information Builders, Neon Systems, and Striva Corp. extend DB2 Information Integrator reach to classic legacy sources such as Adabas, IMS, VSAM, CA-IDMS, and CA/Datacom.

DB2 Information Integrator fits easily into a service-oriented architecture. Any SQL operation or stored procedure operating over diverse and distributed data can return an XML result through a Web service request. Built-in tooling with WebSphere Studio automatically converts SQL operations into Web services description language operations.

DB2 Information Integrator can reduce cost and speed development of complex WebSphere Business Integration processes by simplifying development and maintenance of collaborations that integrate multiple data sources. And, it can improve the performance of process activities or business objects, which can be modeled as sets and benefit from set-based relational processing. DB2 Information Integrator also facilitates business activity management by correlating event information with related real-time data from production systems or historical data from warehouses that may be required for an effective response.

With WebSphere MQ deployments, DB2 Information Integrator gives database programmers access to message queues through the familiar SQL paradigm. So, it simplifies integration between database and messaging systems, and enables analysis of virtual or physical queue snapshots using standard analytical software.

### **1.3.4 DB2 Information Integrator and Life Sciences**

Life Sciences is an all new world of disparate data and applications, different from relational, with a tremendous need for integration. Life Sciences Data Connect enables a DB2 federated system to integrate genetic, chemical, biological, and other research data from distributed sources.

DB2 Information Integrator has a similar architecture and now provides equivalent functions to Life Sciences Data Connect, and more. DB2 Information Integrator supports the data sources depicted in Figure 1-6.

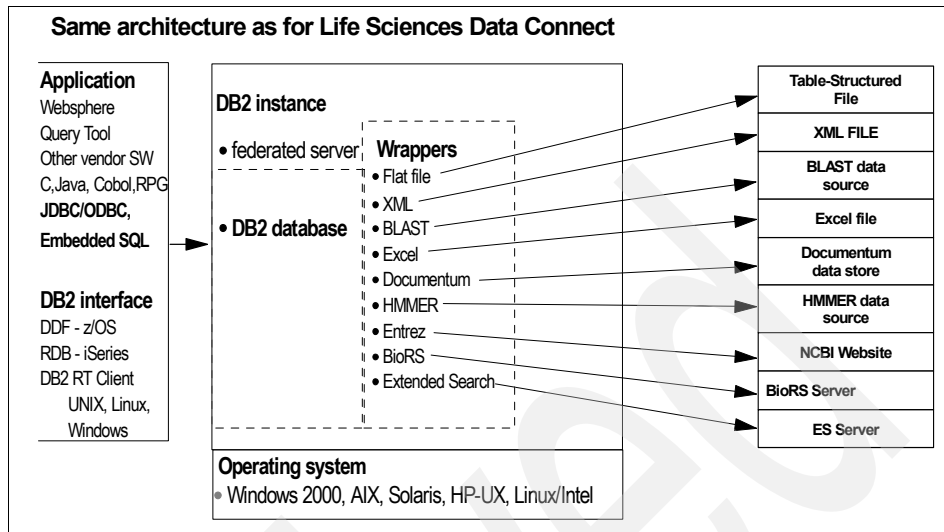


Figure 1-6 The Life Sciences integration environment

All wrappers that were available in the Life Sciences Data Connect V7.2 can be migrated to DB2 Information Integration V8.1.

The DiscoveryLink offering is a set of middleware software and services tailored specifically to life sciences research and development requirements providing a way for scientists and researchers to efficiently combine and query large amounts of data residing in multiple heterogeneous sources and locations. DiscoveryLink is now based upon IBM's DB2 information integration database technology, and uses DB2 Information Integrator rather than the predecessor products.

For details, refer to the Web site:

<http://www.ibm.com/solutions/lifesciences/solutions/discoverylink.html>

### 1.3.5 The evolution

The DB2 Information Integrator products are part of the IBM strategic information integration framework. They have added enhanced capabilities to the following pre-existent products or offerings:

- ▶ DB2 V7.2 with DB2 Relational Connect provides single query access across DB2, Informix, Oracle, Sybase, and Microsoft.
- ▶ IBM DiscoveryLink includes DB2 V7, DB2 Relational Connect and Life Sciences Data Connect providing single query access across DB2, Informix, Oracle, Sybase, Microsoft, Excel, Documentum, flat files, and BLAST for the



life sciences industry. For more information on this offering, you can refer to *IBM Life Sciences Solutions: Turning Data into Discovery with DiscoveryLink*, SG24-6290.

- ▶ DB2 DataJoiner® provides federated read and write support for DB2, Informix, Oracle, Sybase, Microsoft, Teradata (read and write), ODBC (read and write), and more. And, it provides heterogeneous replication among DB2, Informix, Oracle, Sybase, and Microsoft databases.
- ▶ IBM Enterprise Information Portal provides content federation, extended search, text mining, and integrated workflow designed businesses with sophisticated content management requirements. Several Redbooks are available on this topic. Refer to the Appendix “Related publications” on page 413 for their publication numbers.

DB2 Relational Connect and Life Sciences Data Connect are features available with DB2 UDB Version 7 only.

DB2 Information Integrator V8.1 replaces DB2 DataJoiner, DB2 Relational Connect, and DB2 Life Sciences Data Connect. We have already mentioned that DB2 Information Integrator for Content is the replacement for IBM Enterprise Information Portal. The withdrawal from marketing of these products has been announced for September 2004, and a migration plan for current licenses is in place.

Leveraging the DB2 UDB Version 8 code base, this new product introduces several new capabilities that have not been in the market in either DataJoiner or Relational Connect. This new product offers features such as expanded federated sources, MQTs over nicknames, far richer SQL capability, and new replication technology just to name a few. Customers are able to federate far more sources than before including Web services, MQSeries® queues, XML documents, flat files, spreadsheets, and more. Customers who have a software maintenance agreement for DB2 Relational Connect, DB2 Life Sciences Data Connect, or DB2 DataJoiner are entitled to upgrade to a DB2 Information Integrator V8.1 product.

The federation capabilities for unstructured content provide a key element in the IBM framework for information integration. As the IBM DB2 Information Integrator products are delivered and enhanced, DB2 Information Integrator for content is a fundamental offering to help customers access, manipulate, and integrate existing repositories and applications.

Figure 1-7 gives a historical perspective of the evolution. In 1995, DataJoiner V1 was introduced as the first product to provide distributed request support. By creating nicknames, it enabled read-write access to DB2 family member data sources, and non-IBM products including Oracle, Sybase, and Microsoft SQL Server, as well as Informix, Teradata, ODBC, Classic Connect. Version 2.1.1 is

the most current version of DataJoiner. Since its architecture is based on DB2 Common Server V2 (the predecessor of DB2 UDB), functional limitation exists. For example, several functions provided by later DB2 versions are not supported.

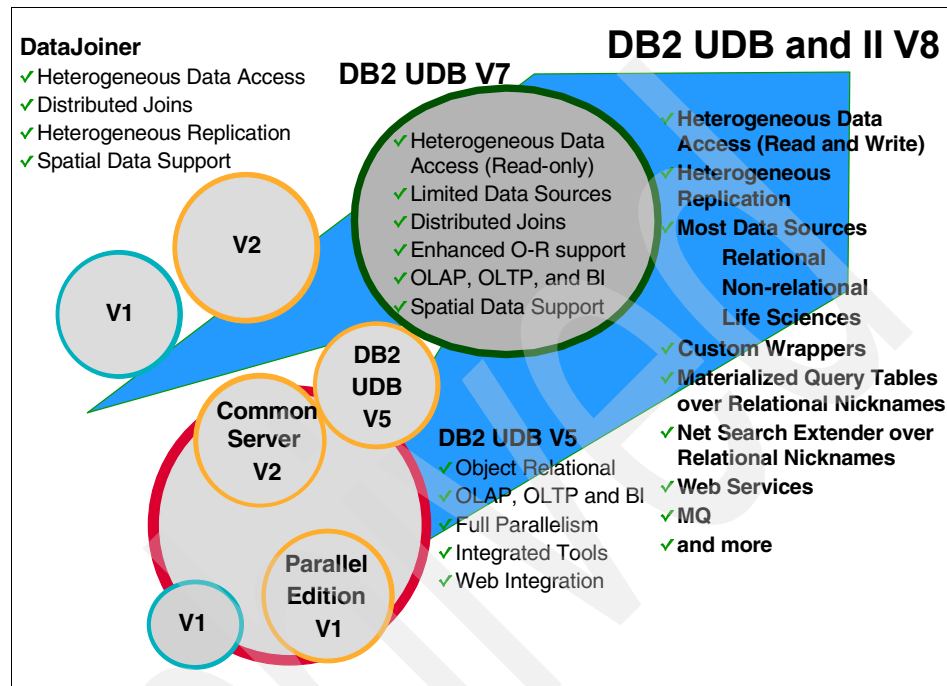


Figure 1-7 The history of DB2 federated products

DB2 UDB for Multiplatforms V8.1 combined with DB2 Information Integrator V8.1 replace most of the old DataJoiner functionality, and provide a lot of additional functionality. Using just DB2 UDB for Multiplatforms V8, access is allowed to the DB2 family products (including Informix).

DB2 Information Integrator V8.1 advances that vision, but there is more to come. XQuery will become a key query language. As structured and unstructured data become more blended, the query languages that express them will blend to provide precise query and free-form search. Data placement must become more policy-driven and require less administration. Autonomic concepts introduced in DB2 UDB V8.1 must be extended to the federated server and replication environments. Metadata management must fulfill its promise to knit together the fabric of the enterprise. DB2 is bringing together the fundamental technologies to realize this vision.

# IBM DB2 Information Integrator V8.1

In this chapter we provide a brief technical overview of DB2 Information Integrator V8. We present its major components, functions, and performance options. The objective is to provide enough information to understand the basics of DB2 Information Integrator before tackling the installation, configuration, and performance topics presented in the following parts of this redbook. More information is provided in the standard documentation listed in Appendix “Other publications” on page 413.

This chapter is structured in:

- ▶ Overview
- ▶ DB2 Information Integrator functions and objects

## 2.1 Overview

With DB2 UDB for Multiplatforms Version 7.1, IBM has introduced the first commercial database management system capable of integrating both relational and non relational data. With the data federation capability (see Figure 2-1) the federated system acts as a virtual database, with remote objects configured similar to local tables. IBM DB2 Information Integrator Version 8.1 has extended the integration capabilities to non-IBM and several non-relational data sources.

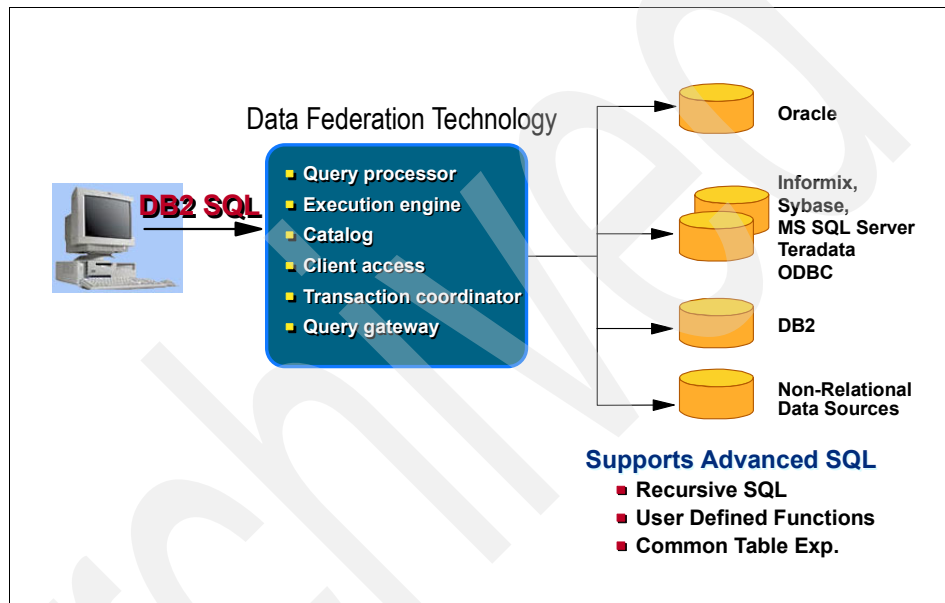


Figure 2-1 Data federation technology

Through the relational and non relational wrappers, a DB2 table, an Oracle table, a simple text file or spreadsheet, a specialized application such as a Documentum document or an XML document, can all be accessed within the same query from a SQL application or tool. Users have the power of DB2 as well as the power of the remote data sources, plus the benefits of transparency, heterogeneity, function compensation, autonomy, extensibility, and optimization.

A DB2 federated system is a special type of DBMS. A federated system consists of a DB2 instance that operates as a federated server, a database that acts as the federated database, one or more data sources, and clients (users and applications) that access the database and data sources.

Applications can query integrated views across diverse and distributed data sources as if they were a single database. The queries can be expressed using

standard SQL statements. SQL expressions can be used to transform the data for business analysis or data exchange. XML documents can be created for flexible presentation. Any Web service can be converted into a function call and used as a transformation. For example, a Web service that provides currency conversion can be used inline within the SQL expression. The queries can produce standard SQL answer sets or XML documents.

## Federated systems

Figure 2-2 shows how the existing federated system architecture has been extended to allow federated access to a large number of diverse data sources.

Starting with DB2 UDB for Multiplatforms V8, the major functions of DataJoiner are implemented in the DB2 engine, including read-write access using nicknames. However, the federated function in DB2 UDB is available only for DB2 and informix data sources. Functions for accessing other data sources are now provided by DB2 Information Integrator, a separately priced product from DB2. Using DB2 Information Integrator, you can *read* and *update* non-DB2 relational data sources. Using DB2 Information Integrator Version 8.1, you can also *read* non relational data sources. This approach maps well to the predominantly read access scenarios common to enterprise wide reporting, business intelligence, portal infrastructures, and customer care requirements.

Applications can insert, update, or delete rows to federated relational databases. Initially, this is limited to single-site updates with only one-phase-commits. This means that the only update action supported is updating nicknames for tables at the same data source. Selecting from nicknames at other data source and from tables in the Information Integrator database is supported within this transaction. In fact, the update statement itself can select from nicknames at other data sources.

The following are not supported in a single transaction (commit):

- ▶ Updating nicknames for tables at more than one data source
- ▶ Updating nicknames and also tables in the Information Integrator database

Also, applications that require a distributed unit of work (multi-site update, two-phase-commit), such as CICS® and IMS transaction manager on mainframes, cannot update nicknames. However, these applications are able to select from nicknames.

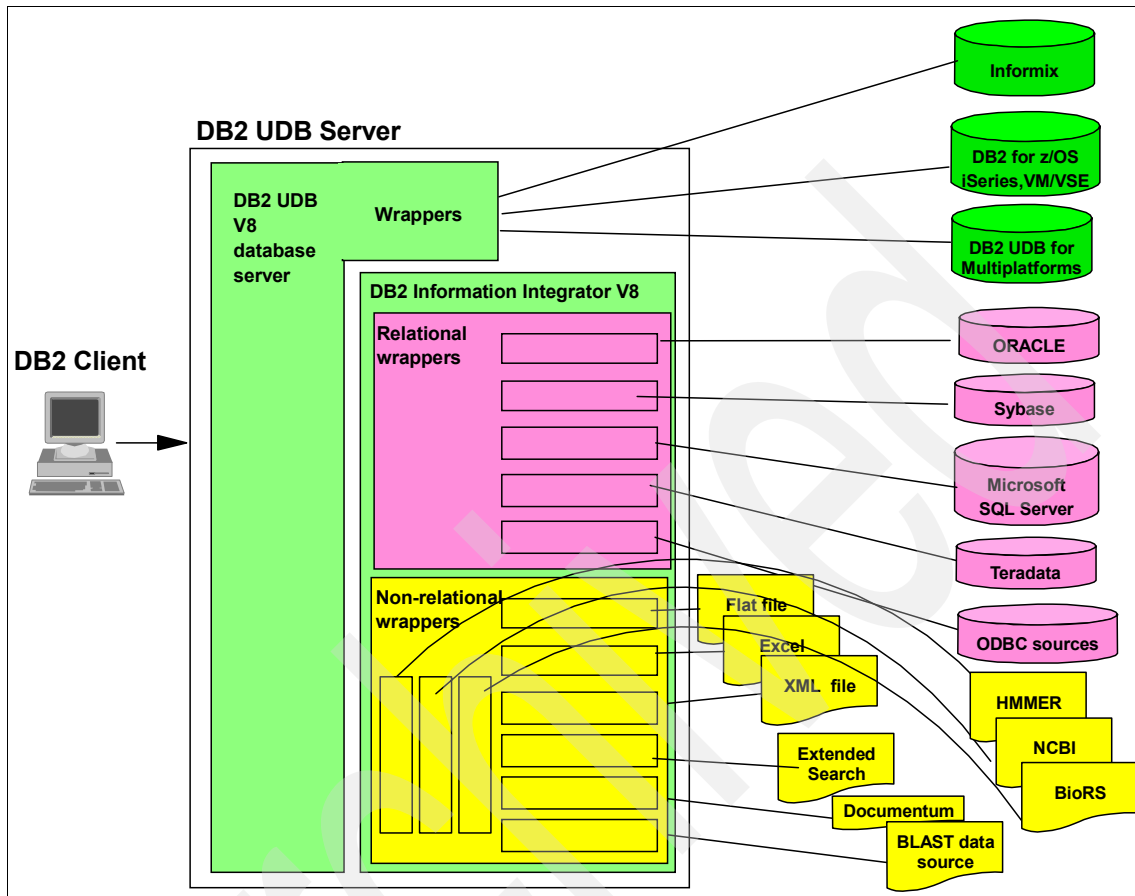


Figure 2-2 Federated database with DB2 Information Integrator V8.1

In a future version, DB2 Information Integrator will support multi-site update (two phase commit) with nicknames.

In general, applications should update data sources through the application API to preserve business rules. Applications can access the federated server through traditional database clients or Web service clients. Stored procedures can access the federated data.

With a federated system, you can send distributed requests to multiple data sources within a single SQL statement. For example, you can join data that is located in a DB2 Universal Database table, an Informix table, and an XML tagged file in a single SQL statement.

Data replication technology is also provided with DB2. Both log-based change capture and refresh styles, of replication are supported across the DB2 family. With DB2 Information Integrator, non-DB2 databases can also be the replication target or source.

The federation technology has features such as:

- ▶ Write operations against remote relational sources
- ▶ Empowering the user to define materialized query tables (MQTs) over relational nicknames
- ▶ A larger set of non relational wrappers (including the XML wrapper)
- ▶ Enhanced GUI for federated system configuration
- ▶ Web services, WebSphere MQ, UDFs
- ▶ Text index support from Net Search Extender (NSE) over relational nicknames

These federation features make DB2 Information Integrator an ideal product for building an enterprise federated system.

DB2 Information Integrator V8.1 is based on industry standards such as SQL, XML, Java™, and Web services to provide broad inter-operability. The product's access-in-place capabilities reduce the need to rewrite or replace systems to make it work together. The product provides a strategic, reusable, and open information integration platform, which matches a variety of specific business needs and strategies.

## DB2 Information Integrator Editions

DB2 Information Integrator is available for IBM AIX, Linux for Intel®, Microsoft Windows, Hewlett Packard HP-UX, and Sun Solaris Operating Environment. DB2 Information Integrator is available in the following editions: Replication, Standard, Advanced, and Developer. See Table 2-1 for a quick summary. For details and current information refer to the offerings information available from:

<http://www.ibm.com/common/ssi/>

Table 2-1 DB2 Information Integrator packaging

DB2 Information Integrator Components	Replication Edition	Standard Edition	Advanced Edition	Developer Edition
DB2 ESE	X	X	X	X
Relational wrappers	X	X	X	X
Non relational wrappers		X	X	X

DB2 Information Integrator Components	Replication Edition	Standard Edition	Advanced Edition	Developer Edition
Net Search Extender		X	X	X

## DB2 Information Integrator components

DB2 Information Integrator contains the following components:

- ▶ DB2 Enterprise Server Edition is a multiuser version of DB2 that you can use to create and manage non partitioned or partitioned database environments.
- ▶ The relational wrappers are used for non-IBM relational databases. In DB2 for Multiplatforms Version 8, relational wrappers are required if you want to access data that is stored in Oracle, Sybase, Microsoft SQL Server, ODBC, and Teradata data sources.
- ▶ Non relational wrappers are used by the DB2 federated system to integrate non relational data sources, such as flat files and XML files, and genetic, chemical, biological, and other research data from distributed sources.
- ▶ DB2 Net Search Extender is used to perform SQL-based searches on full-text documents across your enterprise. DB2 Net Search Extender performs searches efficiently by using text indexes, which Net Search Extender updates dynamically, and stores in-memory reducing scans and physical read operations.

## Data sources

In a federated system you can have different types of remote sources. A *data source* can be a relational DBMS instance (such as Informix, Oracle,...) or a non relational data source (such as BLAST, Excel...). Through some data sources you can access other data sources. For example, through the Lotus Extended Search data source you can access data sources such as Lotus Notes® databases, Microsoft Access, Microsoft Index Server, Web search engines, and Lightweight Directory Access Protocol (LDAP) directories.

The method or protocol used to access a data source depends on the type of data source. For example, DRDA® is used to access DB2 for z/OS data sources, and the Informix client API is used to access Informix data sources. Although there are different protocols and access methods, you still access the data sources through the federated server using native DB2 SQL.

Data sources are semi-autonomous. For example, the federated server can send queries to Oracle data sources at the same time that Oracle applications can access these data sources. A DB2 federated system does not monopolize or restrict access to the other data sources, beyond normal integrity and locking constraints. Table 2-2 shows the currently supported data sources.



Table 2-2 Supported data sources

Type	Data source
Relational data source	DB2, Informix, Oracle, Sybase, Teradata, Microsoft SQL Server, ODBC, OLEDB
Non relational data sources	BioRS, BLAST, Documentum, Entrez, HMMER, IBM Lotus Extended Search, Microsoft Excel, flat (table-structured) files, XML

## Summary

In summary, the power of a federated server has the ability to:

- ▶ Join data from local tables and remote data sources, as if all the data is stored in a local database
- ▶ Update data in relational data sources, as if the data is stored in the federated (local) database
- ▶ Replicate data to and from relational sources
- ▶ Take advantage of the data source processing strengths, by sending requests to the data sources for processing
- ▶ Compensate for SQL limitations at the data source by processing parts of a distributed request at the federated server

## 2.2 DB2 Information Integrator functions and objects

The DB2 Information Integrator architecture provides new functions and introduces new objects. See Figure 2-3. Here we summarize the new terminology and provide a brief description. We will talk about servers, wrappers, and all you need to know before guiding you through the installation and configuration steps of the DB2 Information Integrator product, as shown in the following sections. For more information, refer to the manuals listed in “Other publications” on page 413.

## The basic DB2 federated components

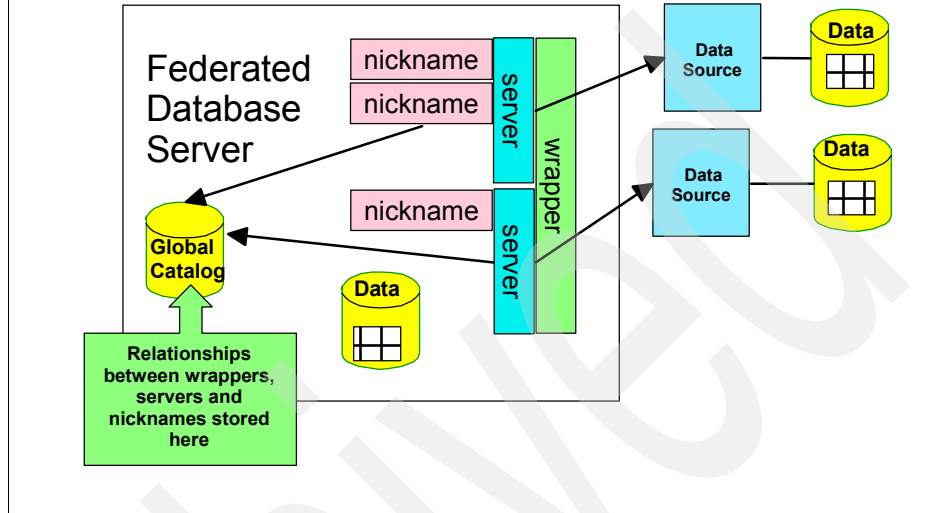


Figure 2-3 The basic federated components

To access data on a remote source, you need to define at least four of the following components at the DB2 Federated Server:

- **Wrappers**

A wrapper is a library that allows access to a particular type of data source or protocols such as Oracle, Sybase, Generic ODBC, Web Services, MQ, etc. It contains information about the remote data source characteristics, and it understands its capabilities.

- **Servers**

A server represents a specific data source that is accessed through a wrapper. There are DDL statements to define a server for every data source that you want to use.

- **Nicknames**

It is a local alias to data object on a remote server (mapped to rows and columns). You define a nickname for any remote data object that you want to use on a given server. In queries against your federated server, you can then use the nicknames as if they were local tables. In fact, the end user or application does not even need to know that they are nicknames, and that they represent remote tables or non relational data.

- ▶ User mapping

It provides a mapping from a user ID that accesses the federated server to a data source user ID and password that Information Integrator will include in connections to the data source on behalf of the federated user.

Two more types of objects are used when accessing data sources:

- ▶ Data type mappings

The data types at the data source must map to corresponding DB2 data types to allow the federated server to retrieve data. The default mappings are in the wrapper library, but additional semantic definitions may be needed.

- ▶ Function mappings

The wrapper library also contains default function mappings between a DB2 built-in function, and the relational data source equivalent. Additional user function mapping may be needed in case of pushing down a function to the data source that does not have a counterpart in DB2.

Another important component is the:

- ▶ Global catalog

The federated database catalog, called the global catalog, contains the definitions of all the previous objects.

The data types at the data source must map to corresponding DB2 data types, so that the federated server can retrieve data from data sources.

## 2.2.1 Database server

The database server, DB2 UDB, provides integrated metadata management and the engine technology for DB2 Information Integrator. DB2 UDB is also used for the metadata store and the caching of nickname in MQTs for better federated query performance.

DB2 Information Integrator Advanced Edition and Advanced Edition Unlimited make the database server available for direct use by applications. For example, a company performing complex analytical calculations over XML, spreadsheets, and a series of relational databases can store and share interim computational results using the DB2 UDB database embedded in the DB2 Information Integrator.

## 2.2.2 Federated database

To end users and client applications, all data sources appear as a single collective database in DB2. Users and applications interface with the *federated*

*database* managed by the federated server. The federated database contains a system catalog. The federated database system catalog has entries that identify data sources and their characteristics. The federated server consults the information stored in the federated database system catalog and the data source wrapper to determine the best plan for processing SQL statements.

The federated system processes SQL statements as if the data sources were ordinary relational tables or views within the federated database. As a result:

- ▶ The federated system can join relational data with data in non relational formats. This is true even when the data sources use different SQL dialects, or do not support SQL at all.
- ▶ The characteristics of the federated database take precedence when there are differences between the characteristics of the federated database and the characteristics of the data sources.

## **Federated server**

Any DB2 server where DB2 Information Integrator is installed, is referred to as *federated server*. You can use existing DB2 instances as federated servers, or you can create new ones specifically for the federated system.

The DB2 instance that manages a federated system is called a *server* because it responds to requests from end users and applications and is the single interface for federated access. Usually the federated server often sends part of the requests to the data sources for processing. An operation that is executed remotely is called a *pushdown* operation.

DB2 UDB for Linux, UNIX®, and Windows can be configured as a federated server. Neither the DB2 for iSeries (AS/400®) nor DB2 for z/OS can be a federated server; however, you can access iSeries and DB2 for z/OS data from the federated server as part of a distributed request.

The federated data server gives applications real-time, integrated access to diverse and distributed data as if it were a single source, regardless of format, location, or operating environment. A federated server provides:

- ▶ Transparency, which masks from the user the differences, idiosyncrasies, and implementations of the underlying data sources
- ▶ Heterogeneity, the ability to federate highly diverse types of data, including structured (for example, relational databases), semi structured (for example, XML documents), and unstructured (for example, free-form text) data
- ▶ Extensibility, meaning that with federation it can be easy to add a new data source to the federation

- ▶ Rich functionality, including the full function available through supported query languages, compensation for missing functions in the back-end data sources, and the ability to incorporate source-specific capabilities into the query language
- ▶ Autonomy for data sources, allowing sources to be federated with little or no impact on existing applications or systems
- ▶ Performance, which depends on sophisticated optimization technology.

A federated data server lets applications query distributed data as if it were a single source.

With DB2 Information Integrator, administrators start by configuring access to the required data sources and may additionally define integrated views across them. DB2 Information Integrator provides built-in access to relational databases, XML documents, content repositories, document repositories, e-mail systems, spreadsheets, flat files, message queues, Web services, the Web, and more. Independent software vendors, systems integrators, or staff developers can extend access to virtually any source in the enterprise. Once configured, these federated objects appear to be simple tables or views on the federated server.

DB2 Information Integrator can sit transparently behind tools from Business Objects, Cognos, Crystal Decisions, and others, enabling end users to combine historical data with real-time data, develop reports that span departmental datamarts or warehouses, and enrich reports with relevant details from content repositories.

Developers use standard development tools, including WebSphere Studio or Microsoft Visual Studio.Net, to build composite applications using standard SQL. The federation engine takes on the burden of query decomposition (a source of accuracy issues for hand-coded tasks), query optimization for reasonable responsiveness, translation between incoming and outgoing SQL and native APIs, connection management, and data type and platform mismatches (for example, EBCDIC/ASCII conversions). IBM experiments indicate that businesses can net a 40% to 65% reduction in hand-coding projects that have to integrate multiple data sources. You can see the details of this experiment at IBM developerWorks® at the Web site:

<http://www7b.software.ibm.com/dmdd/library/techarticle/0305saracco/0305saracco.html>

XML has become the “lingua franca” for data interchange, and DB2 Information Integrator delivers built-in function to search, store, compose, transform, and validate XML documents. Queries can produce standard SQL answer sets or XML documents. SQL expressions can be used to transform the data for business analysis or data exchange.

XML documents can be transformed using XSL for flexible presentation. Any Web service can be converted into a function call and used as a transformation. For example, a Web service that provides currency conversion can be used inline within the SQL expression. Thus, a single SQL request can access disparate data, transform attributes as needed, compose an XML document, validate it against a DTD or schema, and publish it to WebSphere MQ.

Applications can access the federated server through traditional database clients or Web service clients. Applications can also insert, update, or delete rows from federated relational databases. Initially, this capability is limited to single-phase commit.

### 2.2.3 Configuring the federated system

The DB2 Federated Server allows you to access and join data from relational and non relational data sources. This functionality was originally made available in the DataJoiner product, but has been progressively integrated into DB2 V7 and V8. By setting the database manager configuration parameter *federated* to *yes*, the DB2 instance allows federated access to other DB2 sources, Informix, and any OLE DB source.

If you need access to other, non IBM, relational sources such as Oracle, Sybase or Microsoft SQL databases as well as generic ODBC access, and Teradata, then you need to install the DB2 Information Integrator product. After you have installed DB2 Information Integrator, you need to register your data sources by following this process:

1. Ensure that the wrapper module that can access the new data source is registered as a Wrapper in the federated database.
2. Define the new data source as a server in the federated database and customized the server options if necessary.
3. Register the remote authentication information with the federated system in the form of user mappings for each DB2 user that you want to provide access to.
4. Test the connection from the federated system with a pass through session. A pass through (passthru) session allows you to send SQL statements directly to a data source. Ensure proper privileges are granted to those users that can use the pass-through session for this new data source.
5. Determine whether additional data type mappings need to be defined if you are connecting to a relational data source. Specifying additional data type mappings is necessary if you want to change the default mapping between a DB2 data type and a remote data type in the next step.

6. Identify the remote tables as nicknames at the federated server and customize each with options either at the table level, or at the column level, or both. Options are specific to each remote data set. Ensure proper privileges are granted to the users that need access to each data set.

If you are accessing a relational data source, determine whether you need to register additional mappings between the DB2 functions and the remote data source functions. These mappings allow data source functions to be used by the federated system

## Defining wrappers

*Wrappers* are mechanisms by which the federated server interacts with data sources. The federated server uses routines stored in a library called a *wrapper module* to implement a wrapper. These routines allow the federated server to perform operations such as connecting to a data source and retrieving data from it. The wrapper encapsulates data source information and models data as tables. It is aware of the characteristics of the data source and it can expose unique functions. A wrapper provides the programming logic to facilitate the following tasks:

- ▶ Federated object registration

A wrapper encapsulates the data source characteristics from the federated engine. A wrapper knows what information is needed to register each type of data source.

- ▶ Communication with the data source

Communication includes establishing and terminating connections with the data source and maintaining the connection across statements within an application if possible.

- ▶ Services and operations

Depending on the capabilities of the type of data sources that a wrapper is meant to access, different operations are supported. The operations can include sending a query to retrieve results, updating remote data, transaction support, large object manipulation, input value binding, and more.

- ▶ Data modelling

A wrapper is responsible for mapping the data representation of the result of remote queries into the table format as required by the federated engine.

Wrappers are available for each type of data source. For example, if you want to access three DB2 for z/OS database tables, one for DB2 for iSeries™ table, two Informix tables, and one Informix view, you need to define only two wrappers: one for the DB2 data source objects and one for the Informix data source objects.

Once these wrappers are registered in the federated database, you can use these wrappers to access other objects from those data sources.

Example 2-1 shows an XML wrapper definition.

*Example 2-1 Creating a wrapper*

---

```
CONNECT TO FEDDB
CREATE WRAPPER XML LIBRARY 'libdb2lxml.a'
```

---

In Chapter 4, “Configuring DB2 Federated Data support” on page 73, and Chapter 5, “Installing and configuring DB2 Information Integrator” on page 121, we provide several examples of wrapper definitions.

The Wrapper Development Kit enables you to develop your own wrapper modules for non relational data sources. The wrapper module is a shared library with specific entry points that provide access to a class of data sources. DB2 UDB loads it on demand dynamically. The wrapper module is what you will be developing using the specific classes supplied with DB2. It will contain specific building blocks that allow it to act as a translator between your data source and your federated system.

## Defining servers

After wrappers are created for the different data sources, the federated instance owner identifies the data sources and registers each one as a *server*. For relational data sources you need to know the connection specific information. For example, a connection to a specific Informix source is specified through the name of the remote Informix server and the name of the remote database. A server for a relational database management system usually represents a remote database. If the data source is a remote DB2 UDB instance, you can register each database on that instance as a server. Registering each database as a server allows data on each of these remote databases to be accessed from the federated system. Some relational database management systems, such as Oracle, do not use this multiple database per instance structure. Instead, each instance represents a server. On Oracle systems, each instance server ID represents a server. For non relational data sources, you must register a server object because the hierarchy of federated objects requires that specific files that you want to access be associated with a server object.

Example 2-2 shows an XML server definition.



```
CONNECT TO FEDDB  
CREATE SERVER XMLSERV WRAPPER XML
```

---

In Chapter 4, “Configuring DB2 Federated Data support” on page 73, and Chapter 5, “Installing and configuring DB2 Information Integrator” on page 121, we provide several examples of wrapper definitions.

## Server options

Server attributes are used to capture the characteristics of a relational data source at the federated server. These characteristics and restrictions are used by the query compiler in planning the query. These server attributes, which you can set through server options, contain information about data source location, connection security, and some server characteristics that affect performance.

Some attributes such as collating sequence can be adjusted through external server options. For example, data sources, such as DB2 for z/OS, use a collating sequence based on the EBCDIC encoding scheme. The default setting for the server option `COLLATING_SEQUENCE` is ‘N’ for such sources because it is not usual for DB2 for z/OS to have a database with similar encoding. DB2 UDB for Multiplatforms (often still called just UDB in this book) uses mostly ASCII encoding and the default sorting order is dictionary sort. Although some data sources like Oracle use ASCII encoding, their sorting order is different from that of DB2 UDB.

It is possible to create an Information Integrator database with a sorting order to match that of such sources like Oracle; at the time the federated database is created, specify `COLLATE USING IDENTITY` in the `CREATE DATABASE` statement. In this case, the server option `COLLATING_SEQUENCE` can be set to ‘Y’ for sources with ASCII encoding and identity sorting order. This server option allows range comparison (for example `string_col > string_constant`) and `LIKE` predicates to be executed remotely. If the remote data source is case insensitive and the DB2 federated database is set to use case sensitive search collating sequence, the equality comparison operations will also be executed locally.

Pushdown is a very important aspect of federated query processing. If the `PUSHDOWN` server option is set to Y, the optimizer will consider generating a plan that “pushes down” certain parts of the query execution to the remote source. The intent of pushdown is reducing network transport (trips) and exploit the intelligence of the relational remote sources. Pushdown analysis (PDA) is the component of the optimizer that decides which parts of a query can be pushed down and processed remotely at the data sources. The decision of actually

pushing down is cost-based, and influenced by information about the hardware at the remote sources, the network characteristics, the estimated number of rows processed and returned from the remote sources. Pushdown is discussed in Chapter 6, “A performance tutorial” on page 241.

Another external server option for relational nicknames that you might want to consider setting to Y is the DB2\_MAXIMAL\_PUSHDOWN option. For queries that contain nicknames, the federated server determines which operations in the query can be pushed down to the data sources; this option does not affect this phase. But in the next phase called cost optimization, the setting for this option tells the optimizer whether to determine the execution plan based on cost (default behavior), or to favor pushing down the maximum number of operations made eligible for pushdown by pushdown analysis, regardless of cost.

Setting this option to Y directs the query optimizer to favor access plans that tend to allow the remote relational data sources to evaluate as much of the query as possible. Because this setting will change the cost-based decision process in the query optimizer, it is not recommended as the first phase of customization. If your remote data source is as powerful as the federated server, it might make sense to set this option. Setting this option to Y is also useful to compare the performance of your queries. Note that setting this option to Y will affect all queries that reference data from this remote data source.

The federated system provides the SET SERVER OPTION statement for you to use when you want a server option setting to remain in effect while your application is connected to the federated server. When the connection ends, the previous server option setting is reinstated.

Server options are generally set to persist over successive connections to the data source, but can be set or overridden for the duration of a single connection.

In Chapter 10, “Major server options with relational data sources” on page 317, and in Chapter 13, “Major server options with non relational data sources” on page 347, we provide several examples of server options.

## **Remote authentication registration**

In a federated environment the user or application is explicitly connected to and authenticated at the federated server. When the federated server needs to pushdown a request to a remote data source, the server must first establish an under-the-covers connection to the data source on behalf of the federated user. For most of the data sources, the federated server does this by using a valid user ID and password to that remote data source. When a user ID and password are required to connect to a data source, you must define an association between the federated server user ID and the password, and the data source user ID and password. This association must be created for each user ID that will be using

the federated system to send distributed requests. This association is called *user mapping*.

DB2 Information Integrator provides this additional layer of security for all relational data sources, and some non relational data sources. XML files do not require the registration of user mappings. The user ID that runs the federated server opens the XML file on behalf of the federated user, and thus no additional layer of authentication is applied. Here is an example of an SQL statement, which registers the user's mappings for our federated systems:

```
CREATE USER MAPPING FOR user SERVER usa_server
  OPTIONS (REMOTE_AUTHID 'MVSUSER1', REMOTE_PASSWORD
    'password' );
```

## Passthru sessions

You can submit SQL statements directly to data sources by using a special mode called *passthru* or pass through. Submit SQL statements in the SQL dialect used by the data source. After the remote authentication registration step in the section “Remote authentication registration” on page 36, it is possible to test the connection to the data source if *passthru* is supported for the data source. Testing the connection at this stage helps to isolate the configuration problem before you continue to the nickname registration step. The SQL statements and commands submitted in a *passthru* session do not go through the query compiler. DB2 UDB assumes that the *passthru* session includes some write operations against the remote server. As a result, if you have a transaction that contains a *passthru* session to a server, you are not allowed to perform any “write” operations to either local data on the federated server, or remote data on a different server in the same transaction:

```
GRANT PASSTHRU ON SERVER credit_server TO GROUP managers;
```

After the *passthru* privileges are granted, the affected users, such as those belonging to the group managers in the above example, can use the *passthru* facility. The following example initiates the RUNSTATS equivalent operation on the Oracle server:

```
SET PASSTHRU credit_server;
ANALYZE TABLE bad_credit COMPUTE STATISTICS;
SET PASSTHRU RESET;
```

In this example, it is the statement following the SET PASSTHRU statement that causes the federated server to establish the under-the-covers connection to the data source. The statement itself only tells the federated server that the following statements must be sent to the data source, but does not establish the connection. If there is a problem with the server definition *credit\_server*, then an error message will occur in response to ANALYZE TABLE *bad\_credit* COMPUTE STATISTICS.

Note that having the passthru privilege is not enough to access the data source through DB2 Information Integrator. DB2 users are required to have a user mapping set up in order to successfully connect to the specific data source.

The administrator of the federated system controls which DB2 users can open a passthru session to a remote server. The access can be granted using the GRANT PASSTHRU statement.

You can also use a passthru session when you want to perform an operation that is not possible within the DB2 SQL/API. For example, use a passthru session to create a procedure, create an index, or perform queries in the native dialect of the data source. However, the user should be aware that the ability to refer to data at other data sources is lost until you end the passthru session.

**Note:** Currently, the data sources that support passthru support it using SQL. In the future, it is possible that data sources will support passthru using a data source language other than SQL.

## Registering nicknames

After server definitions and user mappings, the federated instance owner creates the nicknames. A *nickname* is an identifier that is used to reference the object located at the data sources that you want to access. The objects that nicknames identify (remote objects such as tables and views) are referred to as *data source objects*.

Nicknames are different from aliases that already exist in DB2. Nicknames are pointers by which the federated server references the remote objects.

For relation nicknames, creating a nickname causes the wrapper to connect to the federated data source and validate the existence of the data source objects, and then to retrieve the column definition and index information. If the statistics maintained on the data source are similar to what is maintained on the federated system, the wrapper nickname registration function looks up and retrieves the statistics information from the remote system catalog. Accurate index and statistics information are fundamental to cost-based decisions in the query optimizer when the nickname is used.

On a federated system, unique index information might be required to perform an UPDATE/DELETE operation for some data sources. If data needs to be stored locally before it is updated, for instance, when the table to be updated is also referenced on the same UPDATE statement in a predicate), a unique index is used to simulate positioned cursors for data sources that do not support the row ID (for example, DB2 family). The information about the column definitions, index, and statistics for a nickname is stored in the DB2 federated database system catalogs.

The registration of nicknames provides location transparency since a nickname looks just like a local DB2 table to the federated server. Since relational data sources usually provide system catalogs about the column definition of an object, the DDL syntax only needs to identify exactly which remote object we are creating a nickname for.

Nicknames are defined with the CREATE NICKNAME statement. The wrapper provides a default mapping between the data types that are used in the data source and the data types that are available in DB2.

When an end user or an application submits a distributed request to the federated server, the request does not need to specify the data sources. Instead, the request references the data source object through the defined nickname. The nicknames are mapped to specific objects at the data source. These mappings eliminate the need to qualify the nicknames by data source names. The location of the data source objects is transparent to the end user or the client application.

The following statement is used to define a relational nickname:

```
CREATE NICKNAME usa.items FOR usa_server.mvsuser1.items;
```

The syntax for defining nicknames for most non relational sources will contain more details. You might need to provide the column definitions when you register a nickname for a source like table-structured flat files. In Example 2-3 we show the statements to create an XML nickname.

---

*Example 2-3 Creating a nickname*

---

```
CONNECT TO FEDDB
CREATE NICKNAME XML.REGION (
  R_REGIONKEY INTEGER NOT NULL OPTIONS(XPATH './r_regionkey/text()'),
  R_NAME CHARACTER (25)NOT NULL OPTIONS(XPATH './r_name/text()'),
  R_COMMENT VARCHAR (152)NOT NULL OPTIONS(XPATH './r_comment/text()'))
FOR SERVER "XMLSERV"
OPTIONS(XPATH '//region',FILE_PATH '/exchange/xml/region.xml')
```

---

Note that the Control Center in DB2 Information Integrator provides an automatic nickname DDL generation facility for XML nicknames, which greatly reduces the mapping complexity.

In Chapter 4, “Configuring DB2 Federated Data support” on page 73, and Chapter 5, “Installing and configuring DB2 Information Integrator” on page 121, we provide several examples of nickname definitions.

In Chapter 9, “Nickname statistics” on page 305, we provide examples of queries with correct statistics.

## Column options

During the registration of a data source object on a relational data source, default type mappings are used to determine how a data source data type is to be mapped to a DB2 type for each column. These mappings are built into the wrapper modules. You might want to change this column data type to a different type to match the data more closely. For example, Oracle DATE type by default is mapped to the DB2 TIMESTAMP type. But if a column of type DATE is only used to store the date of birth of employees, you can map it to the DB2 DATE type.

Also, if the nickname column will be joined with other columns that have the DB2 DATE data type, or will be used with DB2 DATE functions, then the nickname must be altered to change the local type of this column from TIMESTAMP (default nickname column local type for Oracle DATE) to DATE. You can supply the federated database server with additional metadata information about the nicknamed object. This metadata describes values in certain columns of the data source object. You assign this metadata to parameters that are called *column options*. The column options tell the wrapper to handle the data in a column differently from the default behavior. The SQL compiler and query optimizer use the metadata to develop better plans for accessing the data.

With the ALTER NICKNAME statement, you have the option to change the column data type on the federated server for a remote object. You can also use this statement to customize the name of a nickname column, because it is by default set to the remote name of the same nickname column when a nickname is registered.

As in any distributed environment, it is important to allow any remote SQL operations to be performed at the location of the data in order to reduce the amount of data that has to cross the network from the data source to the federated server. Allowing the pushdown of these operations can save the network cost. There are at least two column options that you can set to help the federated server understand the data better in order to encourage more operations to be pushed down. They are VARCHAR\_NO\_TRAILING\_BLANKS and NUMERIC\_STRING.

The *VARCHAR\_NO\_TRAILING\_BLANKS* column option identifies a nickname column that contains no trailing blanks. The query compiler uses this information while checking any character comparison operations to decide the strategy to evaluate the operations. DB2 uses blank padded comparison semantics, i.e. while comparing character strings of unequal lengths; the comparison is made by using a copy of the shorter string, which is padded on the right with blanks so that its length is equal to that of the longer string. This means that the string "A" is considered equivalent to "A " in DB2 UDB. However, this behavior does not apply to all character data types across all data sources, such as the VARCHAR2 data type in Oracle.

In general, comparison operations on string columns without blank padding comparison semantics need to be evaluated locally unless the query compiler is able to find functions to enforce similar logic remotely. For certain operations such as predicates, the federated system maintains performance by rewriting the predicates to ensure the same semantics when these predicates are sent to an Oracle server. Performance of operations such as DISTINCT, ORDER BY, GROUP BY, UNION, column functions (MIN()/MAX()) evaluation, relational comparison and IN predicates might be affected if this column option is set. You can also set this option as a server option if you are sure that all VARCHAR2 columns from this server have no trailing blanks.

In 10.3 “Oracle server option varchar\_no\_trailing\_blanks” on page 327, we provide examples on its influence on queries.

*NUMERIC\_STRING* is another column option that affects performance. This column option applies to character data types and is applicable to those data sources for which the COLLATING\_SEQUENCE server option is set to N. The federated system does not push down any operations that can produce different results by differences in collating sequences at the data source. Suppose that a data source has a collating sequence that differs from the federated database collating sequence. The federated server typically does not sort any columns containing character data at the data source. It returns the data to the federated database, and performs the sort locally. However, suppose that the column is a character data type (CHAR or VARCHAR) and contains only numeric characters (0,1,...9). You can indicate this by assigning a value of Y to the *NUMERIC\_STRING* column option. This gives the DB2 query optimizer the option of performing the sort at the data source. If the sort is performed remotely, you can avoid the overhead of porting the data to the federated server and performing the sort locally. For instance, if we want to indicate that the item\_id column stores only numeric characters, we can indicate this with the following statement:

```
ALTER NICKNAME usa.items ALTER COLUMN item_id
OPTIONS (ADD NUMERIC_STRING 'Y');
```

In Chapter 4, “Configuring DB2 Federated Data support” on page 73, and Chapter 5, “Installing and configuring DB2 Information Integrator” on page 121, we provide several examples of wrapper definitions.

## Index specifications

The *index specification* is a set of metadata catalog information about a data source index. When you create a nickname for a data source table, information about any indexes that the data source table has, is added to the global catalog. The query optimizer uses this information to expedite the processing of distributed requests. During the registration step for relational nicknames, the wrapper modules try to retrieve remote catalog information for the underlying

object. Sometimes, it is not possible to locate the index information when a remote data source might not have a system catalog for indexes, or when the remote object does not appear to have any indexes associated with it in the remote system catalog, such as is the case when nicknames are created for views. You can explicitly define an index specification at the federated server after the nickname is created, if you want to make the federated server aware of the fact that there is an index that can facilitate the access of this remote object. If significant changes at the data source object happen, for instance when you add or drop indexes, they should be reflected in the index specification in DB2. Currently, this is not reflected automatically in the global catalog; the federated instance owner or DBA is responsible for updating this information.

The syntax of the statement to create the index specification is an extension to the CREATE INDEX statement. No physical index is built on behalf of this nickname. Just an entry is added to the system catalog to indicate to the query optimizer that such a remote index exists. This helps the query optimizer in generating remote plans for relational nicknames.

An index specification that defines a unique index also conveys the information about the uniqueness of the index columns to the federated system. Just like a regular unique index definition registered during relational nickname registration, such uniqueness information can help the query optimizer to generate a more optimal plan with strategies such as eliminating unnecessary DISTINCT operations.

In Chapter 14, “Using index specifications” on page 355, we provide an example of queries benefitting from index specifications.

## Data type mapping

The data types at the data source must map to corresponding DB2 data types, so that the federated server can retrieve data from data sources. Some examples of the default data type mappings are:

- ▶ The Informix type DATETIME maps to the DB2 type TIMESTAMP
- ▶ The Informix type BOOLEAN maps to the DB2 type CHAR
- ▶ The Informix type MONEY maps to the DB2 type DECIMAL or DOUBLE

For most data sources, the default type mappings are in the wrappers. The default type mappings for DB2 data sources are in the DRDA wrapper. The default type mappings for Informix are in the INFORMIX wrapper, and so forth.

For some non relational data sources, you must specify data type information in the CREATE NICKNAME statement. The corresponding DB2 UDB data types must be specified for each column of the data source object when the nickname is created. Each column must be mapped to a particular field or column in the data source object. If the field contains only numeric data, you can specify a



numeric data type appropriate for the values in the field as the data type for the nickname column; if the field contains only dates in a valid format, then the DATE type can be specified for the nickname column. For all other instances, a character data type (CHARACTER, VARCHAR) must be specified as the data type for the nickname column.

For relational data sources, you can override the default data type mappings, or create mappings when there is no default.

For examples and details, see Chapter 11, “Using data type mappings” on page 333.

## Function mapping

In addition to the default data type mappings, the wrapper library also contains a set of default function mappings that instruct the query compiler to map a DB2 built-in function (such as SUM) to a relational data source equivalent, if possible. A function mapping is required for a relational data source if you would like the federated server to push down the evaluation of a function to this data source, which can sometimes improve query performance. For relational data sources, you might want to create a function mapping when you want to use a data source function that the federated server does not recognize. The mapping that you create is between the data source function and a DB2 counterpart function at the federated database. Function mappings are typically used when a new built-in function or a new user-defined function becomes available at the data source. Function mappings are also used when a DB2 counterpart function does not exist.

If there is some special data source specific function that is not available on DB2 Information Integrator, a function template definition for that function is needed. A function template is like an extension to a user defined function. It is required to allow queries to reference the functions on the data source that do not have corresponding functions on the federated server. Non relational sources usually have a fixed set of functions that the wrapper recognizes, and thus it is less likely that you will face the need to register a new function template for a non relational data sources.

The CREATE FUNCTION statement has an extension for function templates definitions. For examples and details, see Chapter 12, “Using function mappings” on page 341.

## The global catalog

The DB2 federated database server can hold local data just like any other DB2 instance. The federated database system catalog contains information about the objects (tables, indexes, functions, etc.) in the federated database as well as information about objects (wrappers, remote servers, nicknames, and their

relationships) at the data sources. The information stored is about local and remote column names, column data types, column default values, and index information. The catalog in a federated database is called the global catalog because it contains information about the entire federated system.

The global catalog (see Table 2-3) contains statistical information for nicknames (table and column cardinality), information on remote indexes for nicknames, information on some attributes of each remote source; also it contains type and function mappings.

*Table 2-3 Global catalog contents for remote data sources*

Federated objects	Catalog views	Descriptions
Wrappers	SYSCAT.WRAPPERS SYSCAT.WRAPOPTIONS	Registered wrappers and their specific options. (wraptype='R'/'N' for Relational/Non-relational wrapper)
Servers	SYSCAT.SERVERS SYSCAT.SERVEROPTIONS	Registered remote data sources and their specific options
User mappings	SYSCAT.USEROPTIONS	Registered user authentications for specific servers for a DB2 user. The password setting is stored encrypted.
Nicknames	SYSCAT.TABLES SYSSTAT.TABLES SYSCAT.TABOPTIONS SYSCAT.COLUMNS, SYSSTAT.COLUMNS SYSCAT.COLOPTIONS SYSCAT.INDEXES SYSSTAT.INDEXES SYSCAT.INDEXOPTIONS SYSCAT.KEYCOLUSE	Registered nicknames are identified with TYPE='N' in SYSCAT.TABLES. SYSCAT.TABOPTIONS stores specific options about nicknames. SYSCAT.COLOPTIONS stores specific options about nicknames; for instance, the servername, remote schema, and remote table name. SYSCAT.KEYCOLUSE stores information about primary key.
Index specifications	SYSCAT.INDEXES SYSSTAT.INDEXES	Index specifications created for nicknames

Federated objects	Catalog views	Descriptions
Type mappings	SYSCAT.TYPEMAPPINGS	User-defined type mappings used in nickname registration and transparent DDL. Default built-in type mappings are not stored in these catalog views Mapping direction = 'F'/'R'
Function templates	SYSCAT.FUNCTIONS SYSCAT.ROUTINES	Registered user-defined functions. In V8, SYSCAT.ROUTINES supersedes SYSCAT.FUNCTIONS in V8, but SYSCAT.FUNCTIONS still exists, not documented
Function mappings	SYSCAT.FUNCMAAPPINGS SYSCAT.FUNCMAOPTIONS SYSCAT.FUNCMAAPPARMOPTIONS	User-defined function mappings to map a local function to a remote function.
Passthru privileges	SYSCAT.PASSTHRUAUTH	Authorization to allow users to query a specific server using PASSTHRU.

The information is collected when you implement the steps for configuring your federated system as described in the previous sections. You can verify the information by issuing queries against the catalog.

DB2 query optimizer uses the information in the global catalog and the data source wrapper to plan the best way to process SQL statements. Execution plans for federated queries are chosen by the same DB2 optimizer that optimizes regular queries, the difference is that the federated engine uses the native client interface to each source, and sends queries to it in its own dialect.

## 2.2.4 Replication

Replicating is the copying of data from one place to another. Data can be extracted by programs, transported to some other location, and then loaded at the receiving location. A more efficient alternative is to extract only the changes since the last processing cycle, and transport and apply those changes to the receiving location. Data may be filtered and transformed during replication. There may be other requirements for replication, such as time constraints. In most cases, replication must not interfere with existing applications and must have minimal impact on production systems. The replication processes needs to be managed and monitored. Support for heterogeneous replication is included in the DB2 V8 replication solution.

Figure 2-4 shows the replication architecture and how the server lets administrators manage data placement for performance and availability.

Replication is commonly used to incrementally maintain data warehouses or datamarts to synchronize corporate headquarters systems with branch or retail outlet servers, or with a mobile sales force, or to maintain operational data stores.

The replication server supports movement of data between mixed relational data sources. DB2, Informix Dynamic Server, Microsoft SQL Server, Oracle, Sybase SQL Server, and Sybase Adaptive Server Enterprises are supported as replication sources and targets. Informix Extended Parallel Server and Teradata are supported as replication targets. IMS Data Propagator can capture the changes to IMS data and stores it in the DB2 staging tables on z/OS. Changes or summarized results will be then applied to the DB2 tables as seen by the federated engine.

Data movement can be matched according to the usage requirements of the application. Data movement can be managed table-at-a-time such as for warehouse loading during batch windows, or with transaction consistency for data which is never off-line. It can be automated to occur on a specific schedule, at designated intervals, continuously, or as triggered by events. Transformation can be applied in-line with the data movement through standard SQL expressions and stored procedure execution.

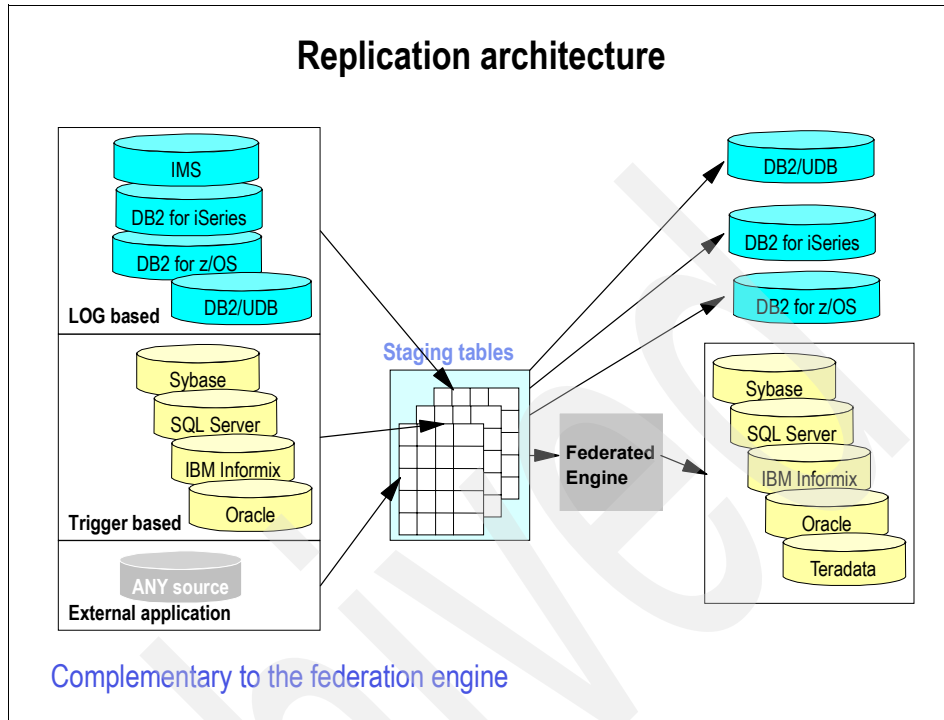


Figure 2-4 Replication architecture

In this redbook we focus on integrating dispersed sources of data. For considerations on functions related to moving or replicating data in order to consolidate the information in one server, please refer to *A Practical Guide to DB2 Data Replication V8*, SG24-6828, and *Moving Data Across the DB2 Family*, SG24-6905.

## 2.2.5 Wrapper development kit

The wrapper development kit (WDK), currently available only in a C++ version, enables you to develop your own wrapper modules. The wrapper module is a shared library with specific entry points, which provide access to a class of data sources. DB2 UDB loads it on demand dynamically. The wrapper module is what you will be developing using the specific classes supplied with DB2. It will contain specific building blocks that allow it to act as a translator between your data source and your federated system. The most important building blocks are shown in Table 2-4.

Table 2-4 Wrapper building blocks

Building block name	Description	Services provided
Wrapper	The code module. It represents a shared library with specific entry points. It is the base for a class of data sources.	Initialization. Access to servers.
Server	It represents a specific data source supported by the wrapper	Access to set of nicknames. Query planning using the Request-Reply-Compensate protocol
Nickname	It represents a specific collection of data at a server	It describes the nickname schema to DB2.
User	End user authentication	It maps DB2 user information to source information. For example user ID and password.
Remote_Connection	It represents DB2's connection to a source	Connect and disconnect to/from source. Transaction management.
Remote_Operation	It represents an active operation at a source: select, insert, update, and delete.	Remote query: read only query of data. Passthru direct session with data source.

The WDK includes a sample that provides the starting point for developing a wrapper with interfaces that the DB2 Federated Server uses, like the interfaces of the DB2 Information Integrator non relational wrappers. The WDK is described in the IBM DB2 Information *Integrator Developer's Guide Version 8*, SC18-7359.

The intended uses for WDK are:

- ▶ To allow to write C++ wrappers to access remote data sources for which no wrapper is available.
- ▶ The kit is not recommended for building relational wrappers, because relational sources require tight integration with the federation engine.

A sample wrapper is available to show how to code a wrapper. The sample wrapper introduces the source code for a sample, and a platform specific makefile. It provides a wrapper for a table structure file with the following properties:

- ▶ Each line in data file maps to a row in a table.

- ▶ Columns in each line are comma separated.
- ▶ Data types supported are: CHARACTER, VARCHAR, INTEGER, DECIMAL and DOUBLE

Archived







## Part 2

# Configuring the federated data solution

In this part we describe the setup and configuration of the DB2 Federated Data solution for the environment that we have selected for this project.

Chapter 3, “The case study” on page 53 contains the description of the platform environments and the data sources represented in our case study.

Chapter 4, “Configuring DB2 Federated Data support” on page 73 provides the details on the definition of our DB2 native functions for federated data support.

Chapter 5, “Installing and configuring DB2 Information Integrator” on page 121 describes the extension to more data sources for our federated data solution, which is made possible by using the IBM DB2 Information Integrator product.

## The case study

In this chapter we introduce and define the platforms, the DB2 system, the data sources, and the application data that we used throughout the project:

- ▶ The federated system

The gradual approach in designing the federated system. Starting with the DB2 instance on AIX, which operates as a federated server, then gradually adding several data sources, and the clients that access the database and data sources

- ▶ Setting up the environment

The details on how we set up the federated environment over several machines and several platforms. Each section adds a new component to the overall data federation solution.

- ▶ The data and the application

The ER data model and the description of the tables that were created on all data sources for later combined query accesses

## 3.1 The federated system

The first step for the database system administrator is to determine what data sources should be included in the federated system and the best way to define it, taking into account the requirements of the current environment and trying to minimize any disruption of service.

As we have seen, a federated system consists of a DB2 instance that operates as a federated server, a database that acts as the federated database, one or more data sources, and clients (users and applications) that access the database and data sources. With a federated system, you can send distributed requests to multiple data sources with a single SQL statement.

We look at:

- ▶ The starting point
- ▶ Using the native DB2 Federated Data support
- ▶ Extending the picture to non IBM databases
- ▶ And finally, adding non relational data sources

### 3.1.1 The base system

Our local system (see Figure 3-1) is the basis for any form of federation.

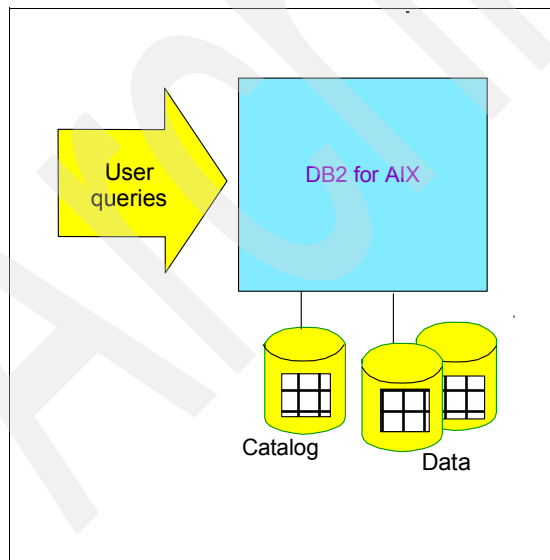


Figure 3-1 DB2 local system

In our environment, we assume that we have a pre-existing DB2 ESE V8.1 installed under AIX 5.1.

### 3.1.2 The DB2 Federated Data

Any DB2 UDB instance, at Version 7 or above, is enabled as a data source for data federation. For the federated server, we used the current version of DB2 UDB ESE V8.1. Federation works on any platform, where DB2 or Informix is available. You can have a DB2 Federated Server on AIX, and connect a DB2 or Informix instance on Windows as a data source, or vice versa. Data sources located on DB2 systems on z/OS or iSeries can be integrated into the federated database server as well.

Applications communicate with the federated server using the supported programming interface. Since a federated system consists of a database, users can also store local data on it, and are able to correlate information across local and remote tables.

Figure 3-2 shows the federated system configuration for our example. A DRDA wrapper provides access to DB2 data sources (on z/OS, Windows, and AIX platforms, and Informix IDS). A library containing the DB2 Connect™ functions is required at the client for the DRDA wrapper being used here. For Informix, the data sources a separate wrapper that needs to be used.

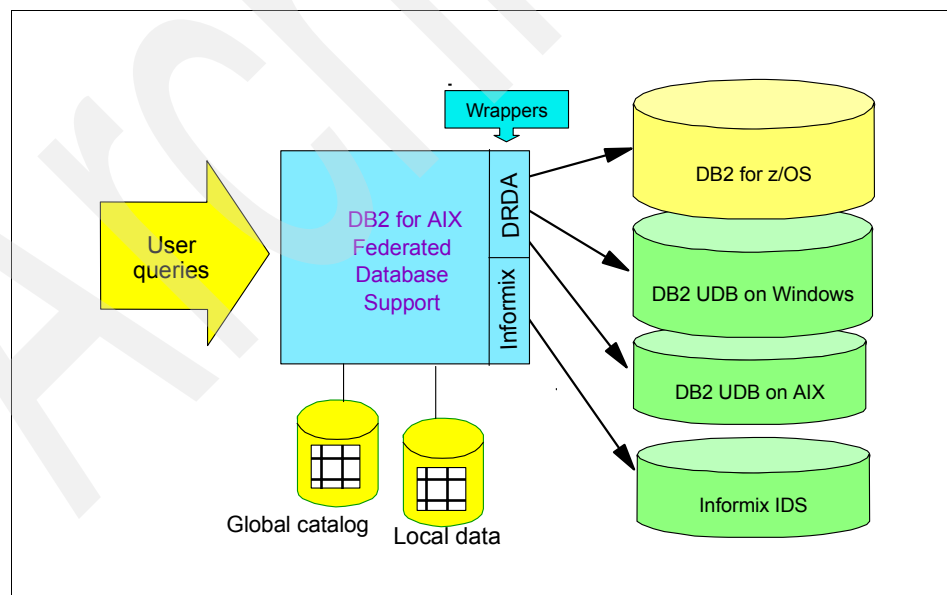


Figure 3-2 DB2 Federated Data

Both these wrappers are already included in DB2 UDB Enterprise Server Edition V8.1. Once all remote objects are registered as nicknames, users with proper privileges can send queries correlating local data, and the data from the four depicted data sources.

### 3.1.3 Information Integrator for relational data sources

We now want to extend the transparent reach of our SQL applications to include other relational sources of data, and we installed DB2 Information Integrator Version 8.1. The pre-existing DB2 instance is enabled for data federation by default in DB2 Information Integrator 8.1. Depending on the wrappers that are enabled in the available DB2 version, applications can access tables located on the federated server and the remote data sources.

Figure 3-3 shows the extended federated system configuration for our example.

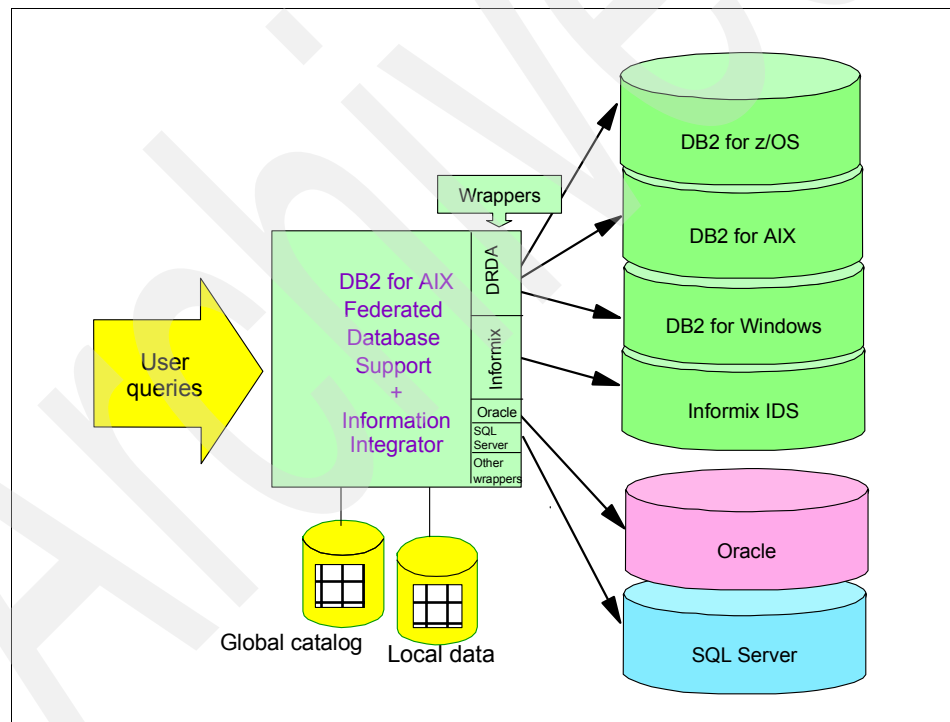


Figure 3-3 DB2 Federated Data and relational data sources

Two additional wrappers, which are installed with Information Integrator, are activated on the federated database server: the Oracle and SQL Server wrapper. Having installed this, you are now ready to access any Oracle or SQL Server

data source on the Windows systems within your network. Besides the two wrappers for Oracle and SQL Server data sources, there are many more wrappers available for different vendors' data sources. Information Integrator relational wrappers are available for Windows (2000, NT, XP), AIX, Solaris, HP-UX, and Linux/Intel (Red Hat).

### 3.1.4 The final picture

As a last step we included in our picture data sources, which are not managed by any data management system, so called non relational data sources like flat files, Excel worksheets, or XML based data. See Figure 3-4.

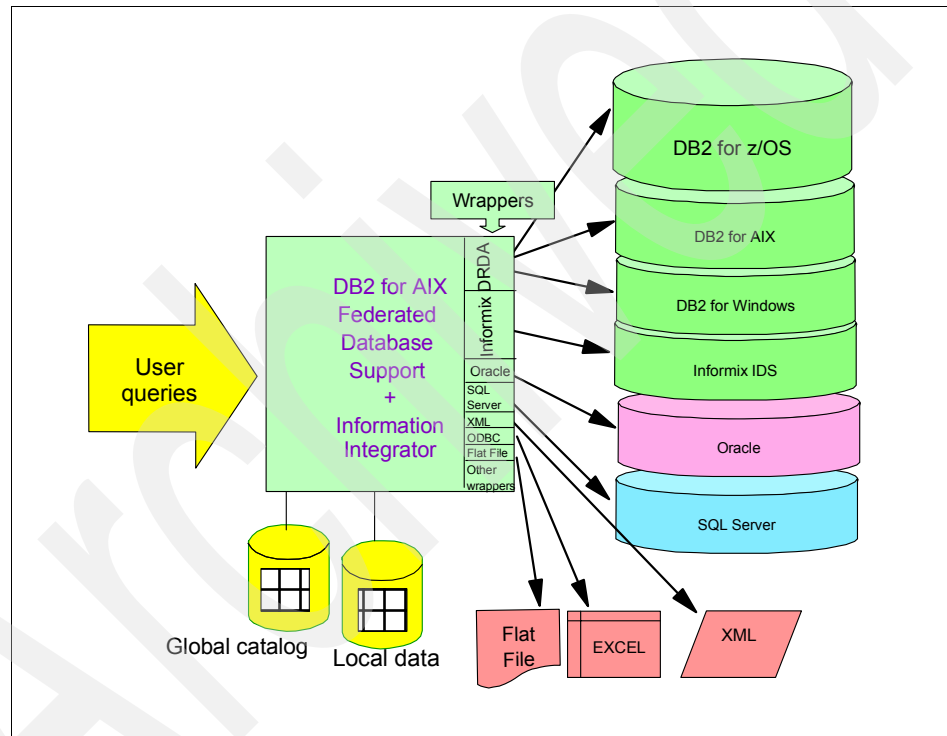


Figure 3-4 DB2 Information Integrator federated data support

The application programmer does not have to worry about implementing the access logic for each data source, the location of data, and the operations to be pushed down to the data source to evaluate; nor one does have to worry about the type of database management system on the data source, the operating system platform of the machine hosting the data source, and the type of network connecting the federated server to the data source. Multiple levels of

transparency in a federated system make the task of programming distributed queries simple.

## 3.2 Setting up the environment

This section describes the setup of a federated environment. Each paragraph involves a major component involved in our entire solution around data federation and information integration.

### 3.2.1 The starting point

The component introduced is DB2 UDB Enterprise Server Edition V8.1 FixPak 2 on the AIX server *baltic*.

We start with a *typical* installation of a DB2 UDB Enterprise Server Edition V8.1 on the host named *baltic*. This IBM RS/6000® machine runs on *AIX 5.1*. DB2 UDB. We set up a 32-bit instance named *db2fed32*. Figure 3-5 gives a graphical illustration of this step.

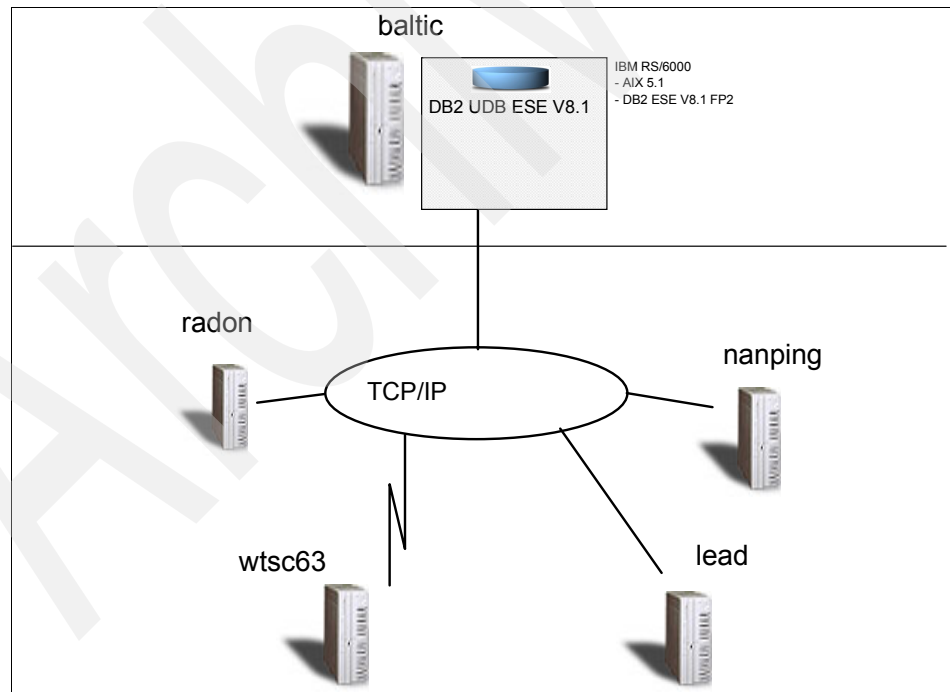


Figure 3-5 Our environment - The starting point



### 3.2.2 Connecting DB2 for z/OS and Informix data sources

The components introduced here are DB2 for z/OS V7 on the zSeries® *wtsc63*, Informix Dynamic Server 9.4 on AIX server *nanping*, Informix Client SDK 2.81 on *baltic*, DB2 DRDA data source support on *baltic*, Informix data source support on *baltic*.

We started using federation, by including DB2 for z/OS and Informix data sources into our picture. DB2 UDB ESE is packaged including federation technology to connect to any DB2 DRDA and Informix data sources.

DB2 DRDA libraries are installed by default during installation of DB2 UDB. To install the Informix libraries, choose the custom installation of DB2 UDB.

We installed a DB2 for z/OS instance and connected the federated data server *baltic* with the DB2 instance on *wtsc63*.

Next, we installed an Informix IDS 9.4 server on *nanping*, running on AIX 5.1 and set up the instance and database. The Informix client software does not come with DB2 ESE, only the Informix wrapper that uses it. To connect *baltic* with the Informix instance on *nanping*, install the Informix Client SDK 2.81 on *baltic*. Figure 3-6 gives a graphical illustration of this step.

Please refer to Chapter 4, “Configuring DB2 Federated Data support” on page 73 for more details.

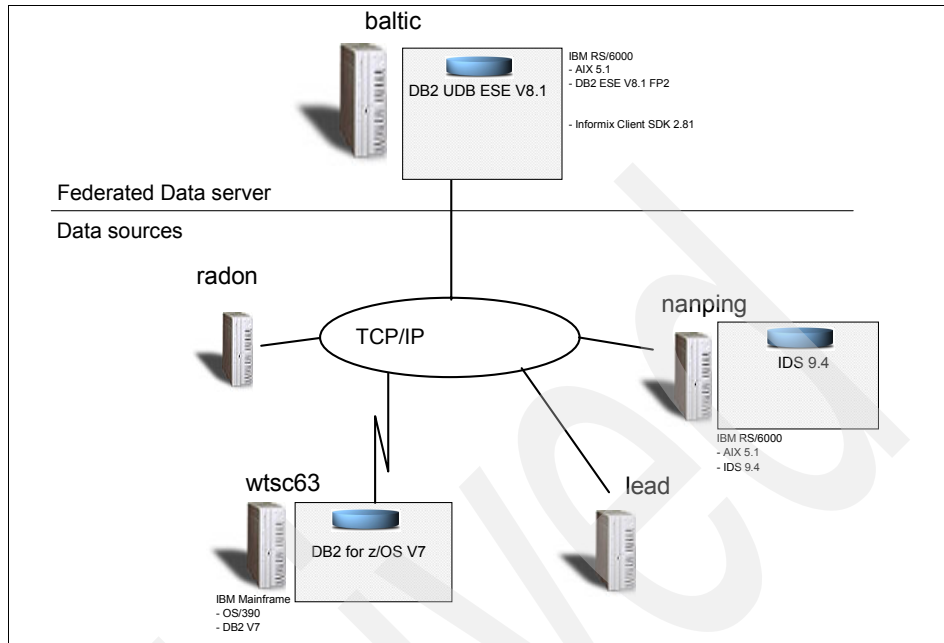


Figure 3-6 Our environment - Adding DB2 for z/OS and Informix data sources

### 3.2.3 Installing Information Integrator

The component introduced here is DB2 Information Integrator V8.1 FixPak 2.

So far, we have used DB2 functionality to use federation within the IBM product family. Now, we want to build the connection to other data sources, relational and non relational, by introducing the Information Integrator product. This product needs to be installed on top of a DB2 UDB ESE V8.1 installation. If you do not have DB2 already installed, the installation process of Information Integrator will do this for you. Figure 3-7 gives a graphical illustration of this step.

We installed DB2 Information Integrator V8.1 FixPak 2 on baltic, including relational data source support for Oracle, Microsoft SQL Server, and ODBC data sources, and non relational data source support for structured file data sources and XML.

Please refer to Chapter 5, “Installing and configuring DB2 Information Integrator” on page 121 for more details.

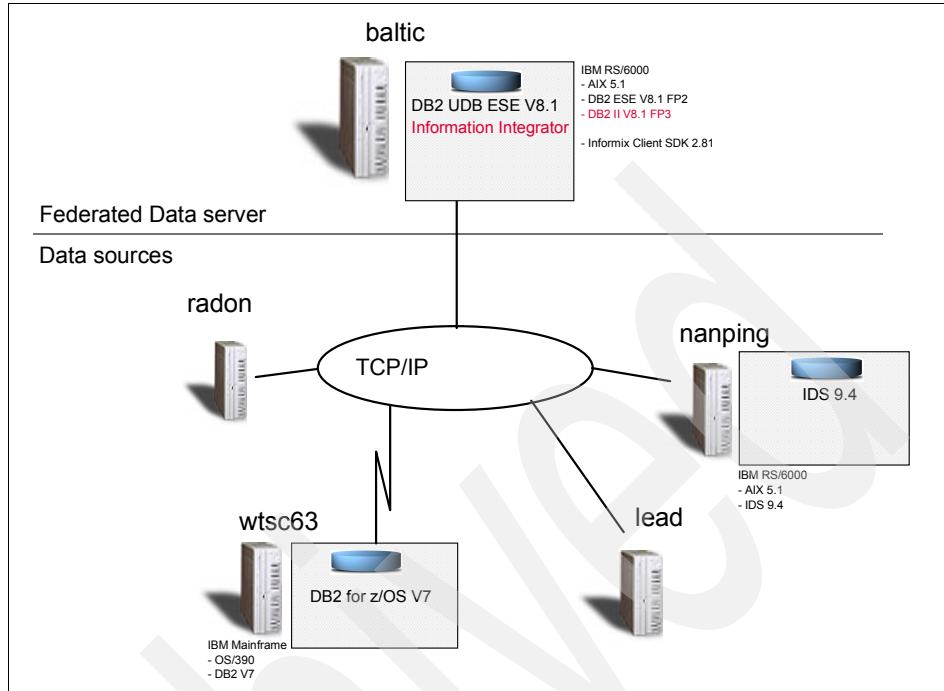


Figure 3-7 Our environment - Installing Information Integrator

### 3.2.4 Connecting Oracle data source

The components introduced here are Oracle 9i server on the Windows server lead, Oracle Client Software on baltic (net8), and Oracle data source support on baltic.

We began using the functionality of Information Integrator with connecting the Oracle data source. Figure 3-8 gives a further graphical illustration. The first step is to install Oracle server version 9i (9.2.0.1.0). In our picture we placed the Oracle installation onto the Windows Workstation lead. To allow connectivity from Information Integrator to the Oracle server, place Oracle Client for AIX software on baltic.

Information Integrator data source support for Oracle utilizes Oracle Call Interface Version 8 (OCI 8) protocol. Please refer to 5.4, “Integrating Oracle 9i” on page 168 for further details on using Oracle server as a data source in Information Integrator.

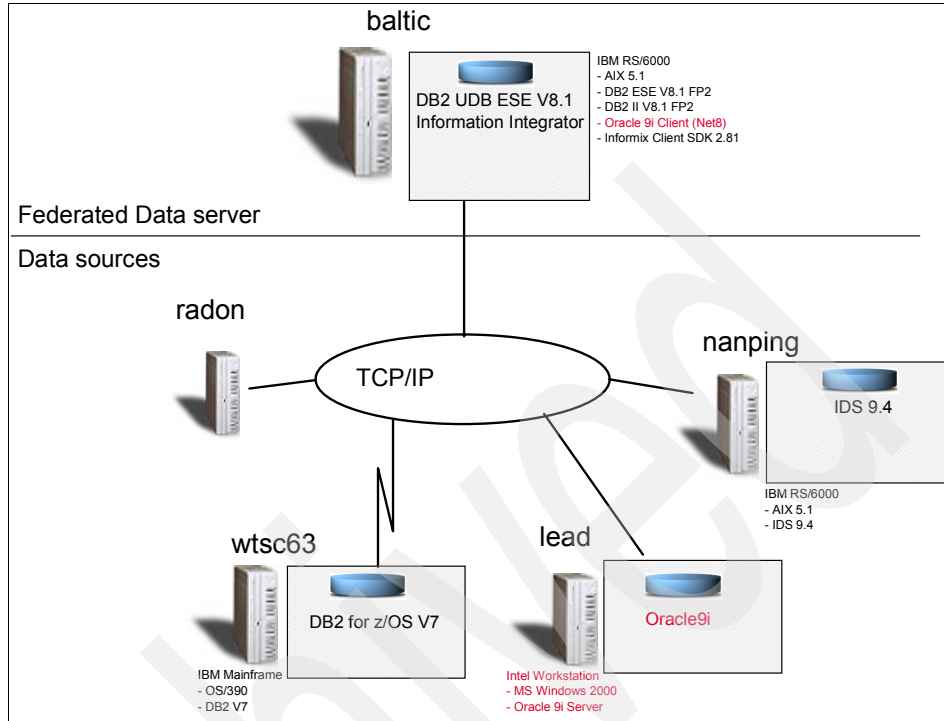


Figure 3-8 Our environment - Connecting Oracle data source

### 3.2.5 Connecting Microsoft SQL Server data source

The components introduced here are Microsoft SQL Server on the Windows server **radon**, and SQL Server data source support on **baltic**, using Data Direct ODBC driver 4.2 for UNIX on **baltic**.

Microsoft SQL Server is placed on the Windows workstation **radon**. Information Integrator uses SQL Server data source support, the technology used behind is actually based on ODBC. We needed to place an ODBC driver on our Information Integrator machine. Information Integrator uses the configured ODBC environment on **baltic** to connect to our SQL Server on **radon**. Please refer to 5.5, "Integrating Microsoft SQL Server 2000" on page 181 for further details on using Microsoft SQL Server as a data source.

For UNIX based information integrator solutions, we recommend using Data Direct Technologies Connect ODBC Version 3.7, or later. For Windows based information integrator solutions, use the Microsoft SQL Server Client Version 2000 driver.

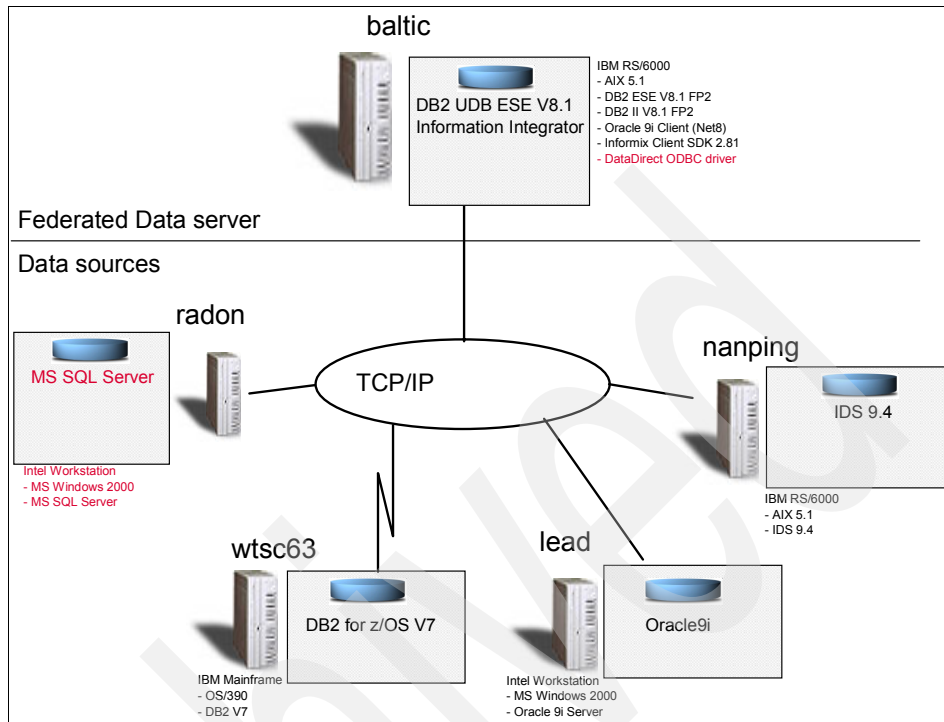


Figure 3-9 Our environment - Connecting Microsoft SQL Server data source

### 3.2.6 Connecting XML data source

The components introduced here are XML data source support, and Samba server on AIX.

With the XML data source, we connected to the first non relational data source. Non relational means in this context that no data management system controls the access of the data. Information Integrator accesses the data source directly; that is, it opens the XML file and reads its contents. The XML file has to be available in the local or any NFS filesystem (for AIX). Figure 3-10 shows basically two (or theoretically several) possible sources for XML files. One of course is the local machine *baltic*. Further locations may be other AIX machines, i.e. *nanping* having exported the fleshiest, and *baltic* having shared this through the Network File System (NFS).

For convenient exchange of files between AIX and Windows, we used Samba software ([tap://www.samba.org](http://www.samba.org)) on AIX enabling us to map a network drive on Windows to an AIX filesystem. With this mechanism, it is easy to create and edit your XML files on a Windows environment and actually save it on UNIX.

Please refer to 5.6, “Integrating XML data source” on page 195 for further details on using XML files as a data source.

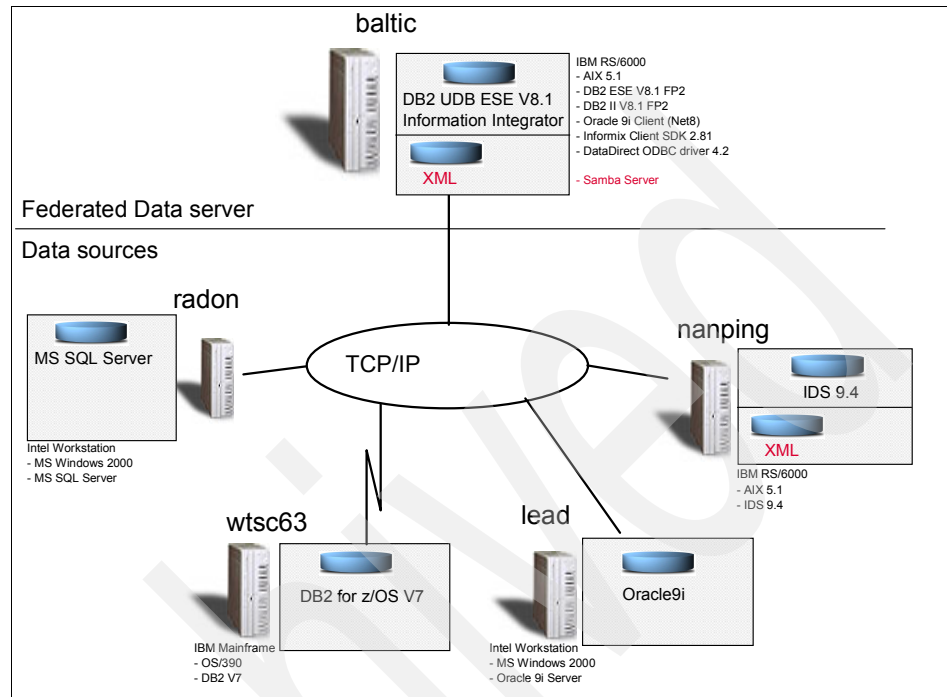


Figure 3-10 Our environment - Connecting XML files

### 3.2.7 Connecting flat file data source

The component introduced here is *flat file* data source support.

Using flat file data sources, we might deal with the easiest and at the same time most flexible form of the data source, the text file. Information Integrator, as with the XML file, has to access the data file in the local or any NFS filesystem. Figure 3-11 shows this. Flat files consist basically of one or many rows; these rows may be delimited into one to many columns by any character delimiter.

**Important:** Do not use the delimiter symbol at the end of a row. If you do, it will result in a failure.

For convenient exchange of files between AIX and Windows, we used Samba software (<http://www.samba.org>) on AIX enabling us to map a network drive on Windows to an AIX filesystem. With this mechanism it is easy to create and edit your flat files on a Windows environment, and actually save it on UNIX.

Please refer to 5.7, “Integrating table-structured files” on page 209 for further details on using Microsoft SQL Server as a data source.

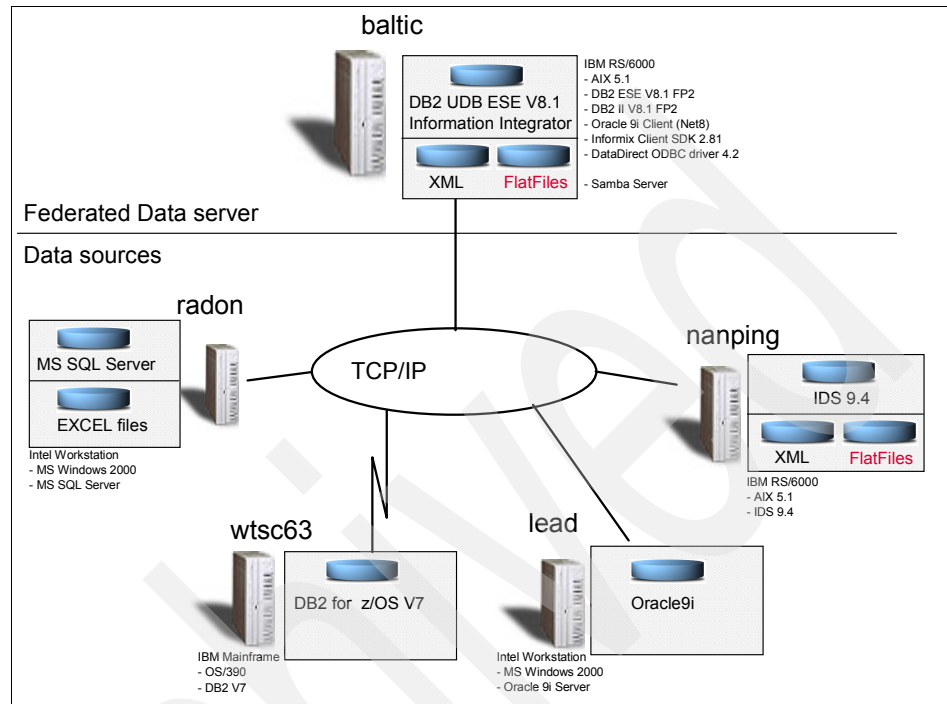


Figure 3-11 Our environment - Connecting flat files

### 3.2.8 Connecting Microsoft Excel data source

The component introduced here is ODBC data source support, using the OpenLink MultiTier ODBC driver.

We used the Intel workstation radon as the source for Excel files. The connection from our AIX federated server baltic to the Windows based radon is established through OpenLink MultiTier ODBC driver. Radon is seen as the ODBC server, baltic needs the configuration of the OpenLink ODBC client. From Information Integrator point-of-view, we used the ODBC data source support to access the Excel files. Information Integrator also offers an individual Excel support driver, which is basically designed to be used on a Windows environment.

We need OpenLink because the Excel file must be read either by Excel or by the Excel ODBC driver, both of which only run on Windows. OpenLink server, collocated with the Excel file and the Excel ODBC driver on the Windows server can read the file for us. We still needed to install the OpenLink client for AIX on

our AIX federated server, and configure the Information Integrator ODBC wrapper to use the OpenLink client to reach the OpenLink server on Windows, which will read the file through the Excel ODBC driver.

Please refer to 5.8, “Integrating Microsoft Excel” on page 218 for further details on using Microsoft SQL Server as a data source.

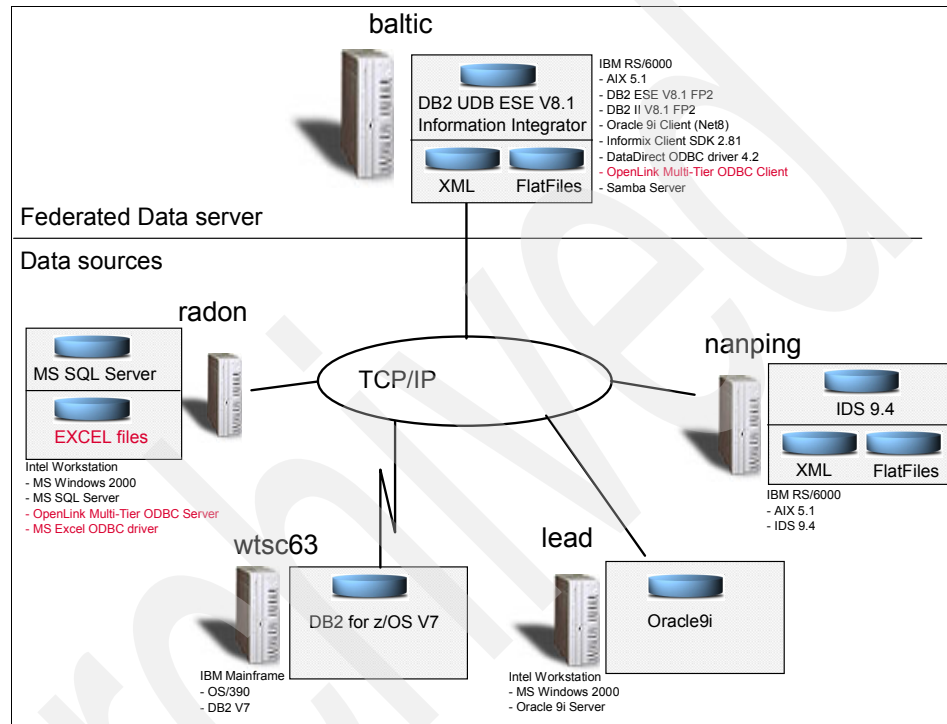


Figure 3-12 Our environment - Connecting Excel files

### 3.2.9 System environment: Summary

Our overall system environment layout approach, as shown in Figure 3-13, is based on the DB2 Federated Server *baltic*. Around this platform we placed all other data sources.

In total we have five machines included in our environment. All machines are connected through the local area network, including a mainframe server, which is in fact placed physically at a different location, but accessible with a local address.



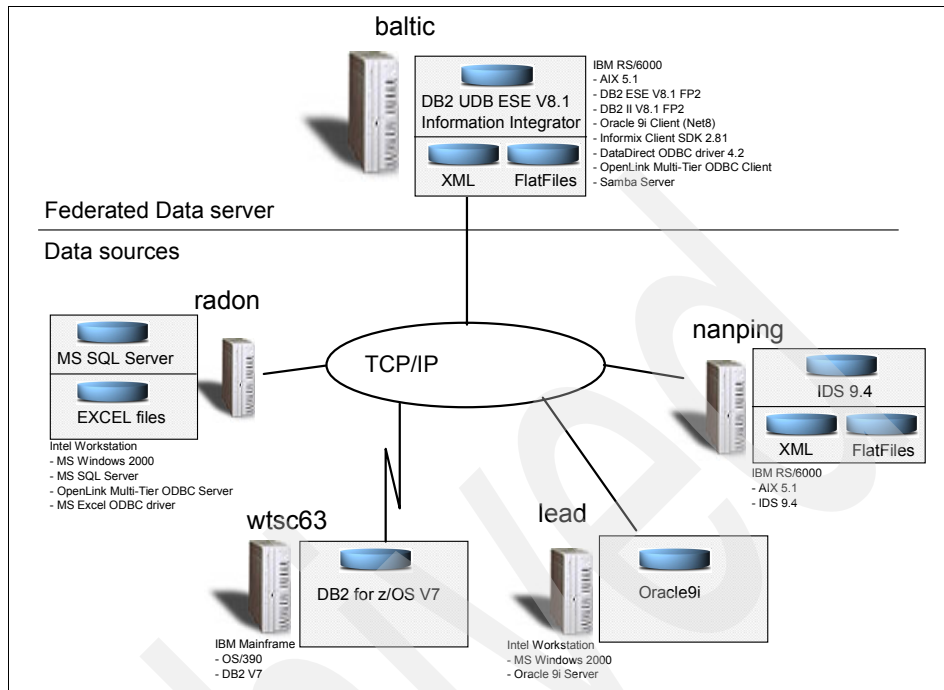


Figure 3-13 Project data sources

The configuration of the machines is as follows:

► **baltic**: IBM Model RS/6000, running on AIX 5.1

This server represents the DB2 Federated Server. The following software components are installed:

- DB2 UDB Enterprise Server Edition V8.1 FixPak 2
- DB2 Information Integrator V8.1 FixPak 2
- Oracle 9i Client software V9.2.0.1.0
- Informix Client SDK 2.81
- DataDirect ODBC driver for UNIX 4.2, available from:  
<http://www.datadirect.com>
- OpenLink MultiTier ODBC Client 5.0, available from:  
<http://www.openlink.co.uk>
- Samba Server 2.2:  
<http://www.samba.org>

**baltic** represents the only interface to client connections.

► **nanping**: IBM Model RS/6000, running on AIX 5.1

This data source server holds the Informix Dynamic Server 9.4 installation. *nanping* can be used for an optional source server of XML files and flat files.

- ▶ **lead**: Intel Workstation, running on Windows 2000

This data source server holds the Oracle 9i server installation.

- ▶ **radon**: Intel Workstation, running on Windows 2000.

This data source server holds the Microsoft SQL Server installation. Additionally, it serves as the source to the Microsoft Excel data. The following software products are installed:

- Microsoft SQL Server 2000
- OpenLink MultiTier ODBC Server 5.0
- Microsoft Excel ODBC driver (comes with Windows)

- ▶ **wtsc63**: IBM mainframe running on z/OS. This data source server holds the DB2 for z/OS subsystem.

## 3.3 The data and the application

A subset of an industry standard benchmark for decision support applications was used. See Appendix A, “Case study” on page 375 for details on the data definition language statements used to create tables and indexes.

### 3.3.1 Data model

The ER model we worked with consists of entities and relationships shown in Figure 3-14.

First some tables were independently created on all data sources in order to verify installation and connectivity, then a design of a database distributed across the data sources was derived, and several queries involving one, or many, or all data sources were executed.

A total of 1 GB of data, corresponding to about 9 million rows was installed on every data source, which enables us to create different scenarios with distributed access onto different data sources.

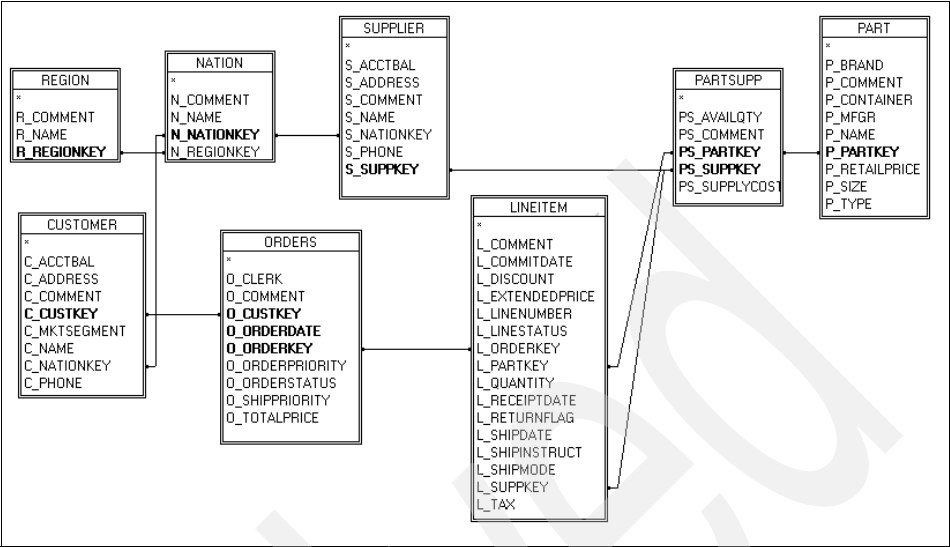


Figure 3-14 Data model

See Table 3-1 for information on the database size.

Table 3-1 Database size for each data source

Table Name	Row cardinality	Row size (bytes)	Approximate table size (MB)
SUPPLIER	10,000	159	2
PART	200,000	155	30
PARTSUPP	800,000	144	110
CUSTOMER	150,000	179	26
ORDERS	1,500,000	104	149
LINEITEM	6,001,215	112	641
NATION	25	128	< 1
REGION	5	124	< 1

For more details on these scenarios, see Part 4, “Exploiting performance options” on page 291, where we show several examples of tuning federated queries.

### 3.3.2 Data distribution

The tables are distributed across the data sources as listed in Table 3-2. We placed our entire data model onto the relational data sources like on a DB2 UDB database, DB2 for z/OS, Informix Dynamic Server 9.40, Oracle 9i, and the SQL Server database. We have not distributed all the tables available in our data model across the non relational data sources (table structured files, XML, and Excel).

Table 3-2 Distribution of tables across the data sources

Data source Table name	DB2 UDB	DB2 for z/OS	Informix	Oracle	SQL Server	Flat files	XML	Excel
CUSTOMER	X	X	X	X	X	X		X
LINEITEM	X	X	X	X	X			
NATION	X	X	X	X	X	X	X	X
ORDERS	X	X	X	X	X			
PART	X	X	X	X	X	X		X
PARTSUPP	X	X	X	X	X			X
REGION	X	X	X	X	X	X	X	X
SUPPLIER	X	X	X	X	X	X	X	X

From the federated server perspective, all the tables on our data sources are accessible with the schema names listed in Table 3-3. It is generally a good idea to use the same names for nicknames and data source tables. Another recommendation at this point is to use the same schema name for all tables or views coming from the same data source, to maintain manageability of your entire federated environment.

Table 3-3 Nickname schema for data sources

Data Source	Schema name for nicknames
DB2 UDB 64-bit Instance	DB2DAT
DB2 for z/OS Instance	DB2ZOS
DB2 for iSeries Instance	DB2OS400
Informix Dynamic Server 9.4 Instance	INFX
Oracle 9i Instance	ORA

Data Source	Schema name for nicknames
Microsoft SQL Server Instance	MSSQL
Table structured file sources	FLAT
XML sources	XML
Excel spreadsheet sources	XLS

### 3.3.3 Data access

Throughout this project, we accessed data from our federated environment in a read only mode. In all our examples starting at Part 4, “Exploiting performance options” on page 291, use read-only access to the data sources.

In our environment, the federated server does not contain any data. In all queries we performed target tables on our remote sources. Instead of putting data in our federated server, we only maintained a set of wrappers, servers, and nicknames. An advantage of this approach is to have a clear separation of actual data stored in one database, and the central administration of links to this data.

In our environment, as far as updating and loading the data is concerned, this was done directly at the respective data source.

The only exception is when we used the federated server as a data store for caching materialized query tables.



# Configuring DB2 Federated Data support

In this chapter we describe the functions natively available within DB2 for federated data support. We show step-by-step how to set up a federated system using DRDA connectivity across most members of the DB2 family (DB2 for AIX, Windows, z/OS, iSeries) and Informix Dynamic Server.

We assume you already have a DB2 UDB V8 instance installed in your AIX 5.1 environment, and that you are now interested in understanding the possibilities of integrating your data by extending access to the data sources that DB2 directly supports. We describe our use of the DB2 Control Center (CC) on Windows to perform the operations, and additionally we provide the explicit command line syntax at the end of each performed operation.

The chapter is structured in the following sections:

- ▶ Background information
- ▶ Setting up the DB2 UDB instance on AIX
- ▶ Integrating DB2 UDB for z/OS
- ▶ Integrating Informix Dynamic Server
- ▶ Integrating DB2 UDB for iSeries

## 4.1 Background information

A wrapper is the mechanism by which the federated server interacts with the data sources. The DRDA and Informix wrappers are part of the base DB2 product. The other wrappers (non IBM relational and non relational) are part of DB2 Information Integrator.

By using wrappers you can send distributed requests to multiple data sources within a single SQL statement. For example:

```
SELECT c.cust_name, o.order_number
FROM db2_cust_table c, infx_order_table o
WHERE c.cust_id = o.cust_id;
```

The information on wrapper functions and the availability across the platforms is listed in *IBM DB2 Information Integrator Federated Systems Guide Version 8*, SC18-7364. This is also true for nicknames and data types.

More recent question and answer type of information relating to the most current FixPak is contained in the *IBM DB2 Information Integrator Installation Release Notes Version 8*, available in softcopy only from the following Web site:

<http://www.ibm.com/software/data/integration/solution>

## 4.2 Setting up the DB2 UDB instance on AIX

The installation of DB2 Information Integrator that we describe here is executed under AIX 5.1 with DB2 UDB ESE V8.1 FP2. For installation on different operating systems and different versions of DB2 UDB, please refer to the manual *IBM DB2 Information Integrator Installation Guide Version 8*, SC18-7036. If you are interested in the procedure to install DB2 Information Integrator under Windows, you can also refer to the redbook *IBM Informix: Integration Through Data Federation*, SG24-7032.

In this section we look at:

- ▶ Pre-installation requirements
- ▶ Installation instructions
- ▶ DB2 database configuration for federation



## 4.2.1 Pre-installation requirements

The hardware and software requirements for the installation are as follows:

### Hardware

- ▶ Memory and disk space sufficient to satisfy DB2 minimum requirements
- ▶ Approximately 30 MB of free disk space for all wrappers (relational and non relational)

### Software

You must install the following software products before you start building your federated system:

- ▶ DB2 UDB ESE V8.1 FP 2

#### Important:

You need to choose **Custom Installation** in order to be able to select the installation of the Informix wrapper libraries.

After installing the Informix wrapper libraries, you need to rerun the installation routine for DB2 UDB ESE (**db2setup**), and then choose:

1. **Install additional features**
2. **Custom**

You can now select the features to install:

- ▶ **Server Support - Informix data source support**

- ▶ DB2 Connect Enterprise Edition V8.1, for access to DB2 for z/OS and DB2 for iSeries. DB2 ESE V8.1 includes DB2 Connect V8.1, so no additional software is required to get this functionality. If DB2 ESE 8.1, which comes with DB2 Information Integrator is installed, the DB2 Connect 8.1 functionality is included, but there is a licensing restriction; only five concurrent users of the DB2 Connect functionality are included with DB2 Information Integrator.
- ▶ Informix Client SDK 2.81, for Informix Dynamic Server (IDS) 9.4. This is not included with DB2 ESE Version 8.1, and must be installed separately.
- ▶ We assume that the DB2 instance is available (in our case db2fed32). Please refer to Chapter 3, “The case study” on page 53 for its description.

## 4.2.2 32-bit or 64-bit DB2 ESE installation and instance

The DB2 ESE instance, the Information Integrator wrappers, and the data source client software need to be either all 32-bit or all 64-bit. There cannot be a mixing of a 64-bit DB2 ESE instance with either 32-bit wrappers or 32-bit data source client software.

In our case, we are required to use a 32-bit DB2 ESE instance because:

- ▶ The DB2 Information Integrator Relational wrapper on AIX for Microsoft SQL Server, and the Data Direct Connect product on AIX are only available in 32-bit mode.
- ▶ The DB2 Information Integrator non relational wrapper on AIX for accessing XML data is only supported in 32-bit mode.
- ▶ The DB2 Information Integrator relational wrapper on AIX for accessing the ODBC data source, which we use with the OpenLink client to access the OpenLink server on Windows to access the Excel spreadsheet, is only available in 32-bit mode.

## 4.2.3 Installation instructions

In this section we describe the installation of the DB2 Instance, DB2 Information Integrator, the DRDA wrapper for the DB2 family of products, and Informix.

The steps to start DB2 installation are:

1. Log in as root.
2. Open an X window.
3. `cd /usr/opt/db2_08_01/instance`
4. If you do not work directly on the console, think of setting the *DISPLAY* environment variable.
5. `./db2isetup`
6. Introduction:

See Figure 4-1 for the introduction screen. This screen describes the possibilities you have with this setup program. The main task you can achieve with this function is creating or setting up an existing instance.

Click **Next** to continue.

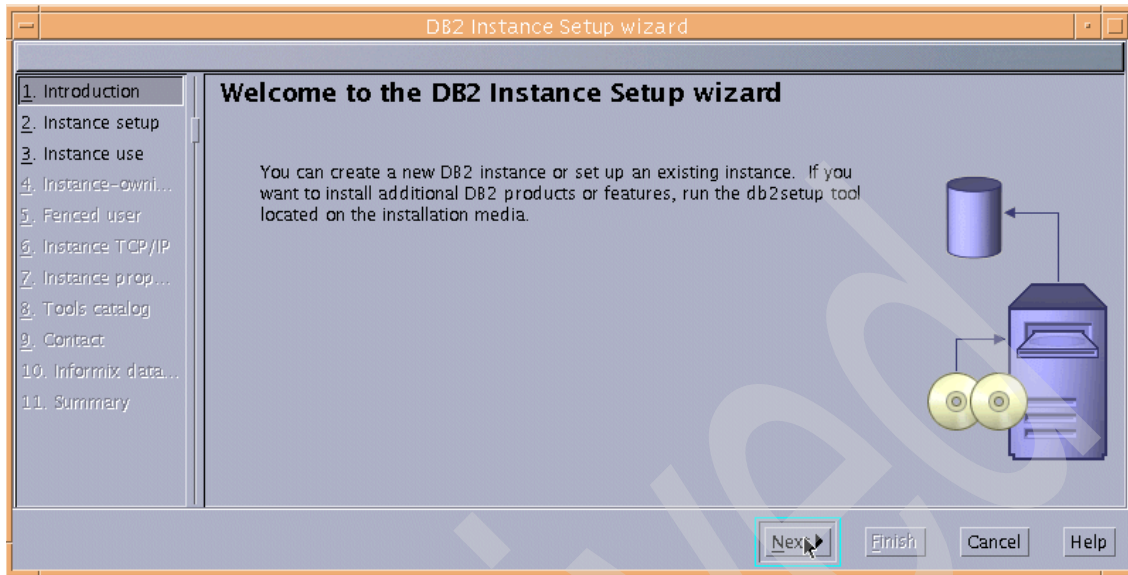


Figure 4-1 DB2 Instance setup - Introduction

#### 7. Instance setup

Figure 4-2 allows you to choose between different possibilities to set up a DB2 instance. Either you want to create a new instance (32-bit or 64-bit), or you want to set up an existing instance. For the existing instance, you need to choose a name out of the list box named **Instance name**. When you create a new instance for federation, you want to check if your DB2 instance supports all the wrappers you wish to use. Please refer to Table 5-1 on page 124 for more details.

Click **Next** to continue.

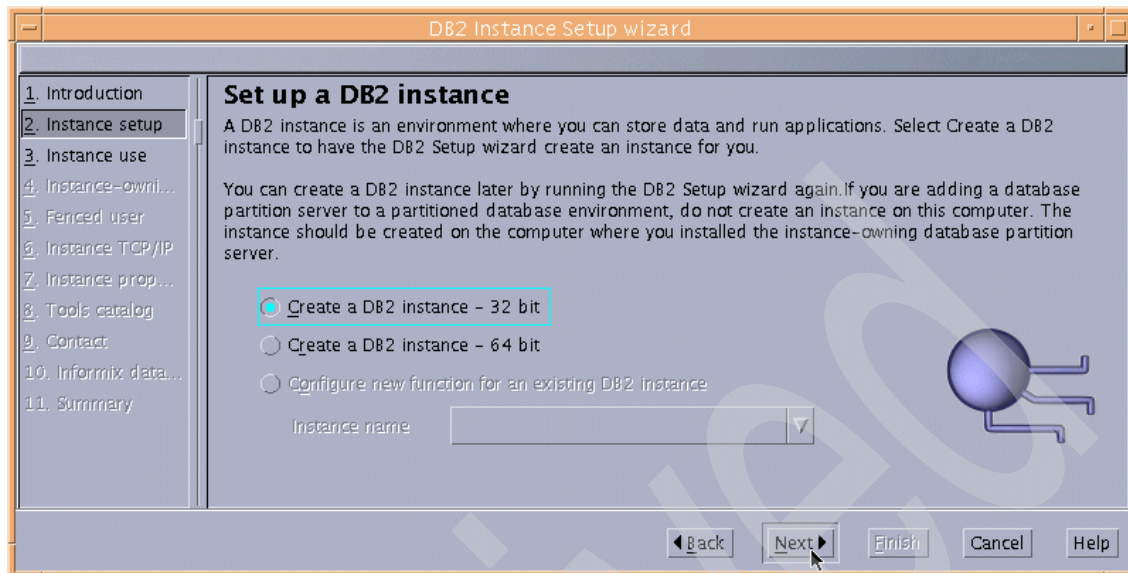


Figure 4-2 DB2 Instance setup - Instance setup

#### 8. Instance use - Single partition instance - Next

In Figure 4-3, you select if you want to use a single partition instance or a partitioned instance. We continue by selecting the single partition instance. For more details on partitioned instances, please consult the standard DB2 UDB documentation.

Click **Next** to continue.

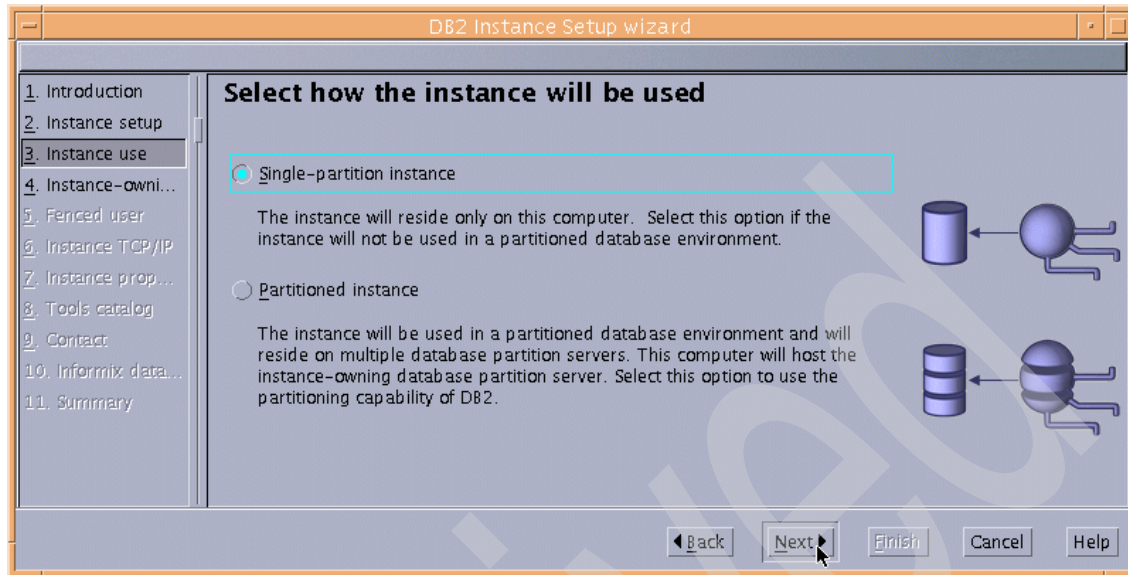


Figure 4-3 DB2 Instance setup - Instance usage

#### 9. Instance-owning user

In Figure 4-4 you need to choose whether to create a new instance owning user, or to use an existing one. This user will be the base for all instance related functions of DB2.

Click **Next** to continue.

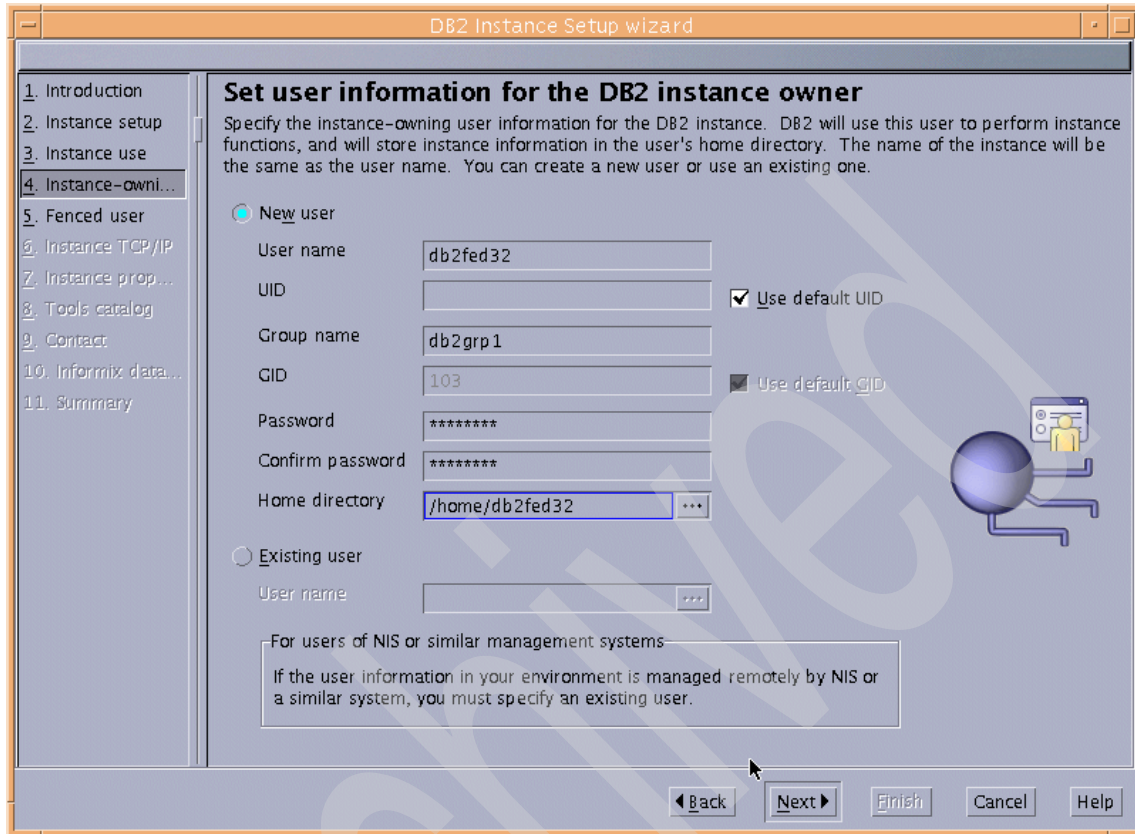


Figure 4-4 DB2 Instance setup - Instance owning user

#### 10.Fenced User

In Figure 4-5, you need to choose whether to create a new fenced owning user, or to use an existing one. DB2 will run fenced user defined functions and stored procedures under this user.

Click **Next** to continue.

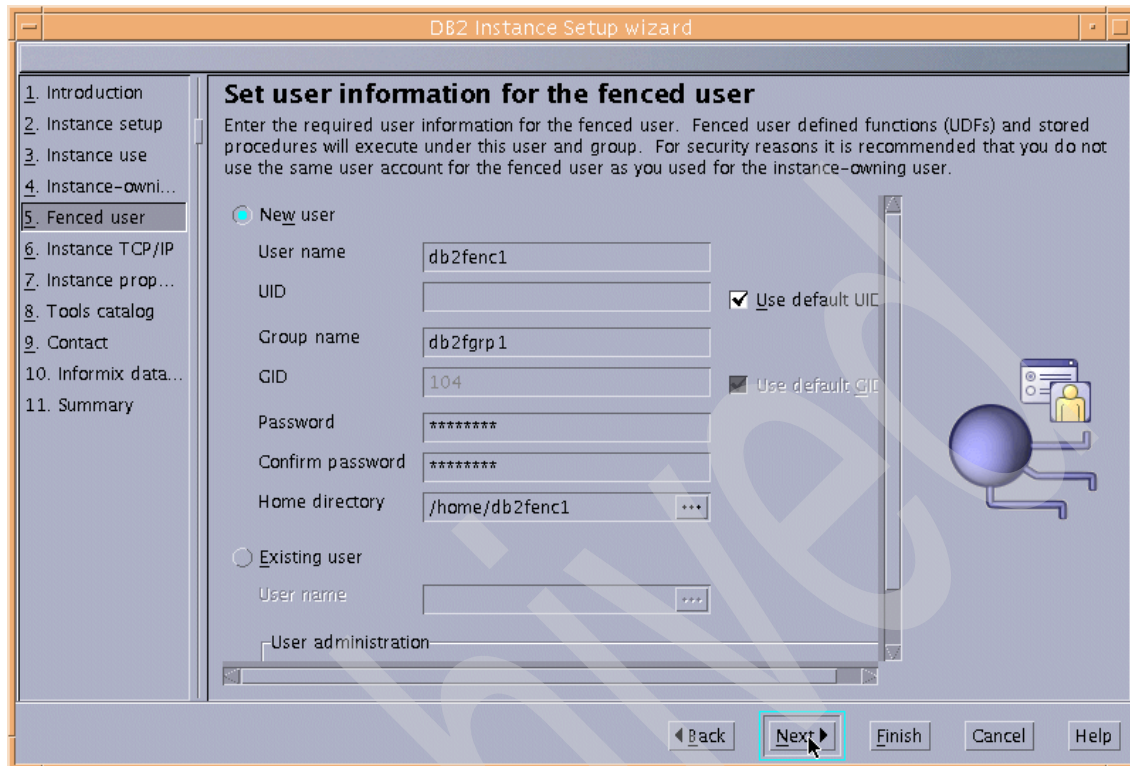


Figure 4-5 DB2 Instance setup - Fenced user

#### 11. Instance TCP/IP

See Figure 4-6 for the TCP/IP settings. You can either defer the DB2 network setting, or specify a service name and the respective port number. Make sure you communicate these setting choices to your client, since this setting enables the clients to access the DB2 instance.

Click **Next** to continue.

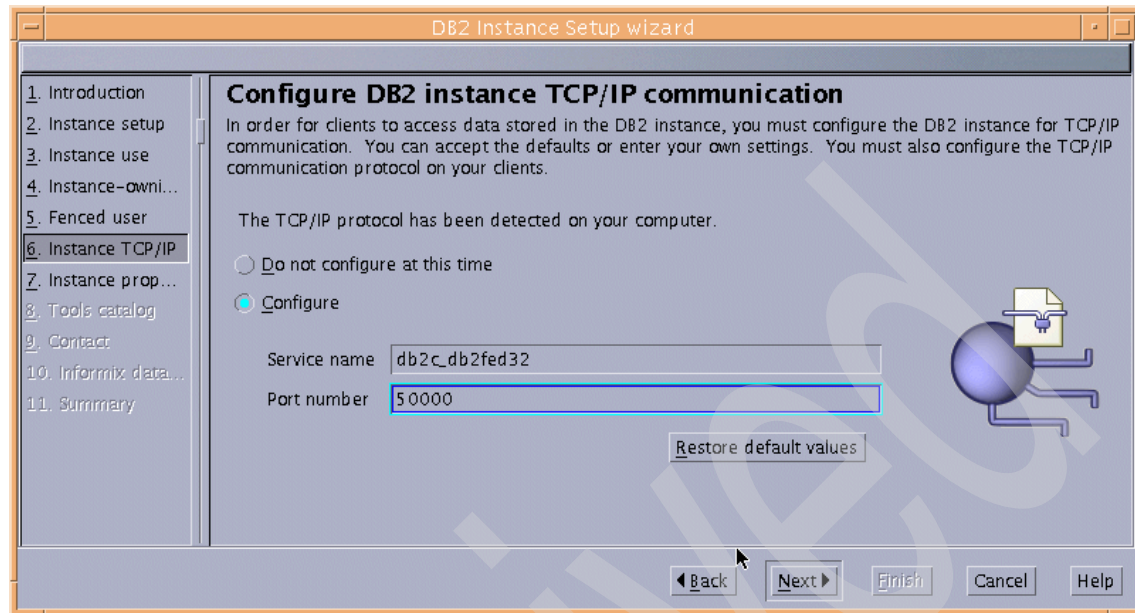


Figure 4-6 DB2 Instance setup - Instance TCP/IP

## 12. Instance properties

See Figure 4-7 for two instance parameters. The database instance authentication type is by default set to server, using the local operating system security.

Click **Next** to continue.



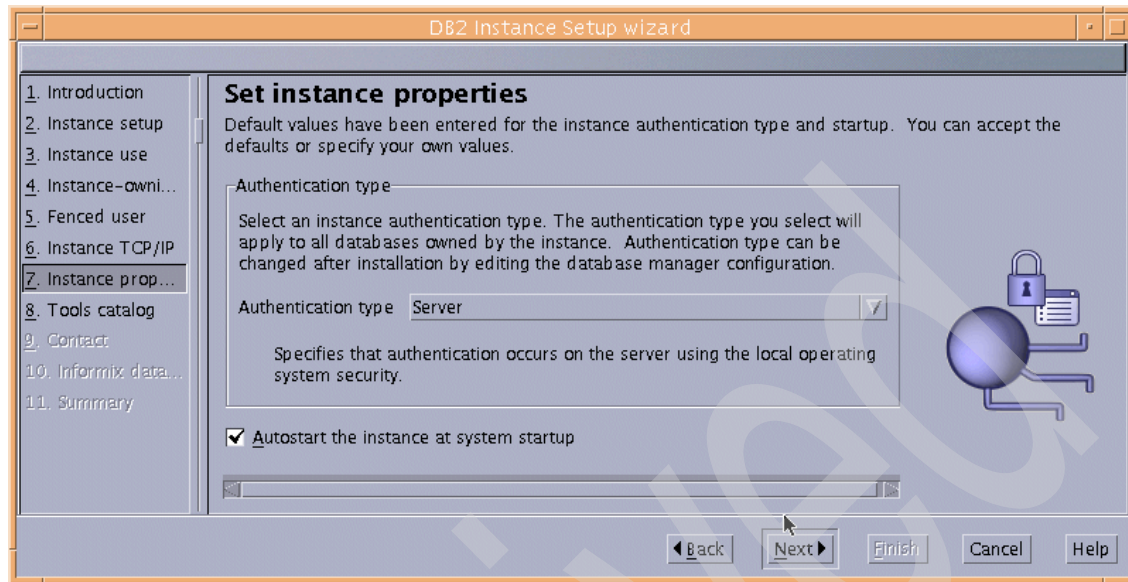


Figure 4-7 DB2 Instance setup - Instance properties

### 13. Tools catalog

In Figure 4-8 you need to select whether you want to prepare the *DB2 tools catalog* or not. To enable you to use some tools like the task center and scheduler, you have to prepare the catalog first.

Click **Next** to continue.

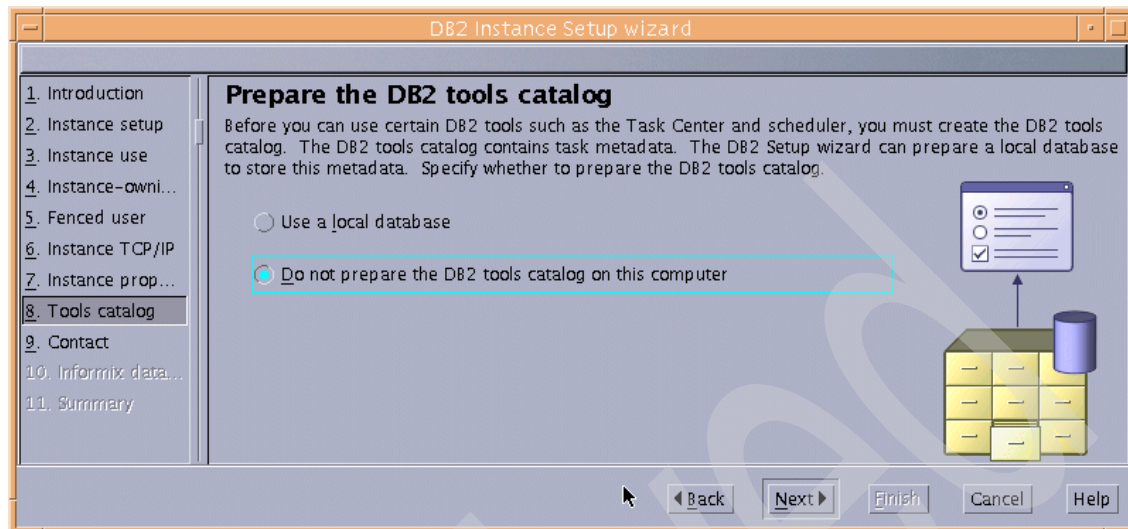


Figure 4-8 DB2 Instance setup - DB2 tools catalog

#### 14. Contact

Figure 4-9 enables you to specify an account for the health monitor to send administrator notifications. You can use an existing contact, or you can defer this task until after the installation.

Click **Next** to continue.

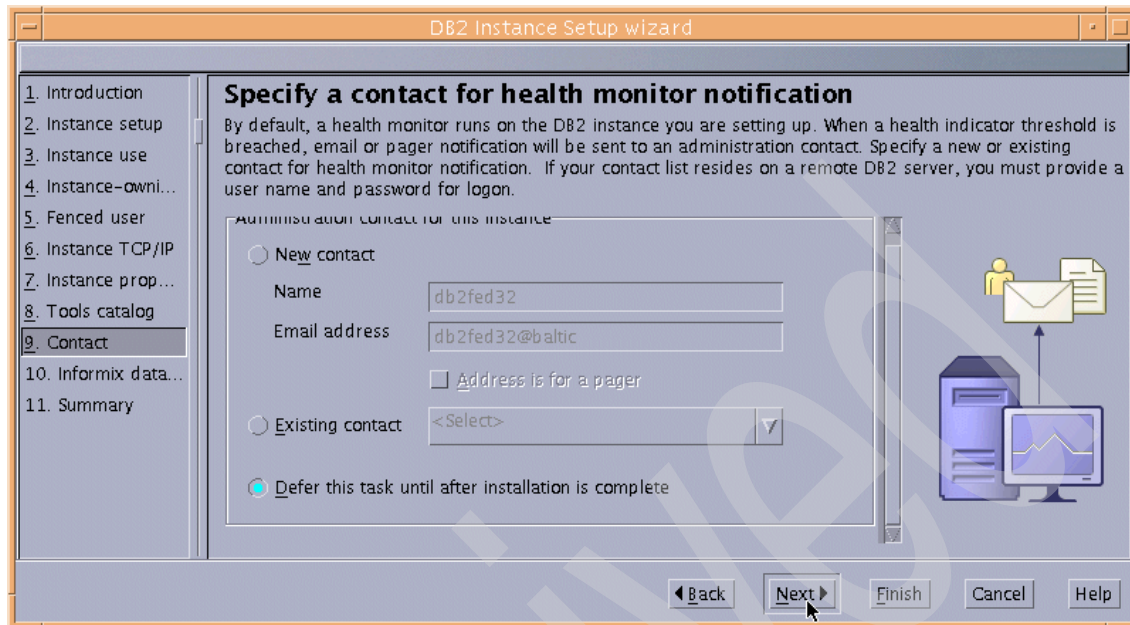


Figure 4-9 DB2 Instance setup - Monitor notification settings

## 15. Informix data source support

Figure 4-10 enables you to initialize a federated environment with an Informix Dynamic Server instance. You must install the Informix Client Software Development Kit (SDK) for Informix Dynamic Server in advance on your DB2 machine, and have the Informix database connection tested.

You then need to specify the **Informix Client Installation Directory** (which is the base directory on your DB2 machine). The value usually corresponds to the value of the INFORMIXDIR environment variable, if this variable is set. Or, you can look at the Informix client installation directories. The Informix Client base installation directory has the subdirectories /lib and /etc.

The second parameter identifies the **Informix server** name, as selected in the sqlhosts file. The value usually corresponds to the value of the INFORMIXSERVER environment variable, if this variable is set. Or, you can look at the Informix sqlhosts file. It is located in the Informix Client's /etc sub-directory. The INFORMIXSERVER value must correspond to the dbservername value in one of the records in the sqlhosts file; the dbservername is the first field in any record in the sqlhosts file.

Click **Next** to continue.

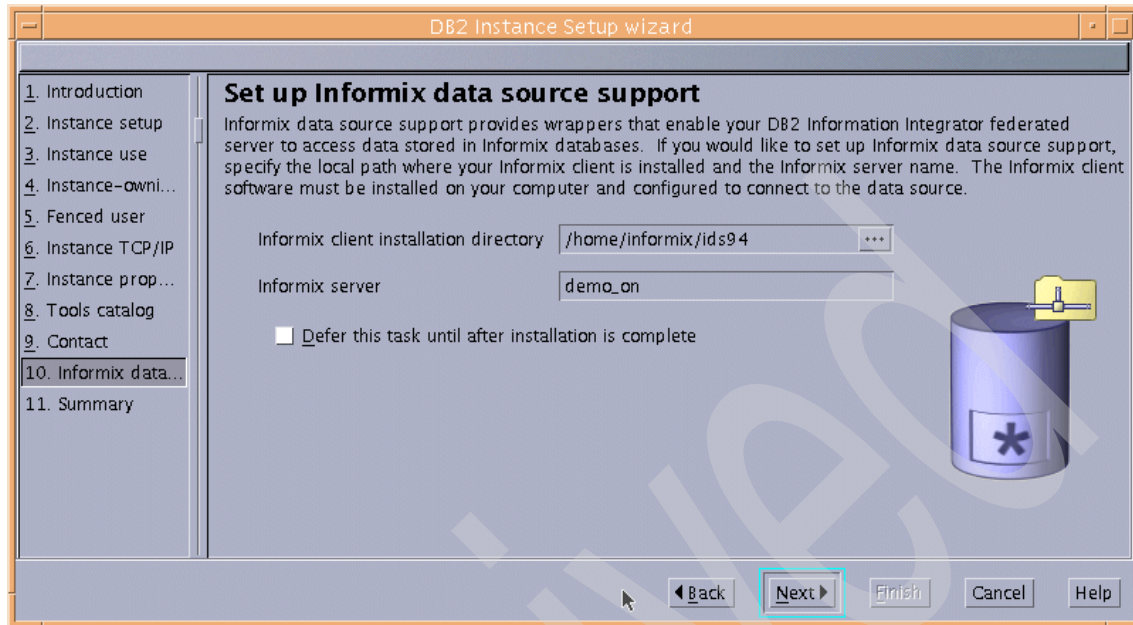


Figure 4-10 DB2 Instance setup - Informix data source support settings

## 16. Summary

Please check all your input, as this is the last chance to go back and correct it.  
Click **Finish** to continue.

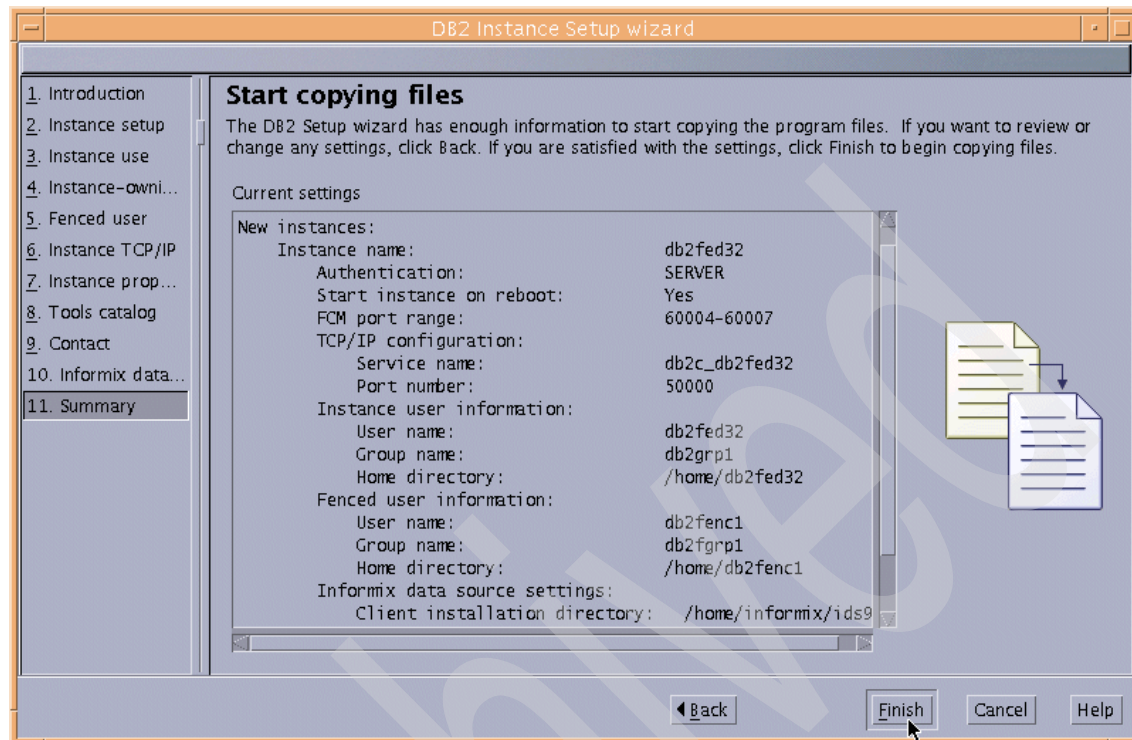


Figure 4-11 DB2 Instance setup - Summary information

#### 17. Setup Complete - Please check the **Status Report**.

In case of failure, please recheck your input in the setup screens, and verify that you installed all parts of the DB2 server correctly.

For Informix integration, please make sure you installed the Informix Client SDK on your DB2 machine, and that you are able to establish a connection to an IDS instance.

### 4.2.4 DB2 database configuration for federation

The steps to configure DB2 for federation support are:

1. Log in as the DB2 instance owner. In our case it is *db2fed32*.
2. Database Manager Configuration - FEDERATED

The DBM CFG parameter FEDERATED must be set to YES. You can check or set this parameter from the Control Center if:

- You select the **Instance-Name** => right-click => **Configure** => **Environment**

And on the command line type:

- **db2 get dbm cfg**

You can look for the setting of the parameter FEDERATED.

If it is not set properly, you need to type the following command to update this parameter:

- **db2 update dbm cfg using federated yes**

You then bounce the system before you continue.

### 3. Database Manager Configuration - AUTHENTICATION, FED\_NO\_AUTH

AUTHENTICATION=SERVER (default) and FED\_NO\_AUTH=NO (default):

- DB2 Information Integrator must receive a user ID and password from the client application or user.
- The user ID and password on the connection to DB2 Information Integrator must be in the user management at the DB2 Information Integrator server. Both user ID and password will be verified in user management at the Information Integrator server. User mapping to data sources is highly recommended. If there is no user mapping, DB2 Information Integrator will use the Information Integrator user ID and password on connections to different data sources.
- Workstation and iSeries DB2 Information Integrator users and applications can specify the DB2 Information Integrator user ID, and password in their connection to the DB2 Information Integrator database.
- DB2 for z/OS Information Integrator users and applications must have a record in the DB2 for z/OS table SYSIBM.USERNAMES to provide a password on the connection to the DB2 Information Integrator database.

AUTHENTICATION=CLIENT and FED\_NO\_AUTH=NO

- DB2 Information Integrator receives only a user ID (no password) from the DB2 Information Integrator client application or user.
- DB2 Information Integrator does not verify the user ID in the user management on the DB2 Information Integrator server.
- User mappings to data sources are recommended. User mappings are required if the data source requires a password.
- Workstation DB2 Client connection to DB2 Information Integrator must be configured for client authentication. User ID and password on the connect to DB2 Information Integrator are verified in the user management at the client side.
- DB2 for z/OS and iSeries connections to DB2 Information Integrator can use already verified connections. With DB2 for z/OS you do not need any

record in the table SYSIBM.USERNAMES, nevertheless, the host ID is sent on connection to DB2 Information Integrator.

AUTHENTICATION=SERVER and FED\_NO\_AUTH=YES

- DB2 Information Integrator must receive a user ID and password from the client application or user.
  - The user ID on the connection to DB2 Information Integrator must be in the user management at the DB2 Information Integrator server. The user ID will be verified in user management, but not the password. If there is no user mapping, DB2 Information Integrator will specify the DB2 Information Integrator user ID and password on the connections to the data sources. User mappings are only needed in DB2 Information Integrator to specify different user ID and/or password to different data sources.
  - Workstation and iSeries DB2 Information Integrator users and applications can specify DB2 Information Integrator user IDs and passwords in their connection to the DB2 Information Integrator database.
  - DB2 for z/OS Information Integrator users and applications must have a record in the DB2 for z/OS table SYSIBM.USERNAMES to provide a password on the connection to the DB2 Information Integrator database. Additionally, this record can translate the host user ID to the DB2 Information Integrator user ID if they are different.
- Database Manager Configuration - SHEAPTHRES
- Sets the maximum number of concurrent sort heaps for all databases in the instance
- db2dj.ini variables
- Default location of the file:
    - UNIX: [instance\_owner\_home]/sqllib/cfg/db2dj.ini
    - Windows: c:\Program Files\IBM\SQLLIB\cfg\db2dj.ini
  - This file contains environment variables that DB2 Information Integrator needs for non-DB2 data source client software.
  - The variable names are specific to the type of data source client, as shown in Table 4-1.

Table 4-1 DB2DJ.INI variables - Required and optional

<b>ORACLE</b>	ORACLE_HOME	required
	TNS_ADMIN	optional
	ORACLE_NLS	optional
	ORACLE_BASE	optional

<b>Informix</b>	INFORMIXDIR	required
	INFORMIXSDERVER	required
	INFORMIXSQLHOSTS	optional
<b>Sybase</b>	SYBASE	required
	SYBASE_OCS	required, with Sybase Open Client V12
<b>Teradata</b>	COPLIB	required
	COPERR	required
<b>DataDirect Connect</b>	ODBCINI	required
	DJX_ODBC_LIBRARY_PATH	required

- This file should not contain any extra lines.
- This file can be created and populated by installing the DB2 Information Integrator Relational Wrapper components. It can also be created, and entries added to it, with an editor such as *vi* on UNIX and *Notepad* on Windows.
- ▶ **db2set variables**
  - **DB2ENVLIST, DB2LIBPATH:**
    - These environment variables are required when using DataDirect Connect to Microsoft SQL Server.
    - db2set DB2ENVLIST=LIBPATH
    - db2set DB2LIBPATH=[directory with SQL Server ODBC driver library]  
Example: location of libivmss18.so is: /opt/odbc/lib  
db2set DB2LIBPATH=/opt/odbc/lib
  - **DB2\_DJ\_INI**
    - This parameter is required for Version 8 as long the file db2dj.ini is stored in default location.
  - **DB2\_DJ\_COMM (optional)**
    - This variable defines if the wrapper libraries should be loaded into memory when DB2 starts. This action saves time on the first federated connection to a data source.
- ▶ **System environment variables**
  - **PATH:** On Windows, the path is usually set when the data source client is being installed. On UNIX, edit the .profile file of the DB2 instance owner.



- Make sure all the environment variables defined in db2dj.ini are set and visible in your environment (env on UNIX, set on Windows).
- ▶ The Database Configuration parameter BUFFPAGE identifies the memory buffers for temporary tables that DB2 Information Integrator may use in execution of queries with nicknames. If federated server must use a temporary table in the execution of a query with nicknames, the temporary table will first be created in the memory buffers allocated for the temporary table space (TEMPSPACE1). If the temporary table is large, its pages will overflow to the file containers of TEMPSpace1. Allocating more memory for the buffer pool associated with TEMPSpace1 will improve the performance of federated queries by reducing the necessity for disk I/O during the execution. Temporary tables are common in the execution of queries with nicknames; they frequently process SQL operations that are not pushed down to data sources.
- ▶ The Database Configuration parameter SORTHEAP identifies the maximum memory buffers for each sort. If not enough buffers are available for a sort, the pages of the sort overflow to the buffer pool for TEMPSpace1. If the buffer pool for TEMPSpace1 (see BUFFPAGE above) cannot contain the sort, then the pages of the sort overflow to the file containers of TEMPSpace1, which requires disk I/O.

## 4.3 Integrating DB2 UDB for z/OS

This chapter describes all the steps necessary to integrate a data source based on DB2 for z/OS into a DB2 Federated Server.

We describe the steps necessary to actually establish a link between the table located on DB2 for z/OS and your federated server as follows:

- ▶ Cataloging DB2 for z/OS on your federated server
- ▶ Creating the DB2 for z/OS wrapper
- ▶ Creating the DB2 for z/OS server
- ▶ Altering DB2 for z/OS server definition and server options (optional)
- ▶ Creating DB2 for z/OS user mappings
- ▶ Altering DB2 for z/OS user mappings (optional)
- ▶ Creating DB2 for z/OS nicknames

We used the Control Center on the Windows platform to perform the necessary administration steps.

### 4.3.1 Cataloging DB2 for z/OS

The information in Table 4-2 is necessary to integrate a DB2 for z/OS data source.

Table 4-2 The DB2 for z/OS system

Parameter	Value
Host name	9.12.6.8
TCP Port	33378
User	<user>
Password	<password>
Location	DB2G
Creator	PAOLOR4

We show how to catalog a database with the DB2 command line:

1. Log in with user db2fed32 to the AIX system.
2. Catalog the node with:  

```
db2 catalog tcpip node DB2ZSRV remote 9.12.6.8 server 33378
```
3. Store information about the remote host in the Database Connection Services (DCS) directory:  

```
db2 catalog dcs database DCSDB2G as DB2G with "Comment on DB2 for z/OS"
```
4. Catalog the database:  

```
db2 catalog database DCSDB2G at node DB2ZSRV authentication dcs
```
5. Test the connection with:  

```
db2 connect to DCSDB2G user <user> using <password>
```

The value specified as the Database Name/DB Alias (DCSDB2G) in our catalog database command and specified in our connection test will be the value we will specify as the setting for the DBNAME server option in our federated server definition for this DB2 for z/OS data source.

Now, we start using Control Center to integrate the DB2 subsystem on z/OS to our db2fed32 instance.

### 4.3.2 Creating the DB2 for z/OS wrapper

**Note:** If there is an existing wrapper for DB2, you can reuse it for your DB2 for z/OS subsystem.

Otherwise, here are the steps to create it:

1. Open the DB2 Control Center.
2. Expand your Instance and database.
3. Select the federated database (FEDDB) and expand the icon tree beneath it.
4. Right-click **Federated Database Objects** for database **feddb** and click **Create Wrapper**.
5. Choose the data source type and enter a name for the wrapper and click **OK**.

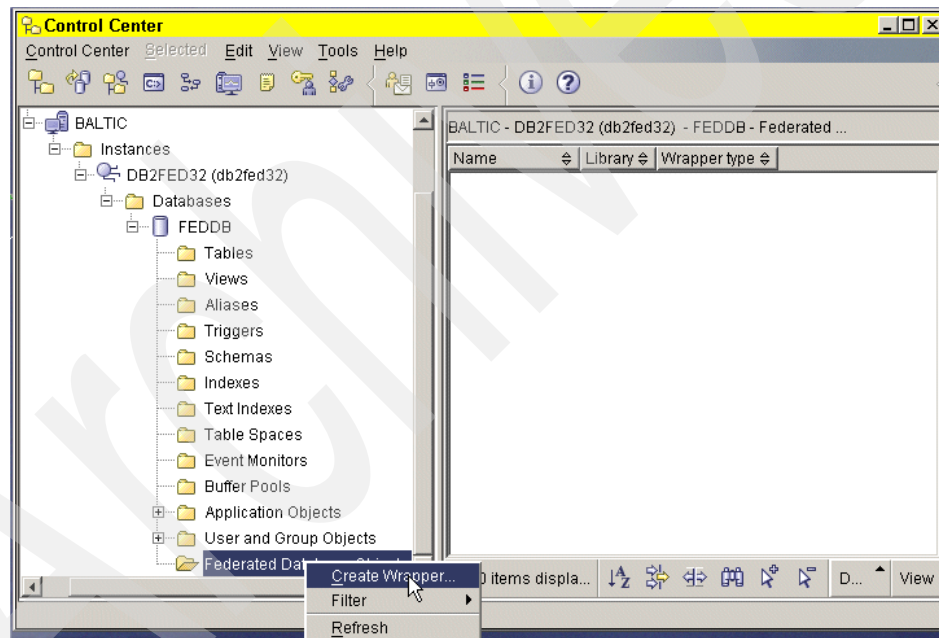


Figure 4-12 DB2 for z/OS - Create wrapper

After calling the Create Wrapper dialog (as displayed in Figure 4-13) you need to define the wrapper details. You need to select **DB2** for the **Data source**, and enter a unique name for the wrapper. Remember, this wrapper is being used by DB2 on all different platforms.

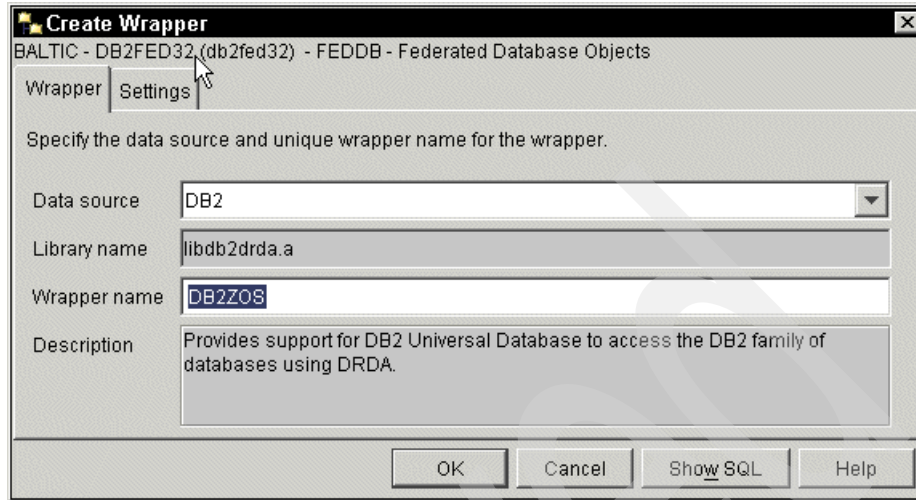


Figure 4-13 DB2 for z/OS - Create wrapper dialog

Example 4-1 shows the command line version for creating the wrapper for your DB2 for z/OS instance. Additionally, you may check your wrapper definition in the DB2 system catalog with the two select statements listed below.

Example 4-1 DB2 for z/OS - Create wrapper statements

---

```
CONNECT TO FEDDB;
CREATE WRAPPER "DB2ZOS" LIBRARY 'libdb2drda.a';

SELECT * from SYSCAT.WRAPPERS;
SELECT * from SYSCAT.WRAPOPTIONS;
```

---

### 4.3.3 Creating the DB2 for z/OS server

A server definition identifies a data source to the federated database. A server definition consists of a local name and other information about that data source server. Since we have just created the DB2 wrapper, we need to specify the DB2 server from which you want to access data. Once the wrapper is defined, you can create several server definitions.

The steps to create a DB2 for z/OS server are shown in Figure 4-14:

1. Select the wrapper you created in the previous step, **DB2ZOS**.
2. Right-click **Servers** for wrapper **DB2ZOS** and click **Create**.

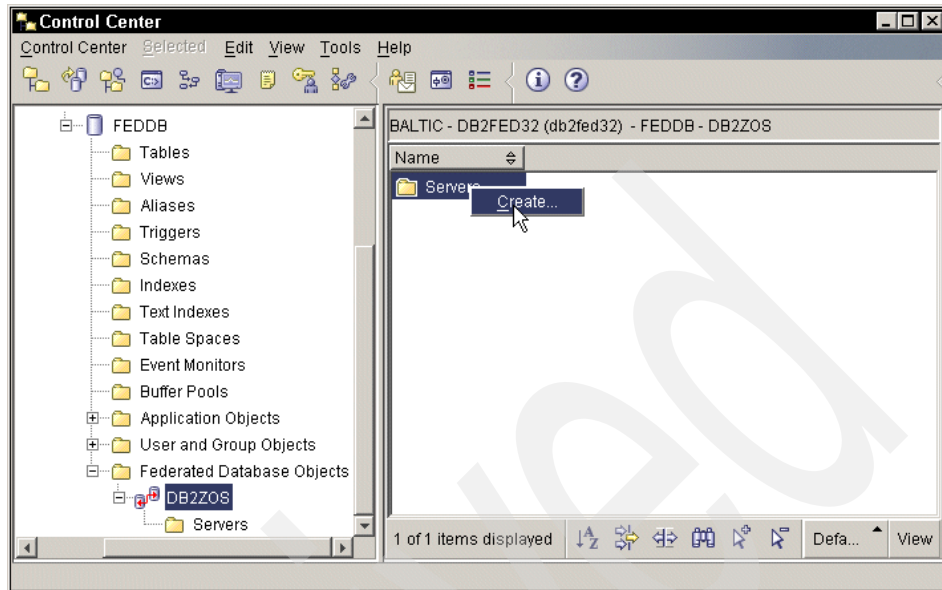


Figure 4-14 DB2 for z/OS - Create server

The server definition requires the input visualized in Figure 4-15:

- ▶ **Name:** The name of the server must be unique over all server definitions that are available on this federated database.
- ▶ **Type:** Select **DB2/390**
- ▶ **Version:** Select **6** or **7**
- ▶ User ID/password

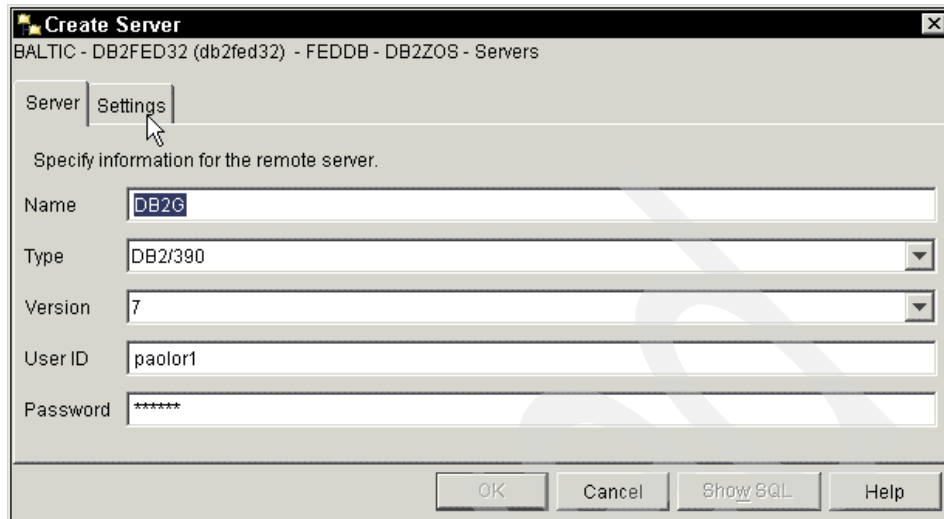


Figure 4-15 DB2 for z/OS - Create server dialog

Switch to the **Settings** menu to complete your server definition. See Figure 4-16.

For server settings, the first two input fields, DBName and Password, are required definitions, the rest are optional.

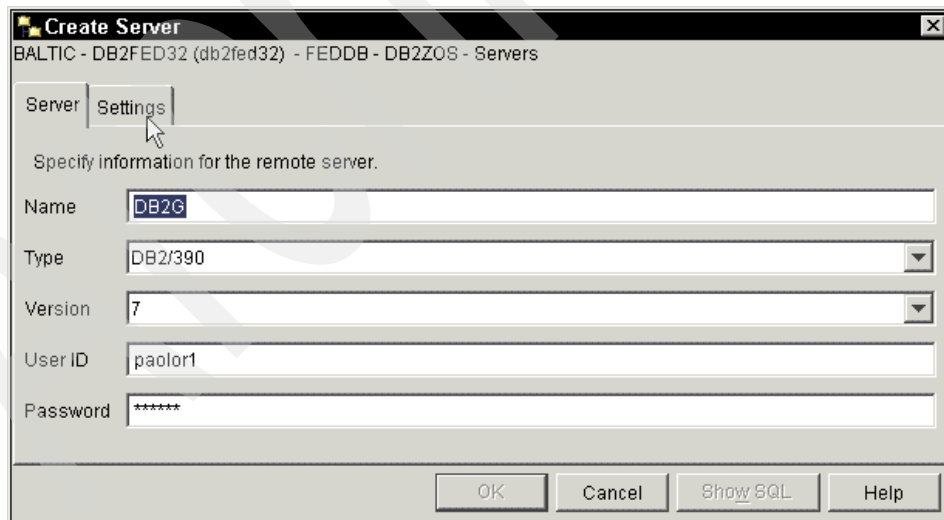


Figure 4-16 DB2 for z/OS - Create server dialog - Settings

Server options are used to describe a data source server. The DB2 for z/OS server has the set of options listed in Table 4-3. You can set the options at

creation time of your server, or you can modify these settings afterwards. Please read 4.3.4, “Altering DB2 for z/OS server definition and server options” on page 98 for more details.

Table 4-3 DB2 for z/OS server options

<b>Connectivity</b>	<b>DBName</b>	<b>DB Alias in DB Directory on DB2 Information Integrator server</b>
	Password	<b>Default=Y</b> , Include the password on connections to DB2 for z/OS
	Fold_ID / Fold_PW	Option to fold or not for the DRDA wrapper. <b>Default=N</b> , Connect once with user ID/password exactly as specified (recommended) <b>Y</b> - connect up to four times with user ID/password with all combinations of lower and uppercase <b>U</b> - connect once with user ID/password in uppercase <b>L</b> - connect once with user ID/password in lower case
<b>Pushdown</b>	Collating_Sequence	It specifies whether the data source uses the same default collating sequence as the federated database. This affects the pushdown of operations on character columns. <b>Default=N</b> <b>Y</b> : both use same collating sequence. ORDER BY can be pushed down without compromising integrity of result <b>N</b> : both use different collation sequence. ORDER BY cannot be pushed down <b>I</b> : case-insensitive. ORDER BY, DISTINCT, WHERE= cannot be pushed down
	Pushdown	<b>Default: Y</b> : the SQL operations are pushed down to data sources based on decision of pushdown analysis and optimizer.
<b>Optimization</b>	CPU_RATIO	<b>Default: 1.0</b> : specifies the ratio of the DB2 Information Integrator server CPU capacity against the data source CPU capacity
	IO_RATIO	<b>Default: 1.0</b> : specifies the ratio of the DB2 Information Integrator server IO rate against the data source IO rate
	COMM_RATE	<b>Default: 2</b> : specifies the effective data rate of network to data source in MB per second.
<b>Other</b>	IUD_APP_SVPT_E NFORCE	<b>Default=Y</b> : should the DB2 federated system use save-points in multi-update transactions?

The following Example 4-2 shows the command line version for creating the server for your DB2 for z/OS instance. Additionally, you may check your server definition in the DB2 system catalog with the two select statements included in the same example.

#### Example 4-2 DB2 for z/OS - Create server statements

---

```
CONNECT TO FEDDB;  
CREATE SERVER DC TYPE DB2/390 VERSION '7' WRAPPER "DB2ZOS" AUTHID "paolor1"  
PASSWORD "*****" OPTIONS( ADD DBNAME 'DCSDB2G', PASSWORD 'Y');  
  
SELECT * from SYSCAT.SERVERS;  
SELECT * from SYSCAT.SERVEROPTIONS;
```

---

### 4.3.4 Altering DB2 for z/OS server definition and server options

You may modify a server definition when you:

- ▶ Upgrade DB2 for z/OS to a new version:  
`alter server db2zos version 7`
- ▶ Want to change server options to different values

Server options are set to values that persist over successive connections to the data source. The values are stored in the federated system catalog. Here is an example to activate (add), set and deactivate (drop) a server option:

```
alter server db2zos options (add IO_RATIO '2.0')  
alter server db2zos options (set IO_RATIO '3.0')  
alter server db2zos options (drop IO_RATIO)
```

To set a server option value temporarily, you can use the SET SERVER OPTION statement. This statement overrides the server option value in the server definition for the duration of a single connection to the federated database. The settings are not stored in the federated system catalog. An example is:

```
set server option COMM_RATE TO '3' for server DB2ZOS
```

The server option DB2\_MAXIMAL\_PUSHDOWN, outlined in Table 4-4, specifies the primary criteria for the optimizer in choosing the access plan.

**Tip:** The server option shown in Table 4-4 is not available through the Control Center, but setting this option to Y might help you in analyzing your federated queries.



Table 4-4 DB2 for z/OS additional server options

<b>DB2_MAXIMAL_PUSHDOWN</b>	Default: 'N' The optimizer can choose between cost optimization and the user requirement to perform as much as possible query processing by the remote data source. 'Y': choose the plan with most query operations to be pushed down to the data sources. 'N': choose the plan with minimum cost.
-----------------------------	---

### 4.3.5 Creating DB2 for z/OS user mappings

When the federated server needs to access the data source, it needs first to establish the connection with it. With the user mapping, you define an association between the federated servers user ID, and a user ID and password at the DB2 for z/OS data source. An association must be created for each user that will be using the federated system. In our case, we only used one user ID, db2fed32.

The steps to create a DB2 for z/OS user mapping are shown in Figure 4-17:

1. Select the server we created in the previous step - **DB2G**.
2. Right-click **User Mappings** for server **DB2G** and click **Create**.

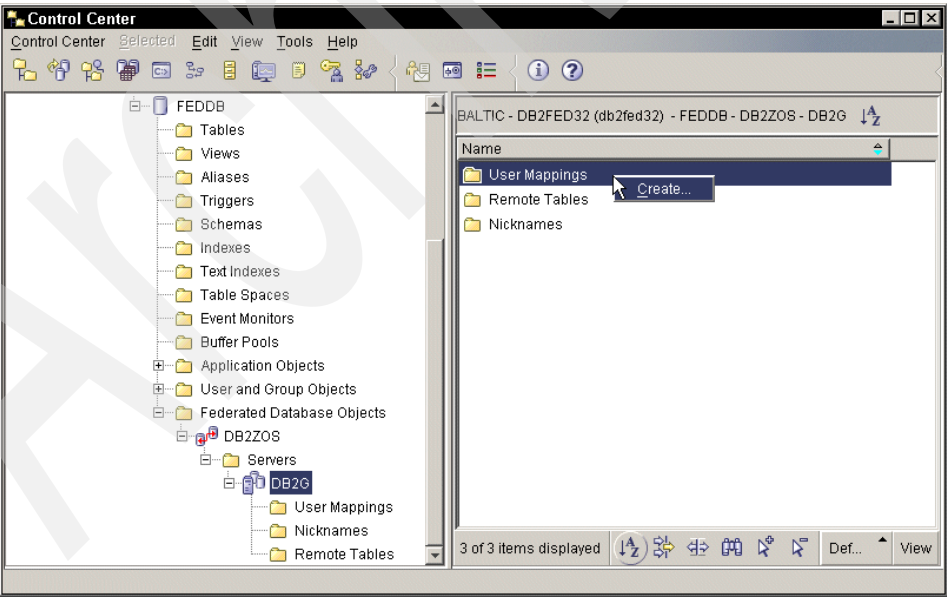


Figure 4-17 DB2 for z/OS - Create user mapping

Figure 4-18 lists all user IDs available on your federated system. Select the user who will be the sender of your distributed requests to the DB2 for z/OS data source. We selected the owner of our federated server instance, **db2fed32**.

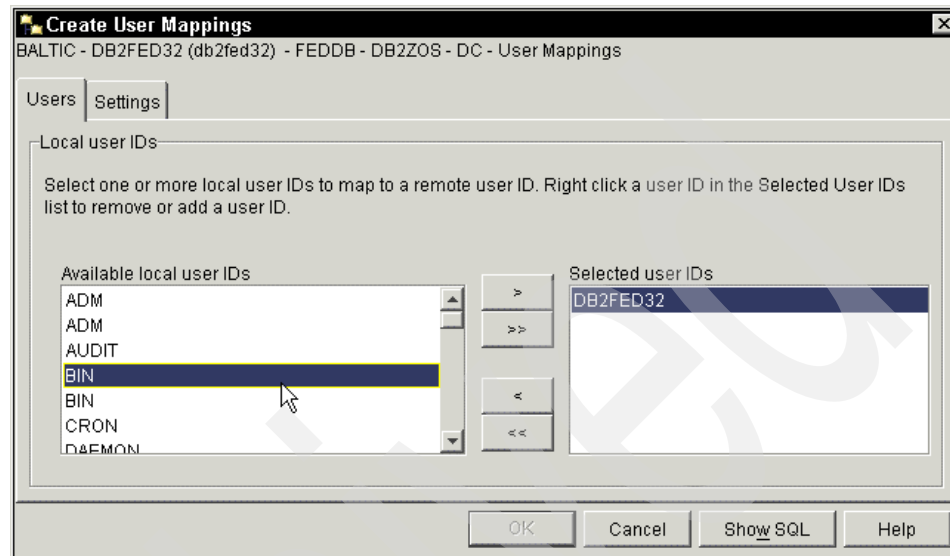


Figure 4-18 DB2 for z/OS - Create user mapping dialog

Switch to **Settings** as shown in Figure 4-19, to complete the user mapping. You need to identify the user name and password to enable the federated system to connect to our DB2 for z/OS data source.

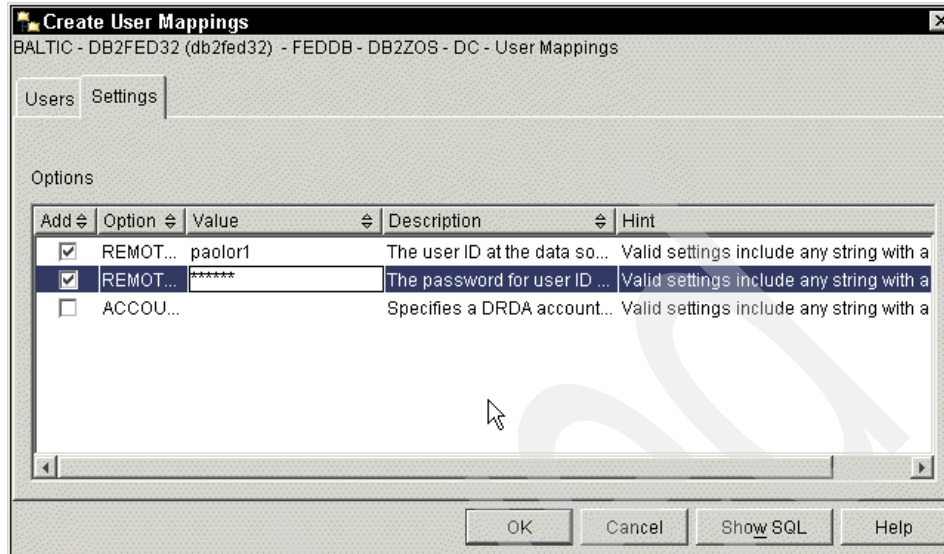


Figure 4-19 DB2 for z/OS - Create user mapping settings

Example 4-3 shows the command line version of creating the user mapping for your DB2 for z/OS instance. Additionally, you may check your user mapping definition in the DB2 system catalog with the select statement listed in the example.

Example 4-3 DB2 for z/OS - Create user mapping statements

```
CONNECT TO FEDDB;
CREATE USER MAPPING FOR "DB2FED32" SERVER "DC" OPTIONS ( ADD REMOTE_AUTHID
'paolor1', ADD REMOTE_PASSWORD '*****' ) ;

SELECT * from SYSCAT.USEROPTIONS;
```

You might also consider adding user mapping option ACCOUNTING\_STRING. DB2 for z/OS is the only data source that uses it.

### 4.3.6 Altering DB2 for z/OS user mappings

You can use the ALTER USER MAPPING statement to modify a user mapping. It is needed to change the authorization ID or password that is used at the DB2 for z/OS data source:

```
alter user mapping for db2fed32 SERVER DB2ZOS options
(set remote_authid 'paolo')
alter user mapping for db2fed32 SERVER DB2ZOS options
(set remote_password 'newpass')
```

**Note:** The REMOTE\_AUTHID and REMOTE\_PASSWORD user options are *case sensitive* unless you set the FOLD\_ID and FOLD\_PW server options to U or L in the CREATE SERVER statement.

The user mapping can also be altered in the DB2 Control Center.

### 4.3.7 Creating DB2 for z/OS nicknames

After having set up the DB2 wrapper, the server definition, and the user mapping to our DB2 for z/OS database, we now finally need to create the actual link to a table located on our remote database as a nickname.

When you create a nickname for a DB2 for z/OS table, catalog data from the remote server is retrieved and stored in the federated global catalog.

The steps to create a DB2 for z/OS nickname are shown in Figure 4-20:

1. Select the server **DC**.
2. Right-click **Nicknames** for server **DC** and click **Create**.

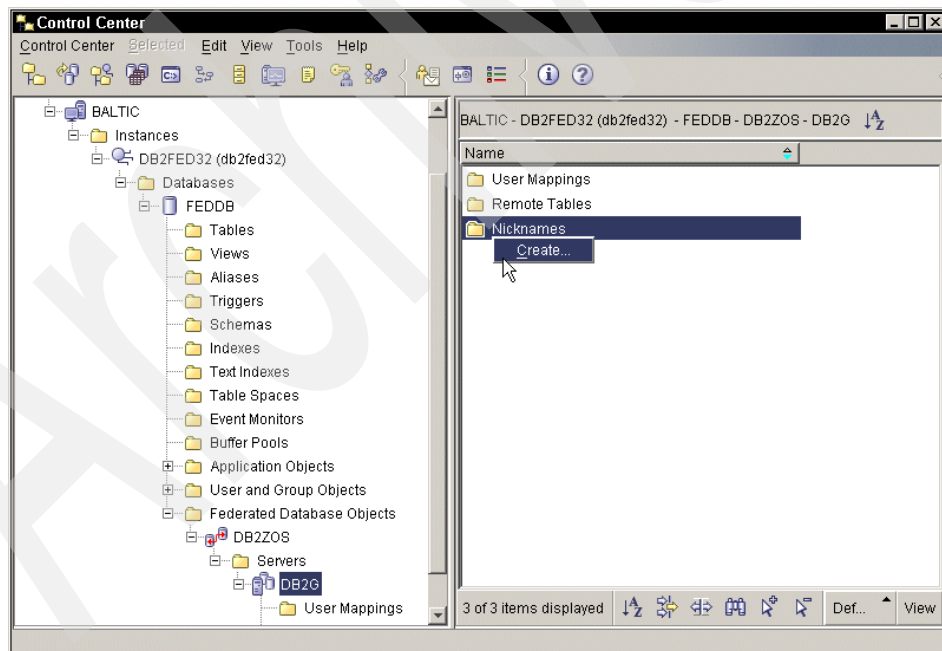


Figure 4-20 DB2 for z/OS - Create nicknames

You will now see the dialog displayed in Figure 4-21 showing up. You have two possibilities to add a nickname. Either you click **Add** to manually add a nickname by specifying local and remote schema and table identification, or you can use the **Discover** functionality, which we describe here.

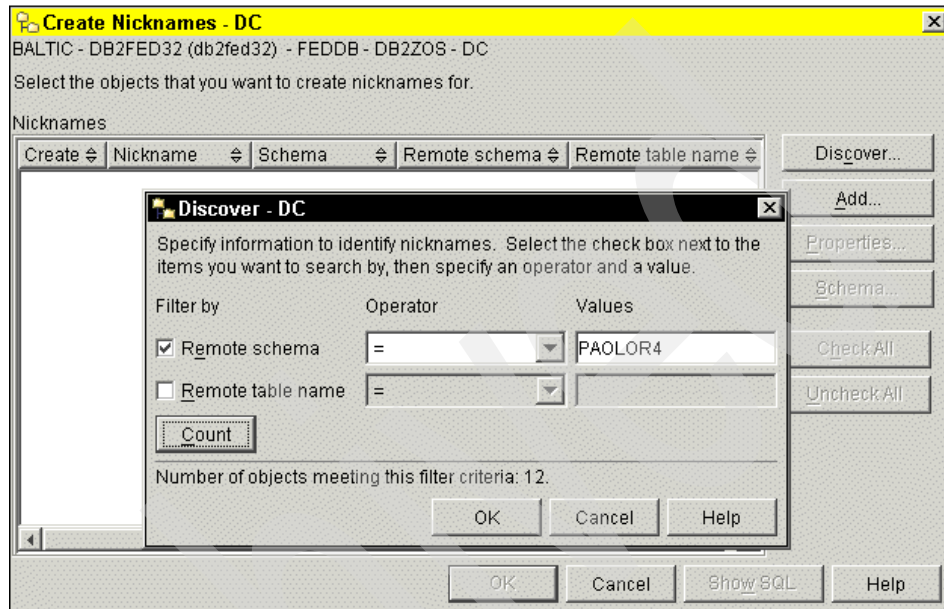


Figure 4-21 DB2 for z/OS - Nickname created

You will *discover* your remote DB2 for z/OS data source by specifying either a **remote schema** or a specific **remote table name** that you want to create a nickname for.

Choose a **filter** method, an **operator** and the comparing **value**, and click **Count**. Upon successful access to your DB2 for z/OS database, the number of objects that meet the filter criteria will be displayed. Adjust your filter criteria if you find the number is too few or too high. Click **OK** to list the actual objects as listed in the dialog displayed in Figure 4-22.

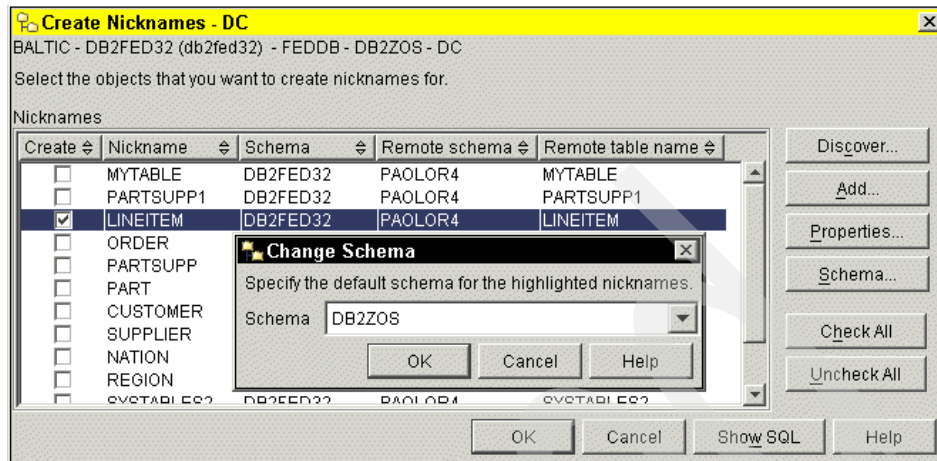


Figure 4-22 DB2 for z/OS - Change schema

Note that when the Create Nickname is populated with records from the Discover Filter:

- ▶ All the entries are checked so that nicknames will be created for all if the **OK** button is clicked. Click the button **Uncheck All** if you do not want to create all these nicknames at once, and if you want to pick which nicknames you will create from the list.
- ▶ The default schema for the nicknames is the user ID that is creating it. In our case that is DB2FED32. We immediately want to change this so that the schema of our new nicknames will adhere to our naming convention.

Use the **Schema** button to change the local schema for your DB2 for z/OS nicknames.

**Tip:** We recommend to use the same schema name for all DB2 for z/OS nicknames.

The following Example 4-4 shows the command line version of creating the nicknames for your DB2 for z/OS instance. Additionally, you may check the nickname definition in the DB2 system catalog with the select statements listed at the bottom of the example.

*Example 4-4 DB2 for z/OS - Create nickname statements*

```
CONNECT TO FEDDB;
CREATE NICKNAME DB2ZOS.LINEITEM FOR DC.PAOLOR4.LINEITEM;
CREATE NICKNAME DB2ZOS.ORDER FOR DC.PAOLOR4.ORDER;
CREATE NICKNAME DB2ZOS.PARTSUPP FOR DC.PAOLOR4.PARTSUPP;
```

```

CREATE NICKNAME DB2ZOS.PART FOR DC.PAOLOR4.PART;
CREATE NICKNAME DB2ZOS.CUSTOMER FOR DC.PAOLOR4.CUSTOMER;
CREATE NICKNAME DB2ZOS.SUPPLIER FOR DC.PAOLOR4.SUPPLIER;
CREATE NICKNAME DB2ZOS.NATION FOR DC.PAOLOR4.NATION;
CREATE NICKNAME DB2ZOS.REGION FOR DC.PAOLOR4.REGION;

SELECT * from SYSCAT.TABLES WHERE TABSCHEMA='DB2ZOS';
SELECT * from SYSCAT.TABOPTIONS WHERE TABSCHEMA='DB2ZOS';
SELECT * from SYSCAT.COLUMNS WHERE TABSCHEMA='DB2ZOS';
SELECT * from SYSCAT.COLOPTIONS WHERE TABSCHEMA='DB2ZOS';
SELECT * from SYSCAT.INDEXES WHERE TABSCHEMA='DB2ZOS';
SELECT * from SYSCAT.INDEXOPTIONS WHERE TABSCHEMA='DB2ZOS';
SELECT * from SYSCAT.KEYCOLUSE WHERE TABSCHEMA='DB2ZOS';

```

---

### 4.3.8 Altering DB2 for z/OS nicknames

You can use the ALTER NICKNAME statement to modify the federated database representation of a data source. Use this statement to:

- ▶ Change local data type of a column:
 

```
alter nickname DB2ZOS.REGION alter column r_name local type varchar (50)
```
- ▶ Change local column name:
 

```
alter nickname DB2ZOS.REGION alter column r_name local name "R_REGIONNAME"
```

## 4.4 Integrating Informix Dynamic Server

This chapter describes all steps necessary to integrate a data source based on Informix Dynamic Server into a DB2 Federated Server.

We show how to actually establish a link between the table located on Informix Dynamic Server and your federated server by describing all the steps that you need to perform. The steps are:

1. Informix client configuration
2. Creating the Informix wrapper
3. Creating the Informix server
4. Altering Informix server definition and server options (optional)
5. Creating Informix user mappings
6. Altering Informix user mappings (optional)
7. Creating Informix nicknames

We use the Control Center on Windows platform to perform the necessary administration steps.

## 4.4.1 Informix client configuration

The information in Table 4-5 is necessary to integrate an Informix Dynamic Server (IDS) data source:

Table 4-5 The Informix system

Parameter	Value
INFORMIXDIR	/home/informix/ids94
INFORMIXSERVER	demo_on
ONCONFIG	onconfig.demo
SQLHOSTS	demo_on onsoctcp 30001
User/password	informix/informix
DBName	tpc

**Note:** Make sure you installed the Client Software Development Kit (CSDK) on the federated server, and that you successfully configured and tested the connection to your Informix server.

## 4.4.2 Informix wrapper libraries

The DB2 ESE 8.1 wrapper libraries on AIX for use with Informix are:

- ▶ libdb2informix.a  
It is installed with DB2 ESE 8.1 custom install.
- ▶ libdb2informixU.a  
It is installed with DB2 ESE 8.1 custom install.
- ▶ libdb2informixF.a  
It is created during the installation of the DB2 ESE 8.1 feature for Informix distributed join support by linking with the Informix Client SDK. The input library for this link is libdb2STinformixF.a, which was provided with the DB2 ESE 8.1 installation.

These libraries are created in the DB2 UDB 8.1 /lib directory (/usr/opt/db2\_08\_01/lib) and should also appear in the DB2 instance owner's /lib directory (/home/db2fed32/sqllib/lib).

If the base and 'U' wrapper libraries are there, but not the 'F' library, the **djxlink** script will need to be used to create it. The **djxlink** scripts are located in the DB2 UDB 8.1 /bin directory (/usr/opt/db2\_08\_01/bin). To create just libdb2informixF.a,



use the **djxlinkInformix** script. **Djxlink** is run by root. For more information on **djxlink**, see 5.10, “Troubleshooting” on page 235.

## The Informix wrapper and DB2 FixPaks

Updates to the Informix wrapper are included in DB2 ESE 8.1 FixPaks.

On Windows, the FixPak updates the DB2 engine, and all the files of the Informix wrapper at the same time.

On UNIX, however, the FixPak updates only the base and ‘U’ Informix wrapper libraries to the new FixPak level leaving the ‘F’ library at the old maintenance level. This is because the ‘F’ library is the one that is linked with the Informix client. The FixPak installation process unfortunately does not do this link. If there are updates to the Informix wrapper in the FixPak, it will provide a new input library (libdb2STinformixF.a) for the link with the Informix client, and possibly an updated **djxlinkInformix**. After the FixPak is applied, **djxlinkInformix** must be run by root to create a libdb2informixF.a that is the same level as the DB2 engine and the base and ‘F’ wrapper libraries.

**Note:** If a FixPak is applied and **djxlinkInformix** is not run, what happens when nicknames are created or used is unpredictable.

## Informix sqlhosts file

The Informix sqlhosts file contains information the Informix Client uses to connect to the Informix server. The file is usually located in the /etc sub-directory of the Informix client.

There needs to be an entry for the Informix server that we will be defining federated access to.

The left-most value in a sqlhosts record is called the dbservername value and is the value that will be used as the NODE setting of our federated Server definition to the Informix server.

On Windows, the Informix sqlhosts information is kept in the Windows Registry. Look in the informix folder in the Windows Registry.

## db2dj.ini

As indicated in Table 4-1 on page 89, the DB2 instance owner’s /sqlib/cfg/db2dj.ini file must contain the variables INFORMIXDIR and INFORMIXSERVER.

- ▶ INFORMIXDIR indicates the Informix client’s base directory
- ▶ INFORMIXSERVER indicates an entry in the sqlhosts file.

- INFORMIXSQLHOSTS is optional. It is only required if the sqlhosts file is not in its default location, which is the Informix client's /etc sub-directory.

### 4.4.3 Creating the Informix wrapper

The steps for creating the IDS wrapper are the following:

1. In the Control Center, expand your Instance and database.
2. Right-click **Federated Database Objects** for database **feddb** and click **Create Wrapper**.
3. Choose data source **Informix** and enter a unique **Wrapper name** like shown in Figure 4-23.
4. Click **OK**.

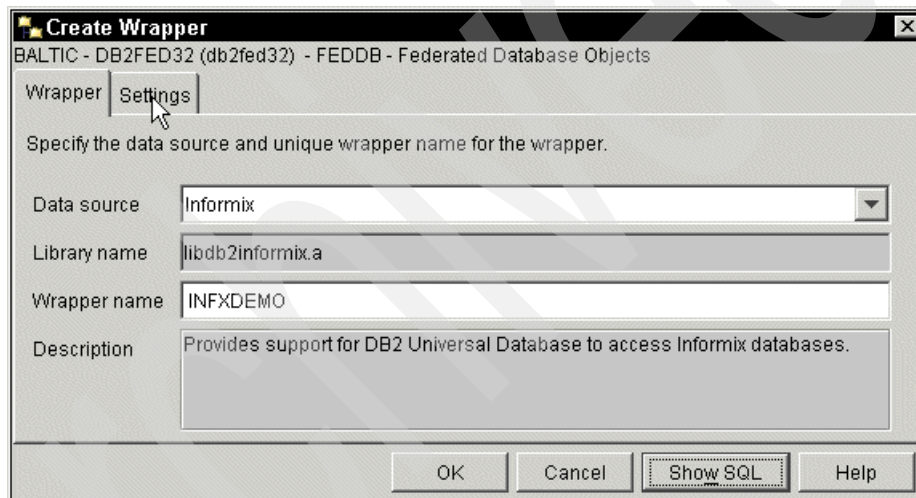


Figure 4-23 IDS - Create wrapper dialog

The following Example 4-5 shows the command line version of creating the wrapper for your Informix instance. Please check the wrapper definition by issuing a select against the two DB2 system catalog tables listed.

#### Example 4-5 Informix - Create wrapper statements

```
CONNECT TO FEDDB;
CREATE WRAPPER "INFXDEMO" LIBRARY 'libdb2informix.a';

SELECT * from SYSCAT.WRAPPERS;
SELECT * from SYSCAT.WRAPOPTIONS;
```

## 4.4.4 Creating the Informix server

A server definition identifies a data source to the federated database. A server definition consists of a local name and other information about that data source server. Since we have just created the Informix wrapper, we need to specify the Informix server, from which you want to access data. For your defined wrapper, you can create several server definitions.

Here are the steps to create an Informix server:

1. Select the wrapper you created in the previous step - **INFXDEMO**.
2. Right-click **Servers** for wrapper **INFXDEMO** and click **Create**.

The server definition demands the following inputs, shown in Figure 4-24:

- ▶ **Name:** The name of the server must be unique over all server definitions available on your federated database.
- ▶ **Type:** INFORMIX - preselected
- ▶ **Version:** Select **7** - IDS 7.xx, **8** - XPS 8.xx, **9** - IDS 9.xx

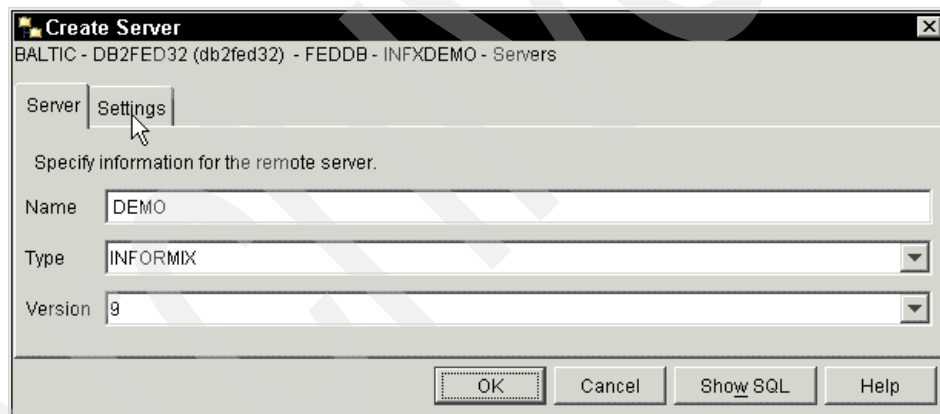


Figure 4-24 IDS - Create server dialog

Switch to the **Settings** menu to complete your server definition. See Figure 4-25 for details.

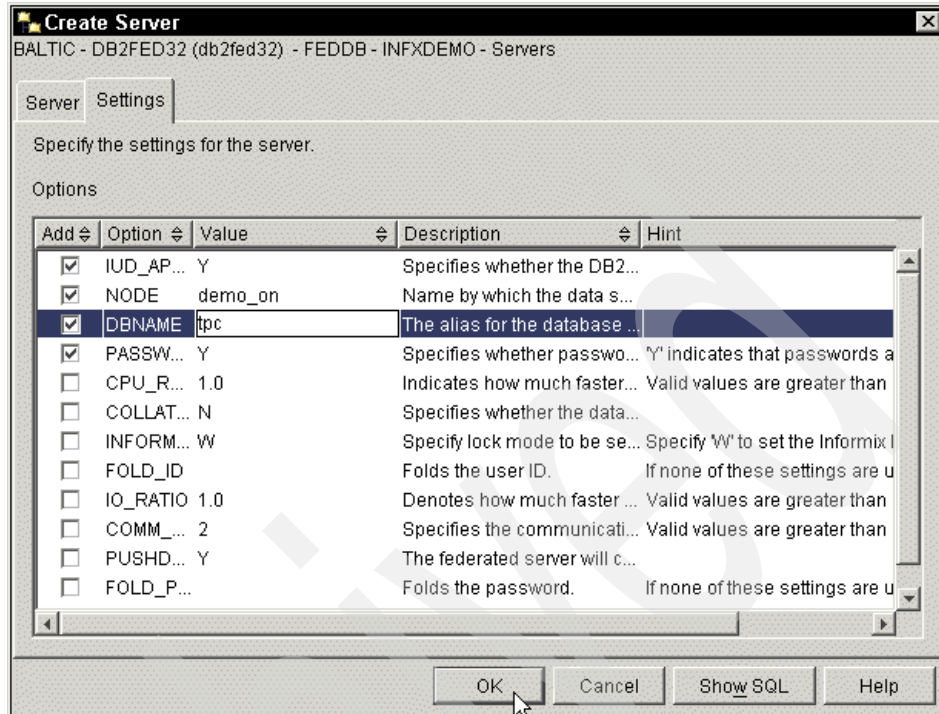


Figure 4-25 IDS - Create server dialog - Settings

For an Informix server you need to specify the first four options, in order to entirely define the connection to your Informix server. All other server options are optional.

Server options are used to describe a data source server. Informix servers have the set of options listed in table Table 4-6. You can set the options at creation time of your server, or modify these settings afterwards. Please read 4.4.5, “Altering Informix server definition and server options” on page 112 for more details.

Table 4-6 Informix server options

<b>Connectivity</b>	Node	sqlhosts entry name, case sensitive
	DBName	database at the Informix server, case sensitive
	Password	<b>Default=Y:</b> Include the password on connections to Informix
	Fold_ID / Fold_PW	<b>Default</b> is wrapper dependent. <b>Y</b> - connect four times with user ID/password with all combinations of lower and uppercase. <b>N</b> (recommended) - connect once with user ID/password exactly as specified <b>U</b> - connect once with user ID/password in uppercase <b>L</b> - connect once with user ID/password in lower case
<b>Pushdown</b>	Collating_Sequence	It specifies whether the data source uses the same default collating sequence as the federated database. This affects the pushdown of operations on character columns. <b>Default=N</b> <b>Y:</b> both use same collating sequence. ORDER BY can be pushed down without compromising integrity of result <b>N:</b> both use different collation sequence. ORDER BY cannot be pushed down <b>I:</b> case-insensitive. ORDER BY, DISTINCT, WHERE= cannot be pushed down
	Pushdown	<b>Default: Y:</b> the SQL operations are pushed down to data sources based on decision of pushdown analysis and optimizer.
<b>Optimization</b>	CPU_RATIO	<b>Default: 1.0:</b> specifies the ratio of the DB2 Information Integrator server CPU capacity against the data source CPU capacity.
	IO_RATIO	<b>Default: 1.0:</b> specifies the ratio of the DB2 Information Integrator server IO rate against the data source IO rate.
	COMM_RATE	<b>Default: 2:</b> specifies the effective data rate of network to data source in MB per second.

Other	IUD_APP_SVPT_ENFORCE	<b>Default=Y.</b> Informix does not support savepoints. If set to 'Y', since the Informix server does not support external savepoints, in case of error in a write operation that requires savepoint processing, DB2 Information Integrator will rollback to the beginning of the user transaction (unit of work) to ensure that the write operation maintains its statement atomicity. If set to 'N', it means that in an error case, DB2 Information Integrator will not rollback for the users, it will return an error code to be handled by the application.
	INFORMIX_LOCK_MODE	<b>Default=W</b> (wait): specifies the lock mode to be set for the Informix data source. The wrapper issues a 'SET LOCK MODE' command just after establishing the connection to a data source. Wait means, to wait for a locked table or row, until its being freed by the locking process. <b>N</b> (no wait): do not wait for the locked object to be released, just return an error. <b>&lt;Nr&gt;</b> (wait <Nr> seconds): Wait number of seconds for the locked object to be released. If the object is not released within this time, an error is returned.

Example 4-7 shows the command line version of creating the server definition for your Informix data source. Additionally, you may check your server definition in the DB2 system catalog with the select statements included.

#### Example 4-6 Informix - Create server statements

```
CONNECT TO FEDDB;
CREATE SERVER DEMO TYPE INFORMIX VERSION '9' WRAPPER "INFXDEMO" OPTIONS( ADD
NODE 'demo_on', DBNAME 'tpc', PASSWORD 'Y', IUD_APP_SVPT_ENFORCE 'Y');

SELECT * from SYSCAT.SERVERS;
SELECT * from SYSCAT.SERVEROPTIONS;
```

**Attention:** NODE and DBNAME are case sensitive!

### 4.4.5 Altering Informix server definition and server options

You may modify a server definition when you:

- ▶ Upgrade Informix server to a new version, for example, from IDS 7.31 to 9.40:

```
alter server DEMO version 9
```
- ▶ Want to change server options to different values:

```
alter server DEMO options (add informix_lock_mode '10')
alter server DEMO options (set informix_lock_mode 'W')
alter server DEMO options (set informix_lock_mode 'N')
```

```
alter server DEMO options (drop informix_lock_mode)
```

To set a server option value temporarily, use the SET SERVER OPTION statement. This statement overrides the server option value in the server definition for the duration of a single connection to the federated database. The settings are not stored in the federated system catalog. For example:

```
set server option informix_lock_mode 'N' for server DEMO
```

The server option DB2\_MAXIMAL\_PUSHDOWN, outlined in Table 4-7, specifies the primary criteria for the optimizer in choosing the access plan.

**Tip:** The server option shown in Table 4-7 is not available through Control Center, but setting this option to Y might help you in analyzing your federated queries.

Table 4-7 Informix additional server options

<b>DB2_MAXIMAL_PUSHDOWN</b>	Default: N The optimizer can choose between cost optimization and the user requirement to perform as much as possible query processing by the remote data source. <b>Y:</b> choose the plan with most query operations to be pushed down to the data sources. <b>N:</b> choose the plan with minimum cost.
-----------------------------	---

## 4.4.6 Creating Informix user mappings

When the federated server needs to access the data source server, it needs first to establish the connection with it. With the user mapping, you define an association between the federated servers user ID and password, to the Informix data source. An association must be created for each user that will be using the federated system. In our case, we only used one user ID, db2fed32.

Steps to create a user mapping definition:

1. Select the server we created in 4.4.4, “Creating the Informix server” on page 109 - **DEMO**.
2. Right-click **User Mappings** for server **DEMO** and click **Create**.

Figure 4-26 lists all the user IDs available on your federated system. Select the user who will be the sender of your distributed requests to the Informix data source. We selected the owner of our federated server instance, **db2fed32**.

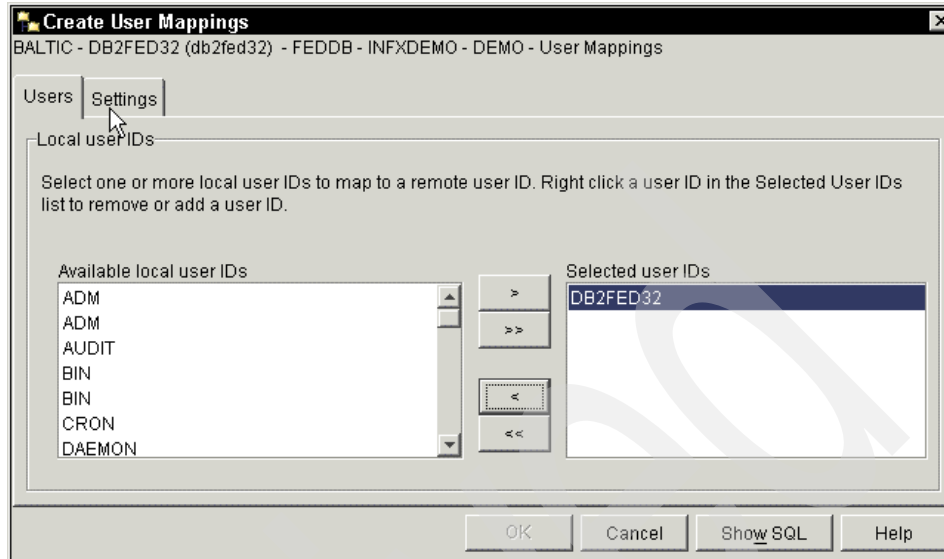


Figure 4-26 IDS - Create user mapping dialog

Switch to **Settings** as shown in Figure 4-27, to complete the user mapping. You need to identify the user name and password to enable the federated system to connect to our Informix data source.

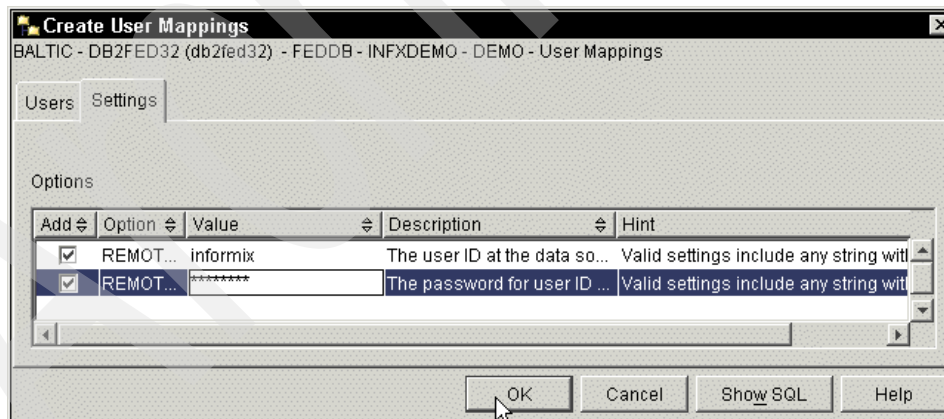


Figure 4-27 IDS - Create user mapping settings

Example 4-7 shows the command line version of creating the user mapping for your Informix instance. Additionally, you may check your user mapping definition in the DB2 system catalog with the select statements listed below.



#### Example 4-7 Informix - Create user mapping statements

---

```
CONNECT TO FEDDB;  
CREATE USER MAPPING FOR "DB2FED32" SERVER "DEMO" OPTIONS ( ADD REMOTE_AUTHID  
'informix', ADD REMOTE_PASSWORD '*****') ;  
  
SELECT * from SYSCAT.USEROPTIONS;
```

---

### 4.4.7 Altering Informix user mappings

You can use the ALTER USER MAPPING statement to modify a user mapping. It is used to change the authorization ID or password that is used at the DB2 for z/OS data source:

```
alter user mapping for db2fed32 SERVER DEMO options  
(set remote_authid 'ifmx9')  
alter user mapping for db2fed32 SERVER DEMO options  
(set remote_password 'sanjose')
```

**Note:** The REMOTE\_AUTHID and REMOTE\_PASSWORD user options are case sensitive unless you set the FOLD\_ID and FOLD\_PW server options to 'U' or 'L' in the CREATE SERVER statement.

### 4.4.8 Creating Informix nicknames

Having set up the Informix wrapper, the server definition, and the user mapping for our Informix database, we finally need to create the actual link to a table located on our remote Informix database as a nickname.

When you create a nickname for an Informix table, catalog data from the remote server is retrieved and stored in the federated global catalog.

The steps to create an Informix nickname are:

1. Select the server **DEMO**.
2. Right-click **Nicknames** for server **DEMO** and click **Create**.

The dialog displayed in Figure 4-28 shows up. You have two possibilities to add a nickname. Either you click **Add** to manually add a nickname by specifying local and remote schema and table identification, or you can use the **Discover** functionality, which we want to describe here.

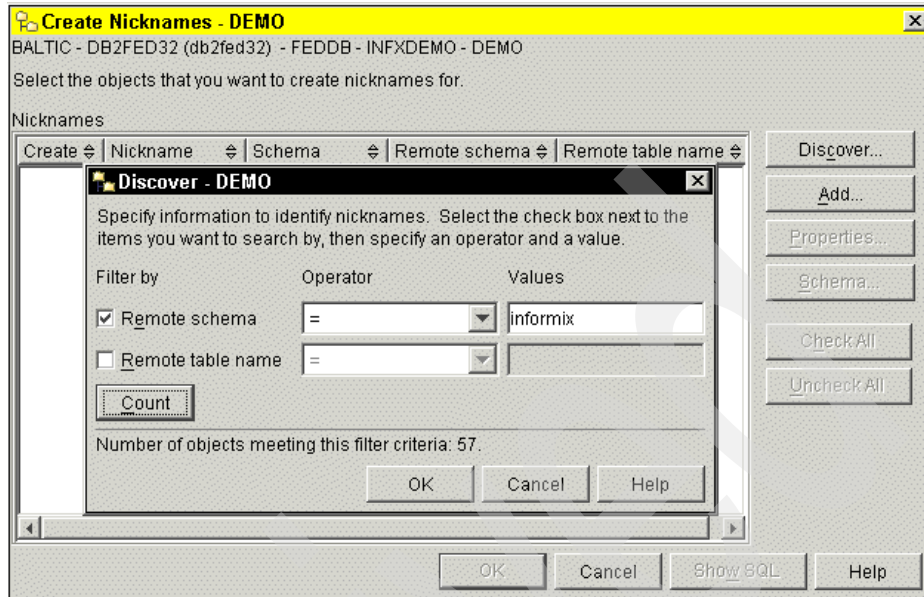


Figure 4-28 IDS - Discover

You will discover your remote Informix data source by specifying either a **remote schema** or a specific **remote table** name that you want to create a nickname for.

Choose a **filter** method, an **operator**, and the comparing **value**, and click **Count**. Upon successful access to our Informix database, the number of objects which meet the filter criteria will be displayed. Adjust your filter criteria if you find the number is too few or many. Click **OK** to list the actual objects in the dialog, as displayed in Figure 4-29.

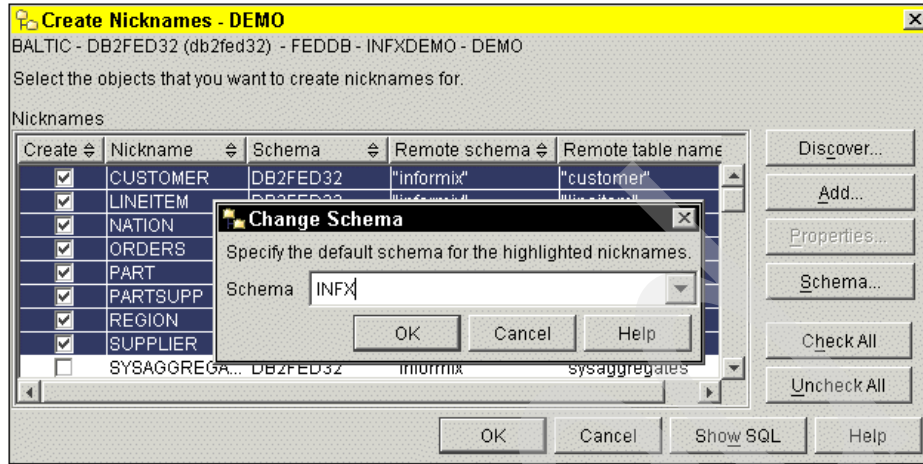


Figure 4-29 IDS - Change schema

If you do not change the schema, the default setting in the Control Center's Create Nickname window is the user ID that is creating the nicknames. In our case, that is DB2FED32. We wanted to change the nickname schema to adhere to our naming convention.

Use the **Schema** button to change the local schema for your Informix nicknames.

**Tip:** We recommend that you use the same schema name for all your Informix nicknames.

Example 4-8 shows the command line version of creating the nicknames for your Informix instance. Additionally, you may check the nickname definition in the DB2 system catalog with the select statements included in the example.

#### Example 4-8 Informix - Create nickname statements

```
CONNECT TO FEDDB;
CREATE NICKNAME INFX.NATION FOR DEMO."informix"."nation";
CREATE NICKNAME INFX.REGION FOR DEMO."informix"."region";
CREATE NICKNAME INFX.PART FOR DEMO."informix"."part";
CREATE NICKNAME INFX.SUPPLIER FOR DEMO."informix"."supplier";
CREATE NICKNAME INFX.PARTSUPP FOR DEMO."informix"."partsupp";
CREATE NICKNAME INFX.CUSTOMER FOR DEMO."informix"."customer";
CREATE NICKNAME INFX.ORDERS FOR DEMO."informix"."orders";
CREATE NICKNAME INFX.LINEITEM FOR DEMO."informix"."lineitem";

SELECT * from SYSCAT.TABLES WHERE TABSCHEMA ='INFX';
SELECT * from SYSCAT.TABOPTIONS WHERE TABSCHEMA ='INFX';
SELECT * from SYSCAT.COLUMNS WHERE TABSCHEMA ='INFX';
```

```

SELECT * from SYSCAT.COLOPTIONS WHERE TABSCHEMA ='INFX';
SELECT * from SYSCAT.INDEXES WHERE TABSCHEMA ='INFX';
SELECT * from SYSCAT.INDEXOPTIONS WHERE TABSCHEMA ='INFX';
SELECT * from SYSCAT.KEYCOLUSE WHERE TABSCHEMA ='INFX';

```

---

## 4.4.9 Altering Informix nicknames

You can use the ALTER NICKNAME statement to modify the federated database representation of a data source. Use this statement to:

- ▶ Change the local data type of a column:  
`alter nickname INFX.REGION alter column r_name local type varchar (50)`
- ▶ Change local column name:  
`alter nickname INFX.REGION alter column r_name local name "R_REGIONNAME"`

## 4.5 Integrating DB2 UDB for iSeries

The information in table Table 4-8 is necessary to integrate a DB2 for iSeries data source.

*Table 4-8 DB2 for iSeries system*

Parameter	Value
Host name	9.5.92.35
TCP Port	446
User/Password	<user>/<password>
RDB Name	ITSO
Library Name - Schema	SAMPLEDB

How to catalog a database with the DB2 command line:

1. Connect with user db2fed32 on the AIX system.
2. Catalog the node with:  
`db2 catalog tcpip node DB2ISRV remote 9.5.92.35 server 446`
3. Stores information about remote host in the Database Connection Services (DCS) directory:  
`db2 catalog dcs database DCSDB0S4 as ITSO with "Comment on DB2/iSeries"`
4. Catalog the database:

db2 catalog database DCSDB0S4 at node DB2ISRV authentication dcs

5. Test the connection with:

db2 connect to DCSDB0S4 user <user> using <password>

For further information on setting up wrappers, servers, user mappings, and nicknames, please refer to 4.3.2, “Creating the DB2 for z/OS wrapper” on page 93 and subsequent pages.

**Note:** If there is an already existing wrapper for DB2, please reuse this for your DB2 for iSeries.



# Installing and configuring DB2 Information Integrator

In this chapter we describe how to install and configure DB2 Information Integrator Version 8.1 for use with several data sources. The topics in this chapter are:

- ▶ General prerequisites
- ▶ Installing DB2 Information Integrator
- ▶ Applying installed wrappers to instances
- ▶ Integrating Oracle 9i
- ▶ Integrating Microsoft SQL Server 2000
- ▶ Integrating XML data source
- ▶ Integrating table-structured files
- ▶ Integrating Microsoft Excel
- ▶ Maintaining
- ▶ Troubleshooting

## 5.1 General prerequisites

You must have installed the following software products before you start installing DB2 Information Integrator:

- ▶ DB2 Enterprise Server Edition Version V8.1.2 minimum.
- ▶ Oracle Client - to integrate Oracle data sources
- ▶ Open database connectivity driver - to use Microsoft SQL Server data sources. We used *DataDirect Connect for ODBC UNIX*. DataDirect Connect for ODBC UNIX is the only one supported by the DB2 Information Integrator Relational Wrapper on AIX for Microsoft SQL Server.
- ▶ DB2 Instance available - in our case db2fed32. Please refer to Chapter 3, “The case study” on page 53 for further information.
- ▶ For the relational data sources, you will need to install the data source client on the DB2 Information Integrator server as follows:
  - **Oracle**  
Install Oracle Client, and create a new entry for the Oracle server in `tnsnames.ora`
  - **Sybase**  
Install Sybase Open Client, and create a new entry for the Sybase server in the *interfaces* file (on UNIX), or *sql.ini* file (on Windows)
  - **Informix**  
Informix Client SDK, and create a new entry for the Informix server in the `sqlhosts` file (on UNIX), or `SQLHOSTS` Registry (on Windows)
  - **Microsoft SQL Server:**
    - DB2 Information Integrator on Windows: SQL Server ODBC driver on Windows. An entry for the SQL Server needs to be added to the system DSNs in ODBC data source administration.
    - DB2 Information Integrator on UNIX/Linux: Install DataDirect Connect ODBC and create an entry for the SQL Server in the *odbc.ini* file.
  - **Teradata**
    - DB2 Information Integrator on Windows: Install Teradata Windows CLI Client and create an entry for the Teradata server in `c:\winnt\system32\drivers\etc\hosts` file.
    - DB2 Information Integrator on AIX: Install Teradata CLIV2 (PIOM and BTEQ are also recommended for testing) and create the entry for the Teradata server in `/etc/hosts`.
  - **ODBC data sources**



Install an open database connectivity 3.0 driver, and create the entry for the data source in .odbc.ini file (UNIX/Linux), or create a System DSN entry for the data source in the ODBC data source administration (on Windows).

- ▶ For non relational data sources:
  - **Excel files**  
Excel 97 or 2000 (DB2 Information Integrator server must be able to read the Excel file.)
  - **Flat file**  
DB2 Information Integrator server must be able to read the file.
  - **XML**  
DB2 Information Integrator server must be able to read the file.

We recommend to install, configure, and test the connectivity to your federated data sources prior to the installation of DB2 Information Integrator. If the connectivity software is added after you install DB2 Information Integrator, you need to:

- ▶ Manually update the environment, with: *db2dj.ini* file (UNIX, Windows).
- ▶ Set the environment variables.
- ▶ Link the wrapper to the data source client software.

The following requirements need to be met in order to ensure a successful installation:

- ▶ DB2 Information integrator, its components, and the data source client software (Oracle Client, DataDirect Connect ODBC, etc.) must be installed on the same server.
- ▶ DB2 Information Integrator installation procedure need graphical user interface support.
- ▶ The installation procedure needs root authority on AIX.

The supported operating systems for relational wrappers and non relational wrappers may vary, depending on which wrappers you install. Table 5-1 shows each wrapper and the operating systems it supports for IBM DB2 Information Integrator Version 8.1.2. at the time of writing. For current information you must verify with the official documentation listed at “Other publications” on page 413.

Table 5-1 Wrappers and supported operating systems

Wrapper	Windows	AIX	HP-UX	Linux	Solaris
<b>DRDA</b>	32-bit, 64-bit	32-bit, 64-bit	32-bit, 64-bit	32-bit, 64-bit	32-bit, 64-bit
<b>Informix</b>	32-bit	32-bit, 64-bit	32-bit, 64-bit	32-bit	32-bit, 64-bit
<b>Microsoft SQL Server</b>	32-bit	32-bit	32-bit	32-bit	32-bit
<b>ODBC</b>	32-bit	32-bit	32-bit	32-bit	32-bit
<b>OLE DB</b>	32-bit				
<b>Oracle NET8</b>	32-bit	32-bit, 64-bit	32-bit, 64-bit	32-bit	32-bit, 64-bit
<b>Oracle SQLNET</b>	32-bit	32-bit			
<b>Sybase CTLIB</b>	32-bit	32-bit, 64-bit	32-bit, 64-bit	32-bit	32-bit, 64-bit
<b>Sybase DBLIB</b>	32-bit	32-bit	32-bit	32-bit	32-bit
<b>Teradata</b>	32-bit	32-bit			
<b>BLAST</b>	32-bit	32-bit		32-bit	32-bit
<b>BioRS</b>	32-bit	32-bit			
<b>Documentum</b>	32-bit	32-bit			32-bit
<b>Entrez</b>	32-bit	32-bit			
<b>HMMER</b>	32-bit	32-bit			
<b>IBM Lotus Extended Search</b>	32-bit	32-bit			
<b>Microsoft Excel</b>	32-bit				
<b>Table-structured files</b>	32-bit	32-bit, 64-bit	32-bit, 64-bit	32-bit	32-bit, 64-bit
<b>XML</b>	32-bit	32-bit	32-bit	32-bit	32-bit

Regarding 32-bit and 64-bit software, DB2 ESE 8.1, the Information Integrator wrappers, and the data source client software need to be 32-bit, or they need to be 64-bit. There cannot be a mixing of DB2 engine, wrapper, and data source client software. The exception is that on 64-bit UNIX systems, the 64-bit Oracle

Client can be used with a 32-bit DB2 instance and 32-bit DB2 Information Integrator Relational Wrapper for Oracle; the Oracle 64-bit client comes with a sub-directory lib32 containing the 32-bit client interface libraries, and the 32-bit Information Integrator Relational Wrapper for Oracle can link to the Oracle Client libraries in the Oracle lib32 directory, instead of the 64-bit Oracle Client interface libraries in the Oracle lib directory.

## 5.2 Installing DB2 Information Integrator

DB2 Information Integrator installs DB2 Enterprise Server Edition, relational wrappers and non relational wrappers. After you install the wrappers, you need to set up a federated server and database to configure and access the data sources.

If DB2 Enterprise Server Edition or DB2 Connect Enterprise Edition is already installed at the correct level, reinstall is not required. The installation wizard within DB2 Information Integrator will detect previous installations.

If you add DB2 Information Integrator relational and non relational wrappers to a DB2 ESE that is at FixPak 3 or higher, then you will need to re-apply the DB2 FixPak after the Information Integrator wrappers are installed in order to apply the updates of the FixPak to the wrappers. On UNIX (AIX, Solaris, HP-UX, and Linux), if you install the relational wrappers for Oracle, Sybase, Microsoft SQL Server, or Teradata, and apply the DB2 FixPak to update the wrappers, you must also run the appropriate **djxlink** scripts to update wrapper libraries to the FixPak level. On UNIX, the wrappers are linked to the data source client software; the FixPak provides an input library for the link, but does not do the link itself. The **djxlink** scripts will do the link with the new input library provided by the FixPak to create a wrapper library that is linked to the data source client, and at the FixPak level.

**Note:** If you do not run the **djxlink** after applying the FixPak, what happens when you try to configure and use Information Integrator is unpredictable, since the DB2 engine and the wrappers will be at different FixPak levels.

See 5.10 “Troubleshooting” on page 235 for more information about **djxlink**.

Since we have selected to work on an AIX environment, we have to run **djxlink** and **db2iupdt** to update the wrappers that are installed by DB2 Information Integrator.

This section is structured as:

- Start the installer.

- ▶ Installation response file examples
- ▶ Installation hints, tips, and techniques

### 5.2.1 Start the installer

To install DB2 Information Integrator:

1. Open AIX terminal window with root authority.
2. `cd /cdrom`
3. Set the DISPLAY environment variable.
4. `./iiSetup.bin`

This call takes you to the dialog displayed in Figure 5-1. Please review the release notes and the installation requirements available with the product for successful installation process.



Figure 5-1 DB2 Information Integrator Installation - Launch screen

By clicking **Install Products**, the installer launches the next dialog.

► Software License Agreement

Click “**I accept the terms in the license agreement**”, as shown in Figure 5-2 to continue with the **Next** dialog.

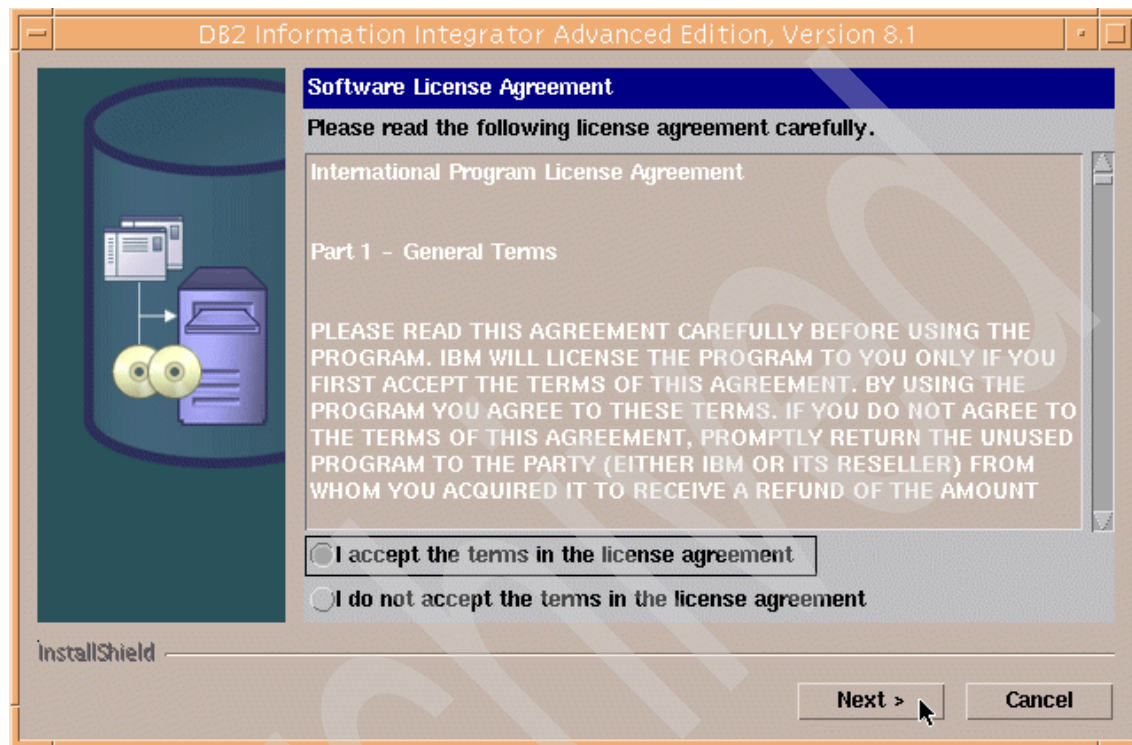


Figure 5-2 SW license agreement

► Product selection

In Figure 5-3 you choose whether you want to install **Relational wrappers** or **Non relational wrappers**.

Relational wrappers include libraries to access relational data sources like Teradata, Sybase, Microsoft SQL Server, and Oracle.

Non relational wrappers include libraries to access structured files (Excel, Table-structured files, XML), application data sources (BioRS, Documentum, Entrez, IBM Lotus Extended Search) and scientific related data (BLAST, HMMER).

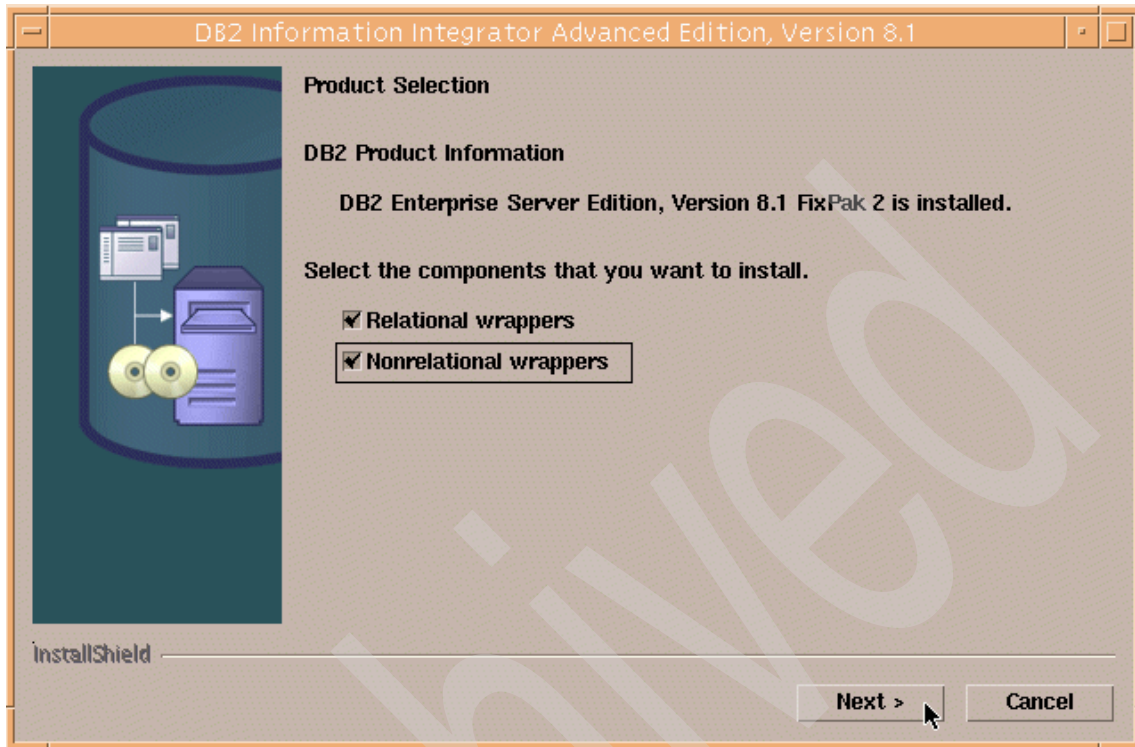


Figure 5-3 Product selection - Relational, non relational wrappers

Click **Next** to continue.

- ▶ Source of relational wrapper installation  
Browse the CD or any other file system to select the path for the relation wrappers source, as shown in Figure 5-4.

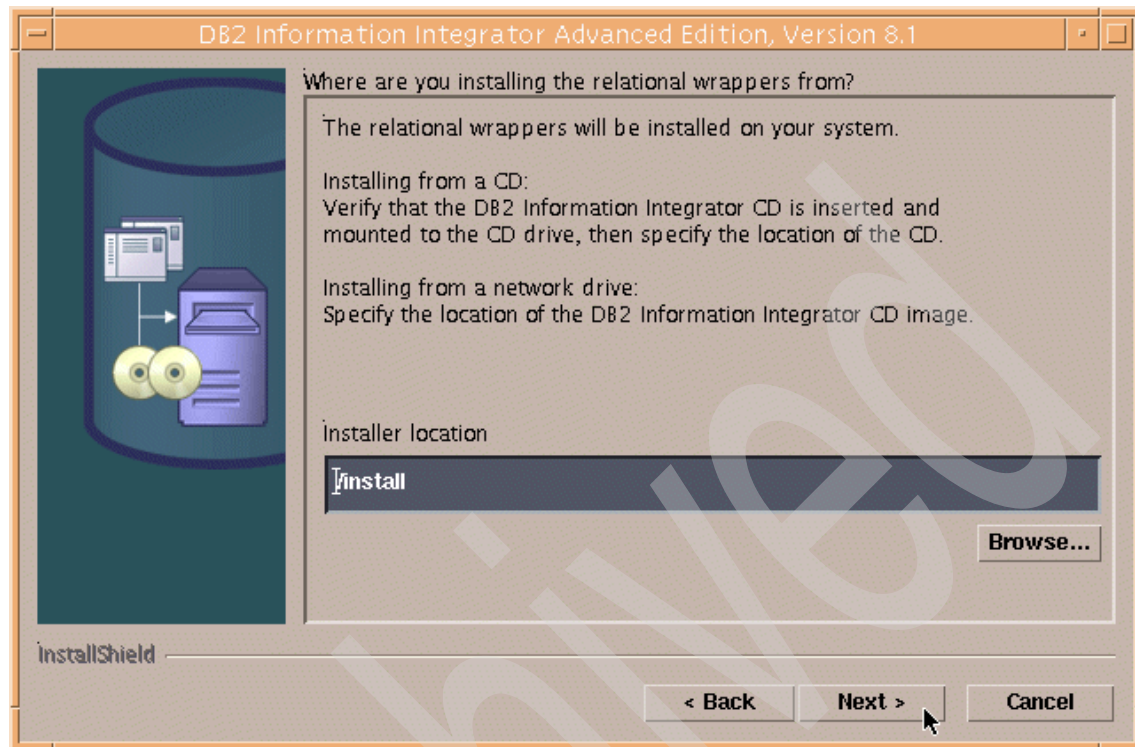
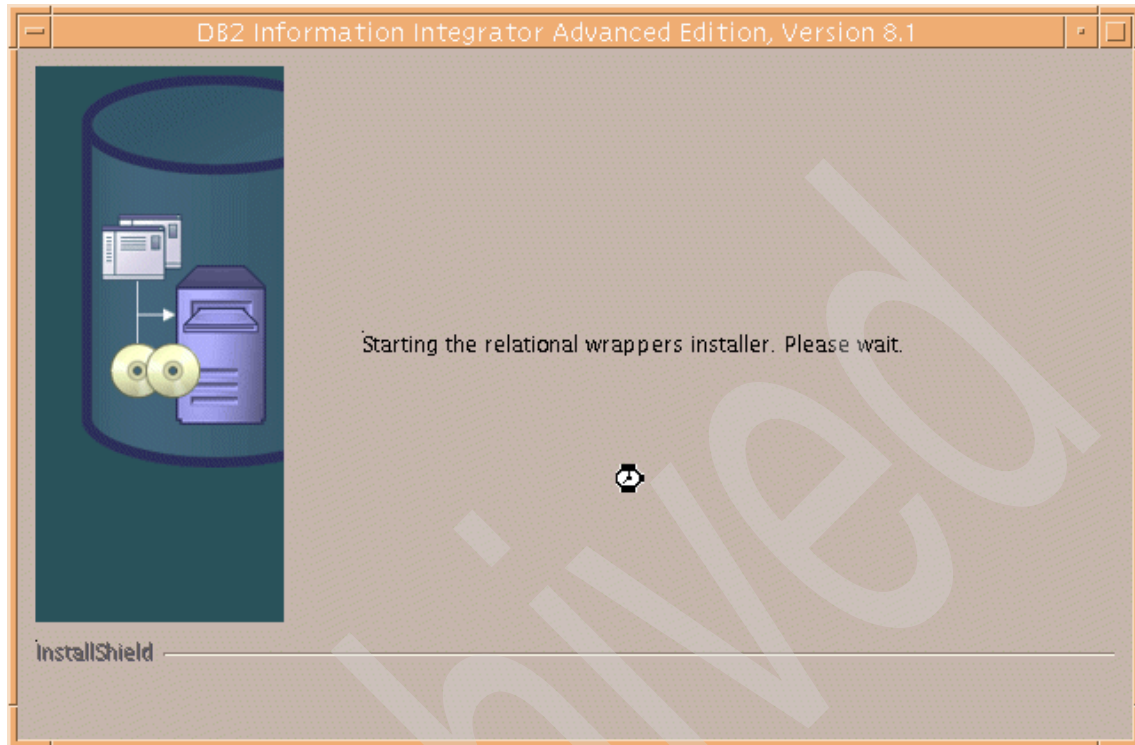


Figure 5-4 Source of relational wrapper installation

Click **Next** to continue.

- Start the relational wrappers installer.

The installation program issues the relational wrappers installer as displayed in Figure 5-5.



*Figure 5-5 Start the relational wrappers installer*

After the relational wrapper installer is being loaded, the next dialog will be launched automatically.

► Relational Wrappers - Setup

Figure 5-6 shows how you can refer to the installation prerequisites and release notes for the most current information on the relational wrappers installation. Installation prerequisites inform you about the minimum software, memory, and disk requirements by providing you with links to related documents.

We recommend that you browse through these documents since current and important information may be documented.



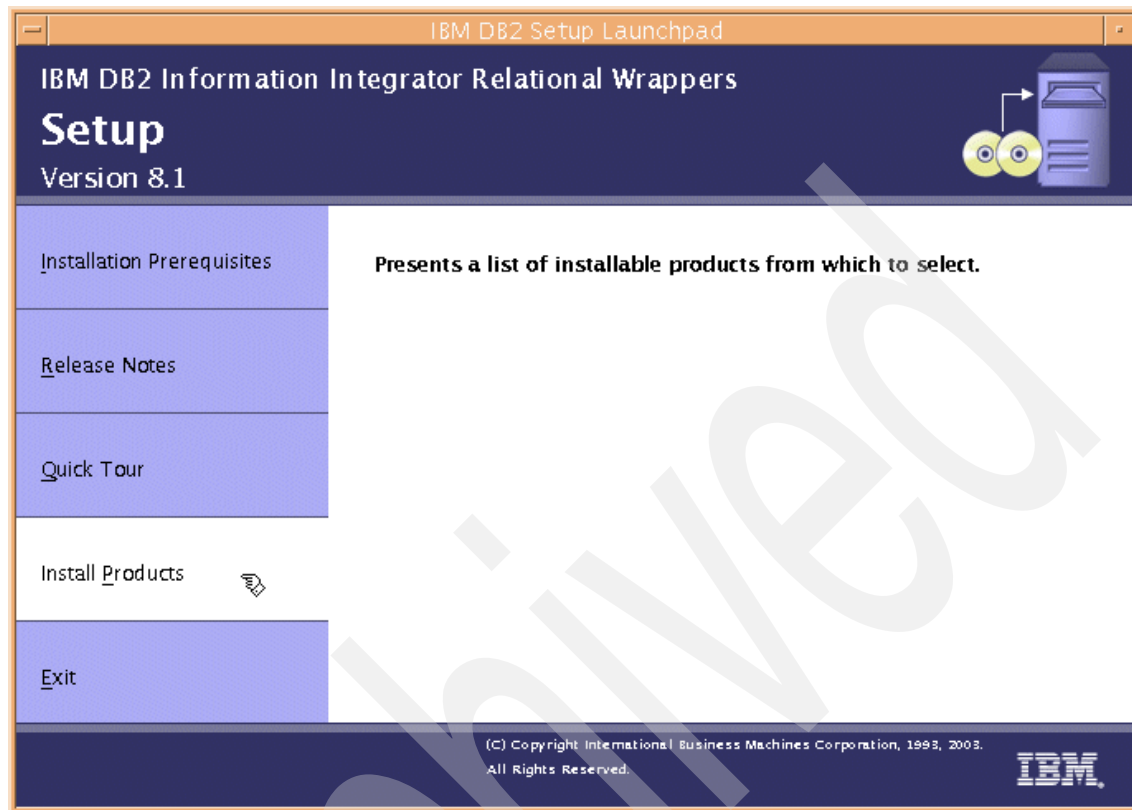


Figure 5-6 Relational wrappers setup

Click **Install Products** to continue.

- Under **Relational Wrappers** select **products**.

In Figure 5-7, select the only option: **DB2 Information Integrator Relational Wrappers**

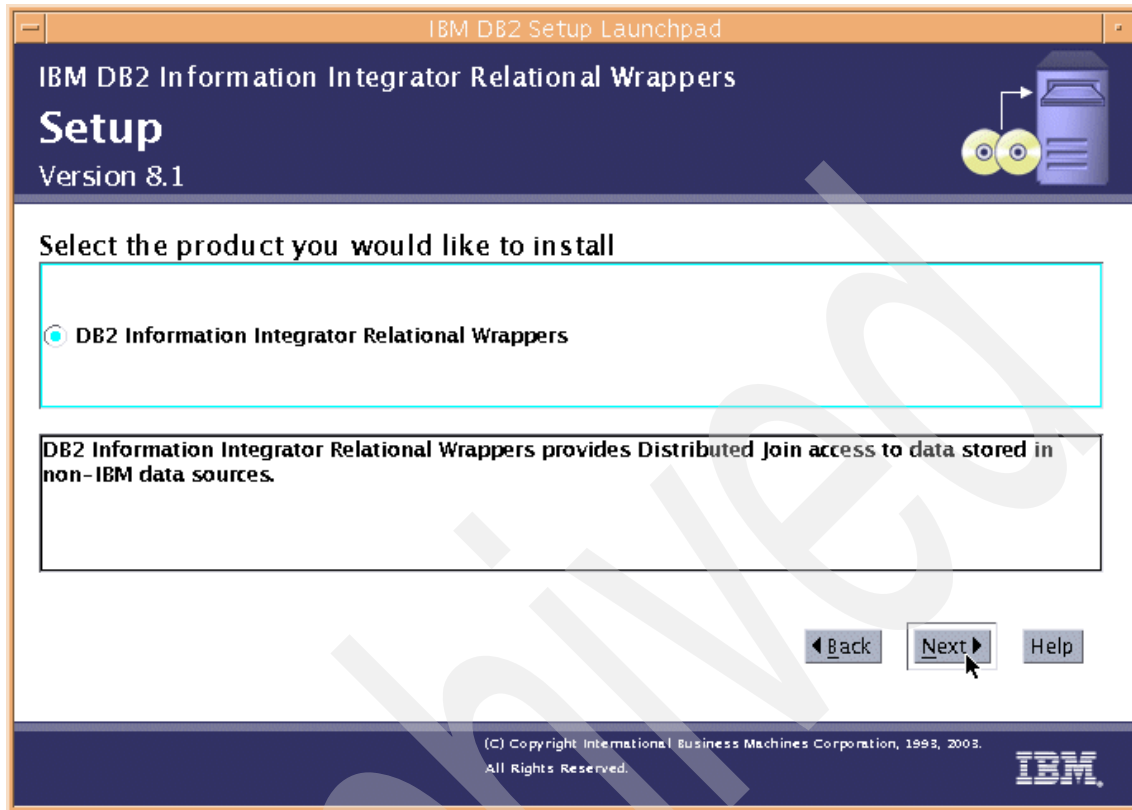


Figure 5-7 Relational wrappers setup - Select products

Click **Next** to continue.

- Relational Wrappers - DB2 Setup Wizard - Introduction

Figure 5-8 introduces the relational wrappers installation procedure. This DB2 setup wizard installs the DB2 Information Integrator Relational Wrappers on your computer.

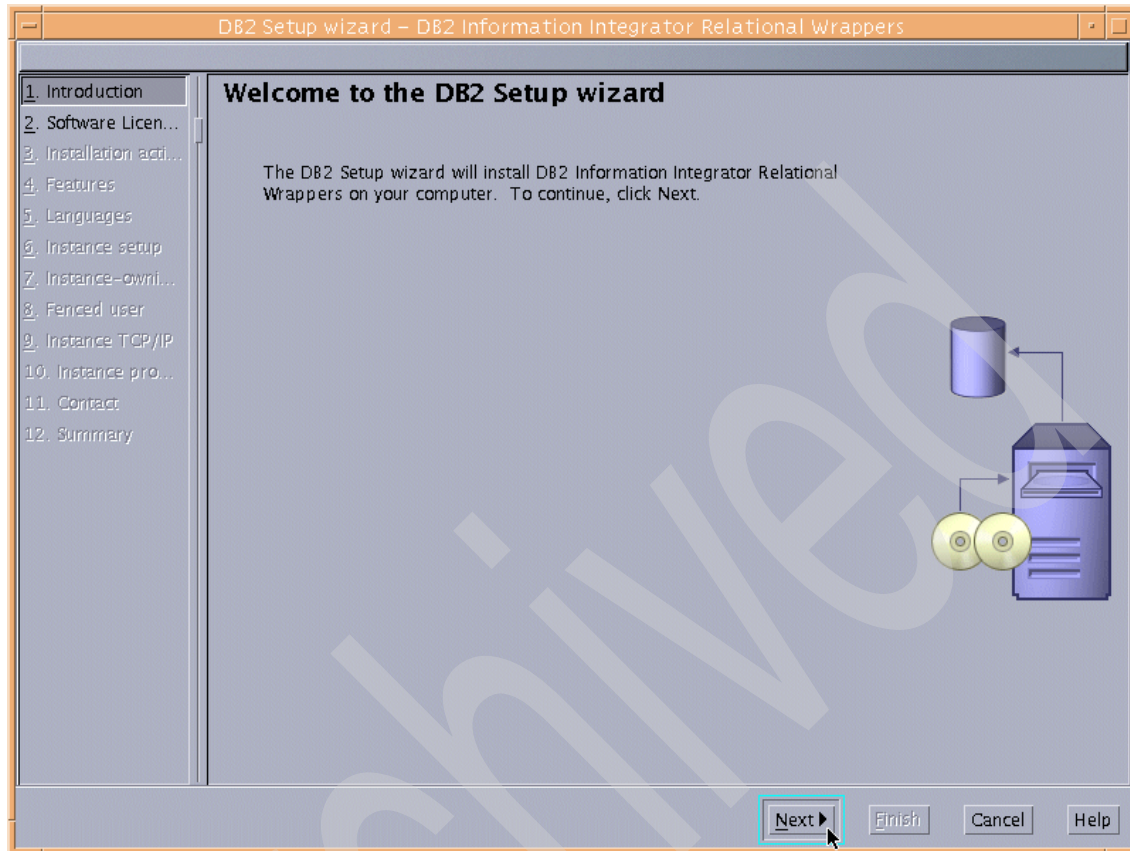


Figure 5-8 Relational wrappers setup - Introduction

Click **Next** to continue.

► Relational Wrappers - DB2 Setup Wizard - Software License Agreement

Please read the license agreement before you proceed. You need to **Accept** the Agreement as shown in Figure 5-9, prior to proceeding further.

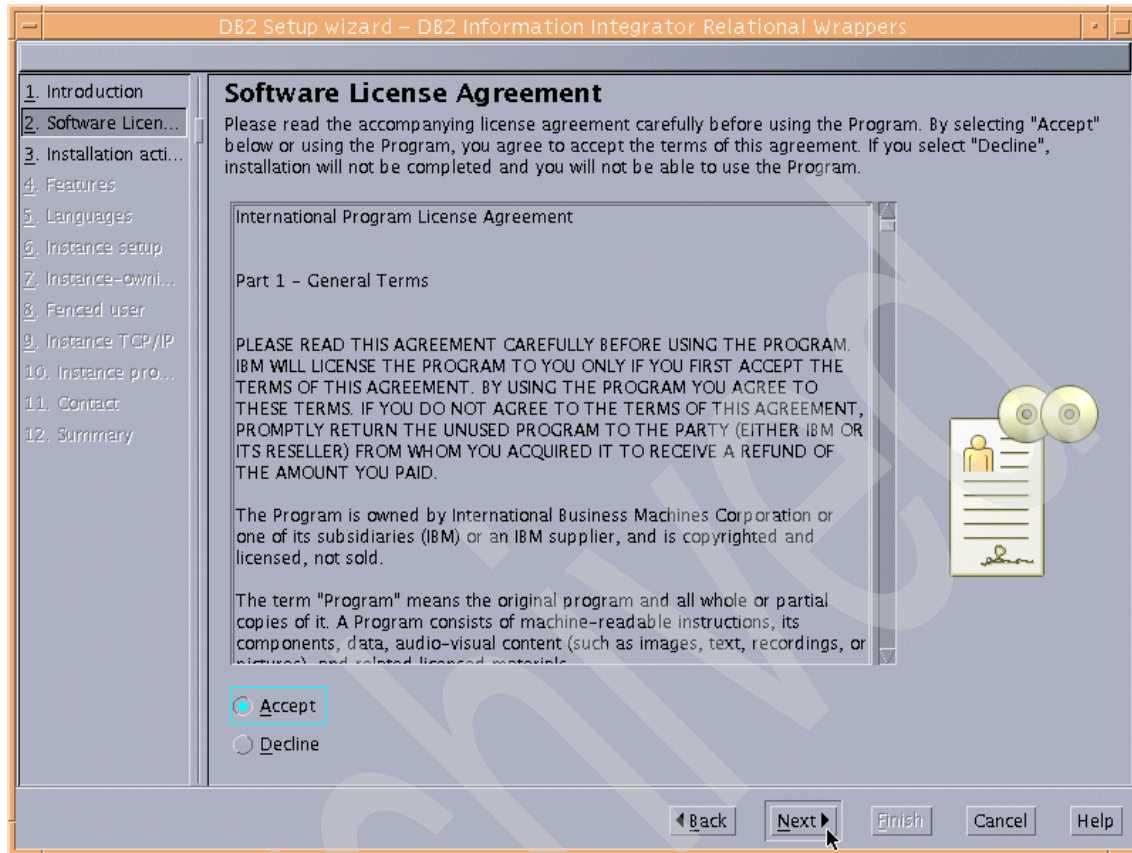


Figure 5-9 Relational wrappers setup - SW License Agreement

Click **Next** to continue.

► Relational Wrappers - DB2 Setup Wizard - Select installation action

Select the check box for installing the relational wrappers on your machine as shown in Figure 5-10.

Select **Save your settings in a response file** if you want to record your interactions with the installation process in a text file. Take a look to 5.2.2 “Installation response file examples” on page 157 to see sample response files.

**Tip:** You can automate the setup of DB2 Information Integrator by giving a response file as an argument for iisetup.bin.



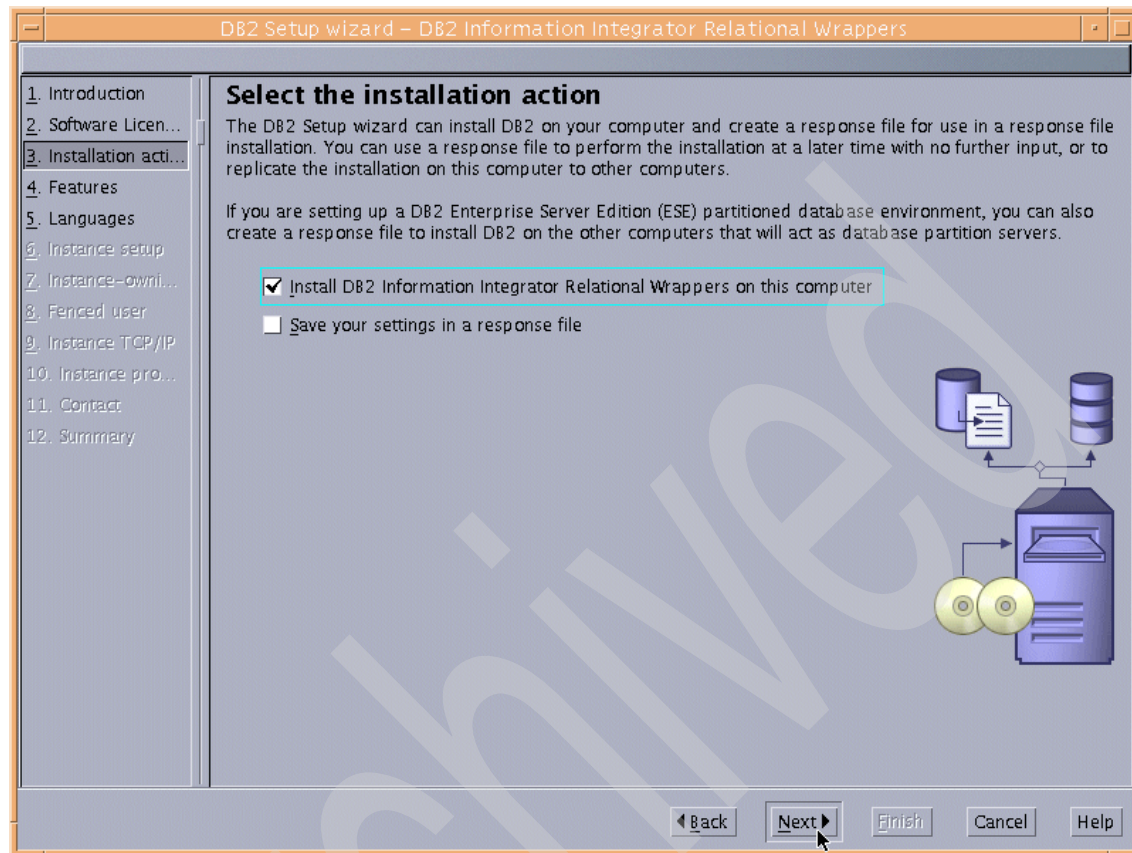


Figure 5-10 Relational wrappers setup - Select installation action

Click **Next** to continue.

► Relational Wrappers - DB2 Setup Wizard - Features to install

Select the data sources you want to connect to from DB2 Information Integrator as shown in Figure 5-11. Besides the well known wrapper libraries supported for relational sources, you need to choose **custom data source support** to enable the possibility of adding new wrappers by FixPak.

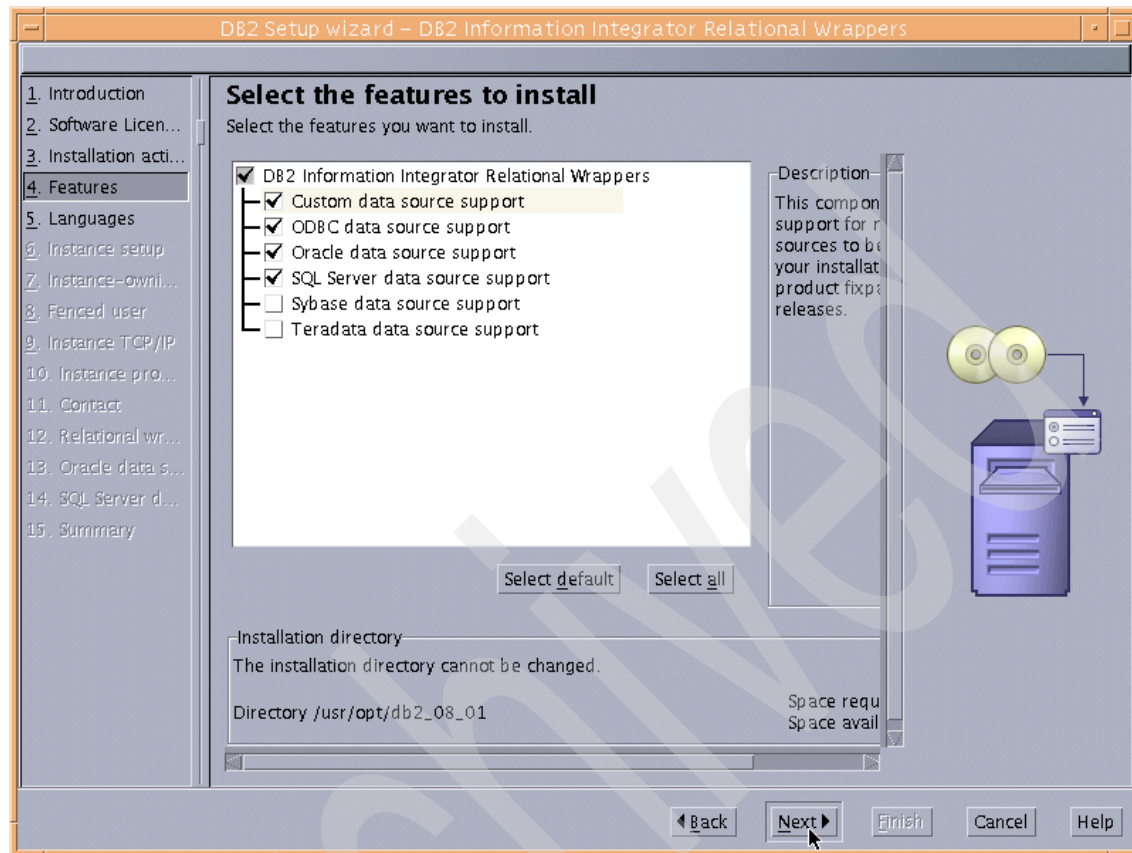


Figure 5-11 Relational wrappers setup - Features to install

Click **Next** to continue.

► Relational Wrappers - DB2 Setup Wizard - Languages

Install and uninstall any languages available from the Available languages and Selected languages list by selecting and moving the setup wizard back and forth with the respective buttons, as displayed in Figure 5-12.

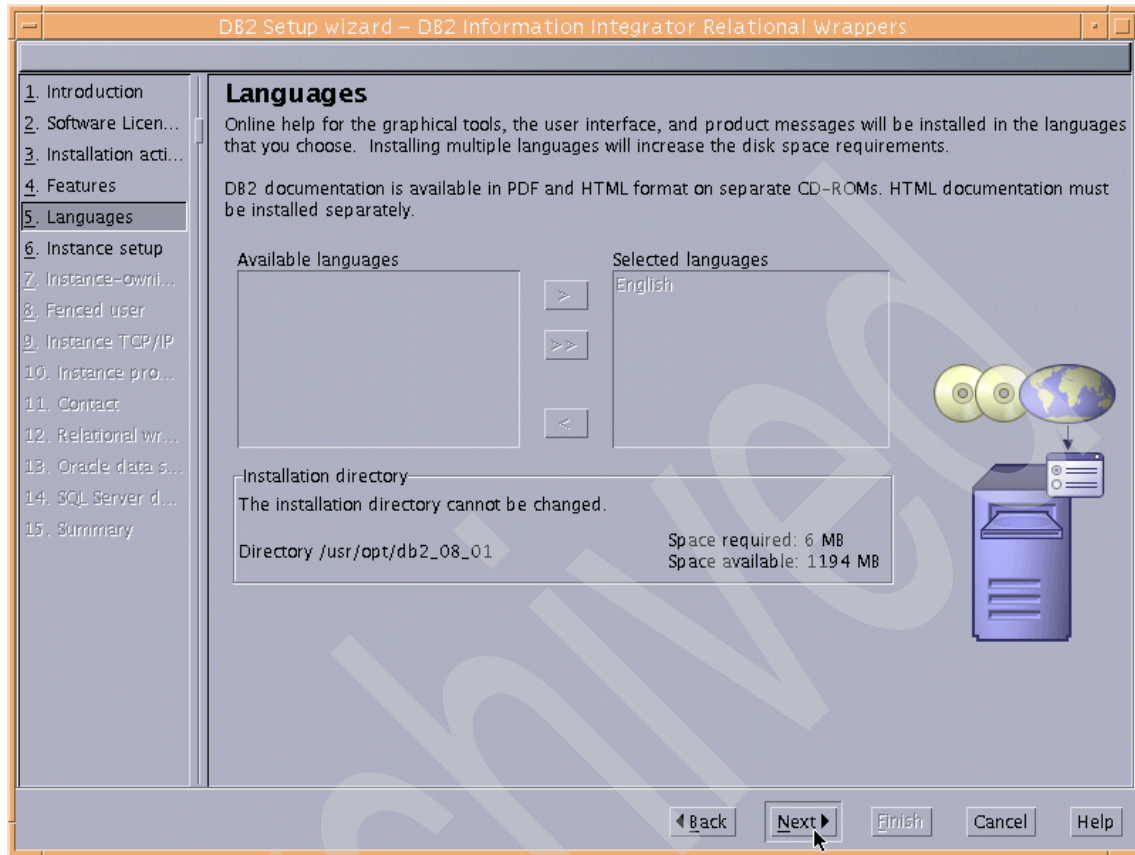


Figure 5-12 Relational wrappers setup - Languages

Click **Next** to continue.

► Relational Wrappers - DB2 Setup Wizard - Setup DB2 instance

Figure 5-13 offers different possibilities if you wish to install relational wrappers in new or existing instances.

Choose **Create a DB2 instance** 32-bit or 64-bit to create a new DB2 instance on your server. To install relational wrappers in an existing instance, choose **Configure new function for an existing DB2 instance** and select the right instance. If you want to defer the installation of relational wrappers, choose **Do not create a DB2 instance**. Or, if you want to install relational wrappers on another instance run **db2i setup** after the installation of DB2 Information Integrator.



**Attention:** Please refer to the documentation about the supported operating systems and data sources with 32-bit and 64-bit instances.

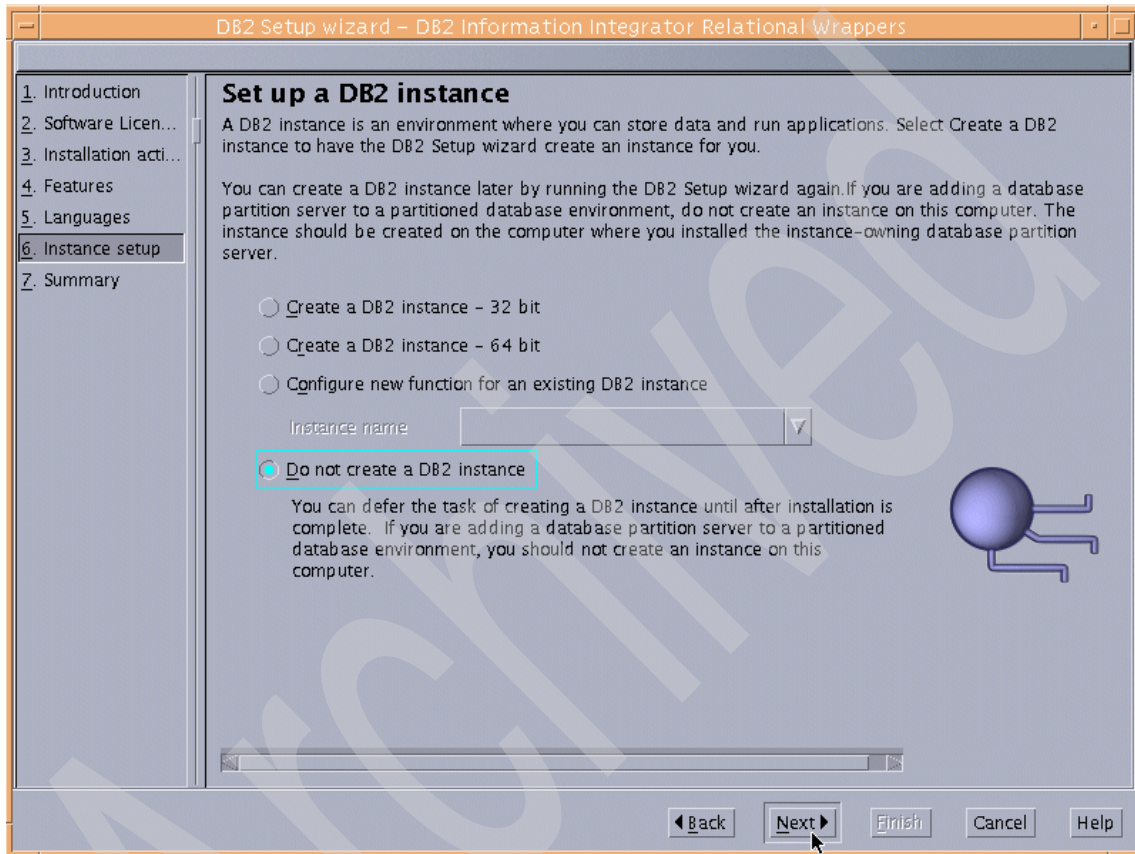


Figure 5-13 Relational wrappers setup - Setup DB2 instance

Click **Next** to continue.

► Relational Wrappers - DB2 Setup Wizard - Summary

The following Figure 5-14 displays your settings during the installation process, and provides the possibility to check inputs.



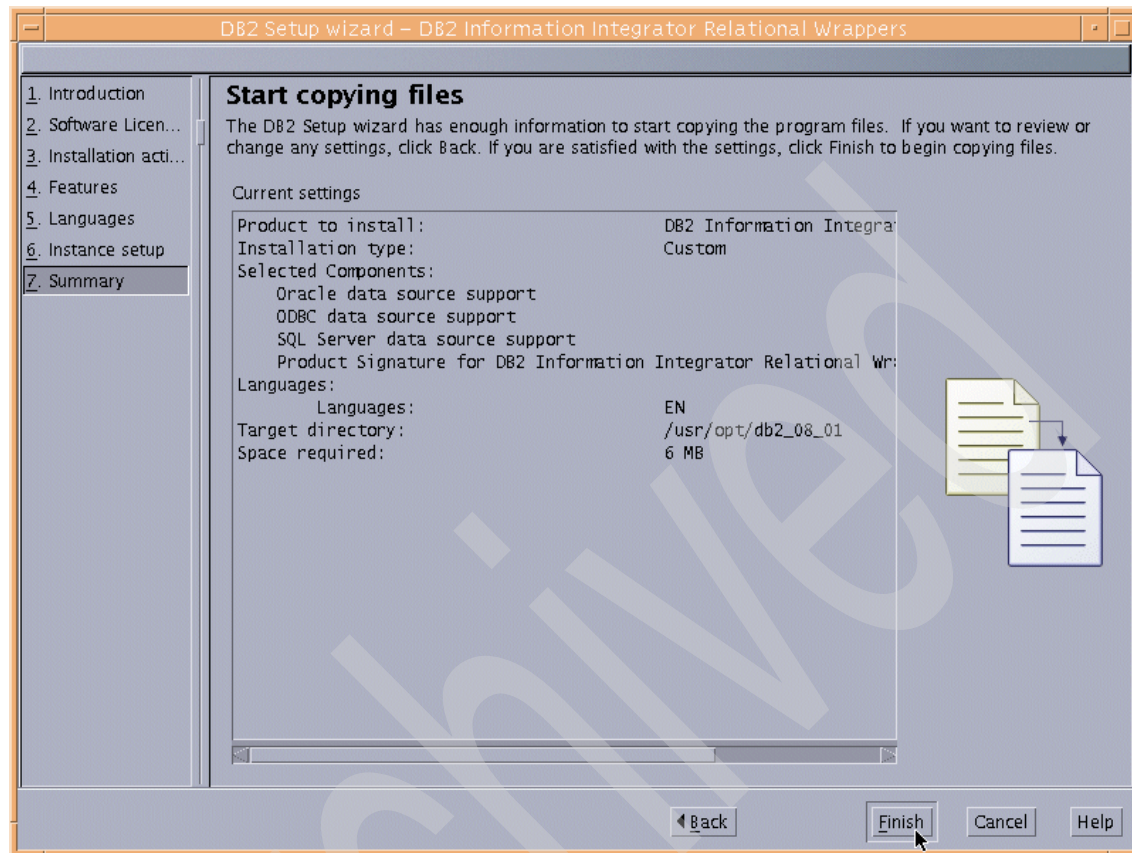


Figure 5-14 Relational wrappers setup - Summary

Click **Finish** to proceed.

► Relational Wrappers - DB2 Setup Wizard - Installation progress

Figure 5-15 shows if the actual installation of all selected relational wrappers is performed. Depending on which instance you chose to update, the installation procedure will update your instance.

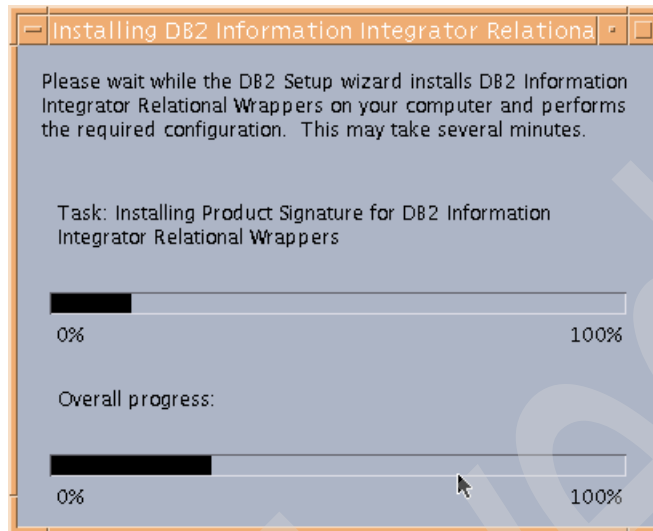


Figure 5-15 Relational wrappers setup - Installation progress

After installing all selected relational wrappers, the next dialog will be launched automatically.

► Relational Wrappers - DB2 Setup Wizard - Summary

Check the success of the installation by viewing the **Status report** displayed in Figure 5-16. Scroll through the list and watch out for any errors. In the case of failures, you will need to consult the documentation, or refer to 5.10 “Troubleshooting” on page 235.

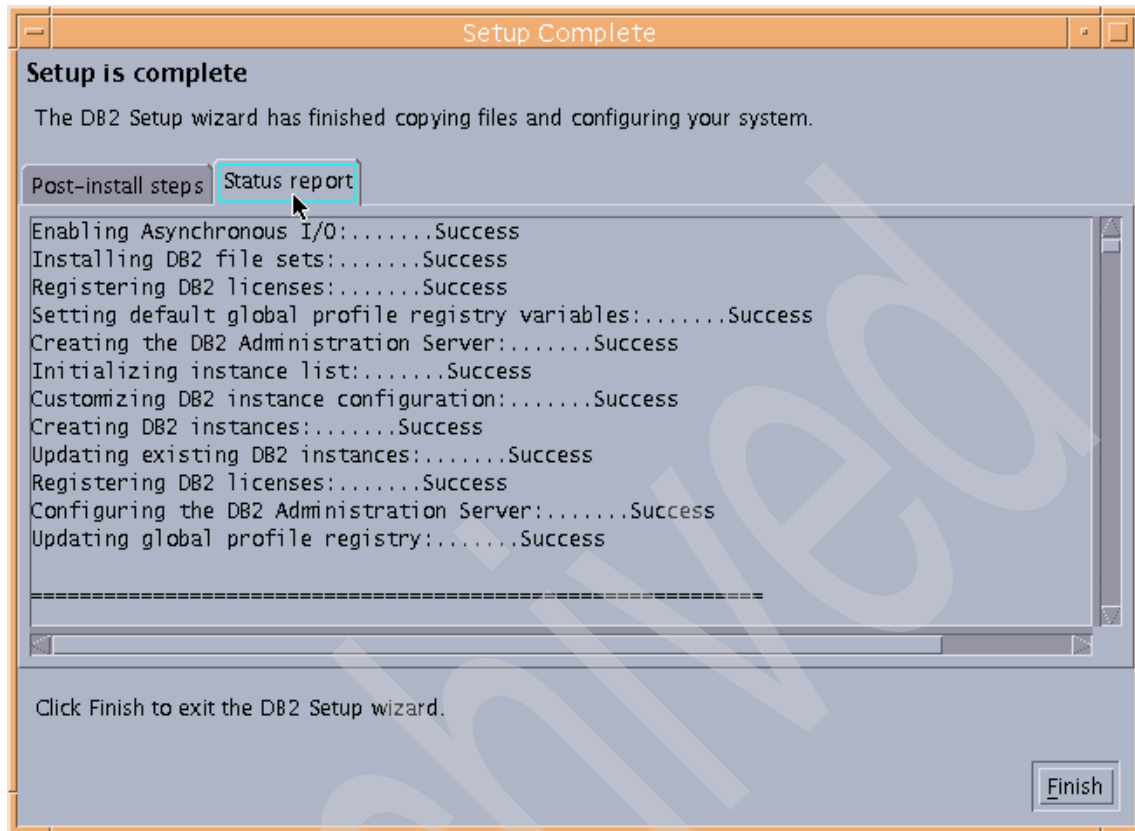


Figure 5-16 Relational wrappers setup - Complete

Click **Finish** to proceed.

► Relational Wrappers - DB2 Setup Wizard - Success

With Figure 5-17 the installation of the relational wrappers is finished.

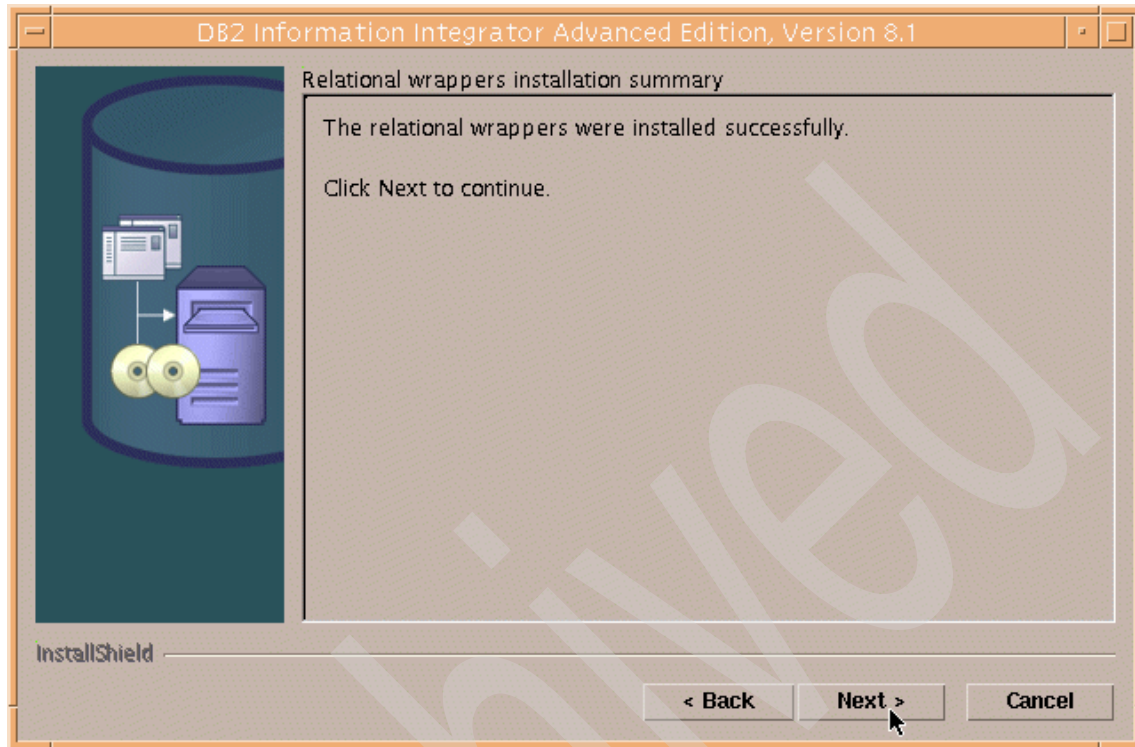


Figure 5-17 Relational wrappers setup - Status

Click **Next** to continue.

- Source of non relational wrapper installation

Browse the CD or any other file system to select the path for the relation wrappers source, as shown in Figure 5-18.

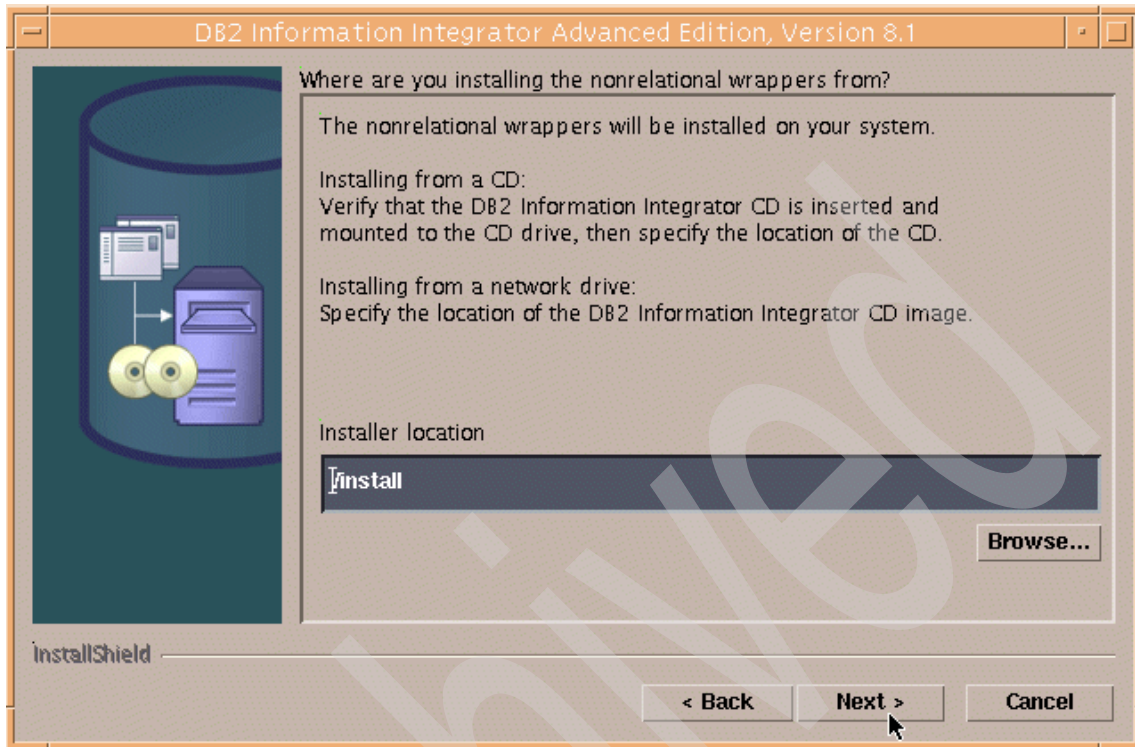


Figure 5-18 Source of non relational wrapper installation

Click **Next** to continue.

- Start the non relational wrappers installer.

The installation program issues the non relational wrappers installer as displayed in Figure 5-19.

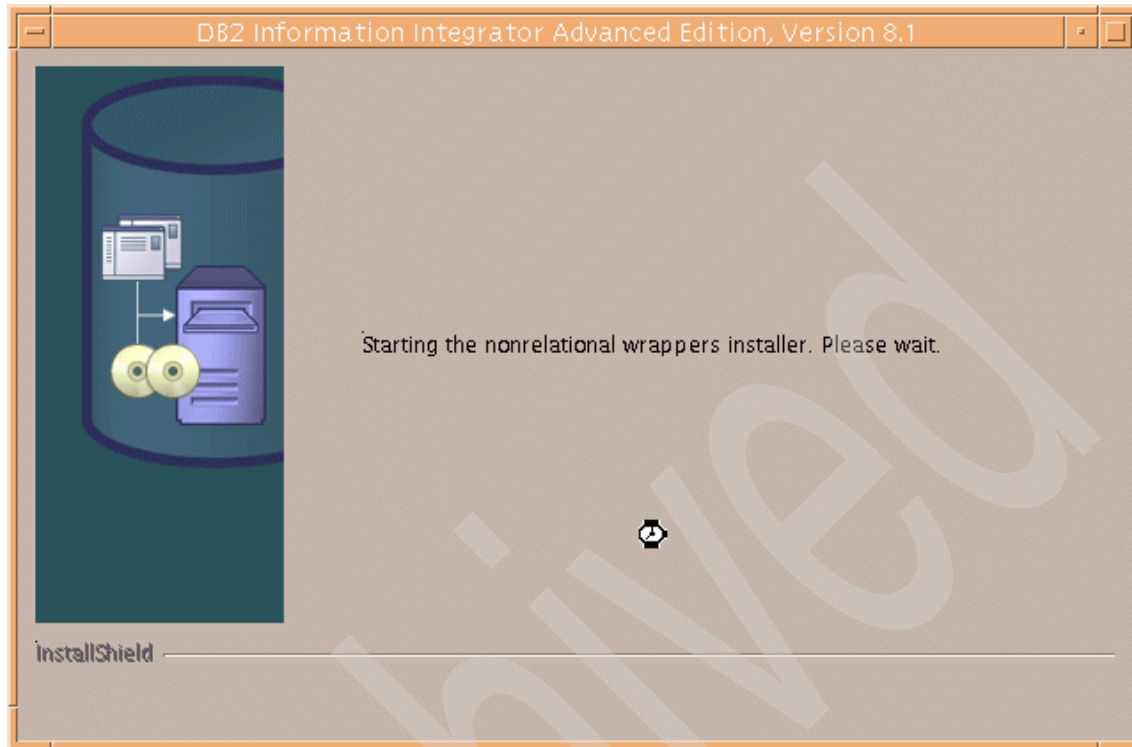


Figure 5-19 Start the non relational wrappers installer

After the non relational wrapper installer is being loaded, the next dialog is launched automatically

► Non relational Wrappers - Setup

Please check out Installation Prerequisites and Release Notes for the most current information on the non relational wrappers installation.

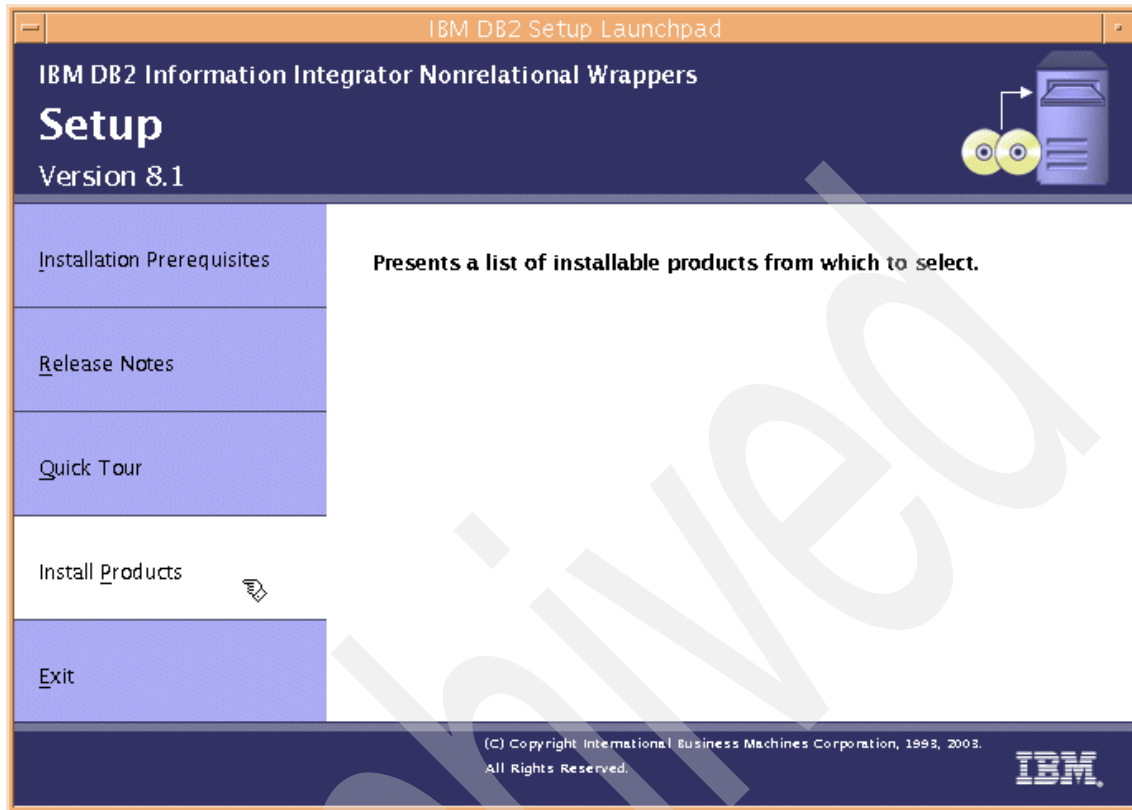


Figure 5-20 Non relational wrappers setup

Click **Install Products** to continue.

- Non relational Wrappers - Select products

In Figure 5-21 select the only option: **DB2 Information Integrator - Non Relational wrappers**, and please continue.

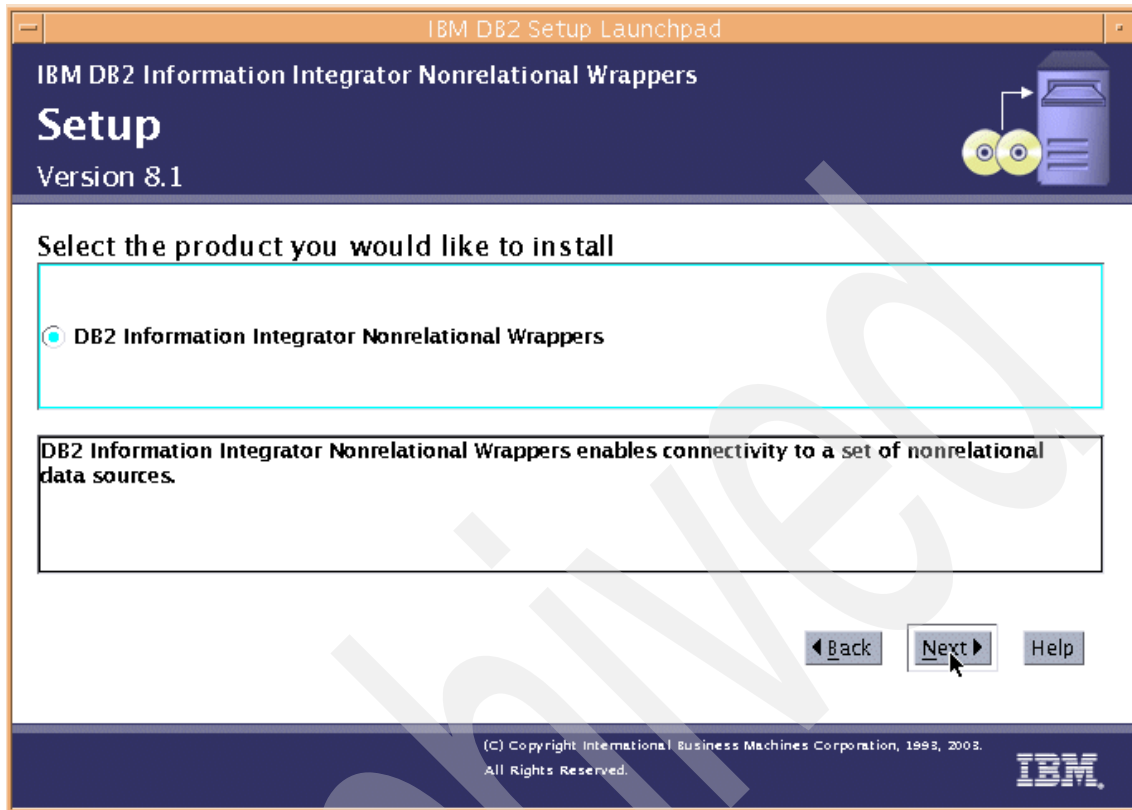


Figure 5-21 Non relational wrappers setup - Select products

Click **Next** to continue.

- ▶ Non relational Wrappers - DB2 Setup Wizard - Introduction

Figure 5-22 introduces the non relational wrappers installation procedure. This DB2 setup wizard installs the DB2 Information Integrator non relational wrappers on your computer.



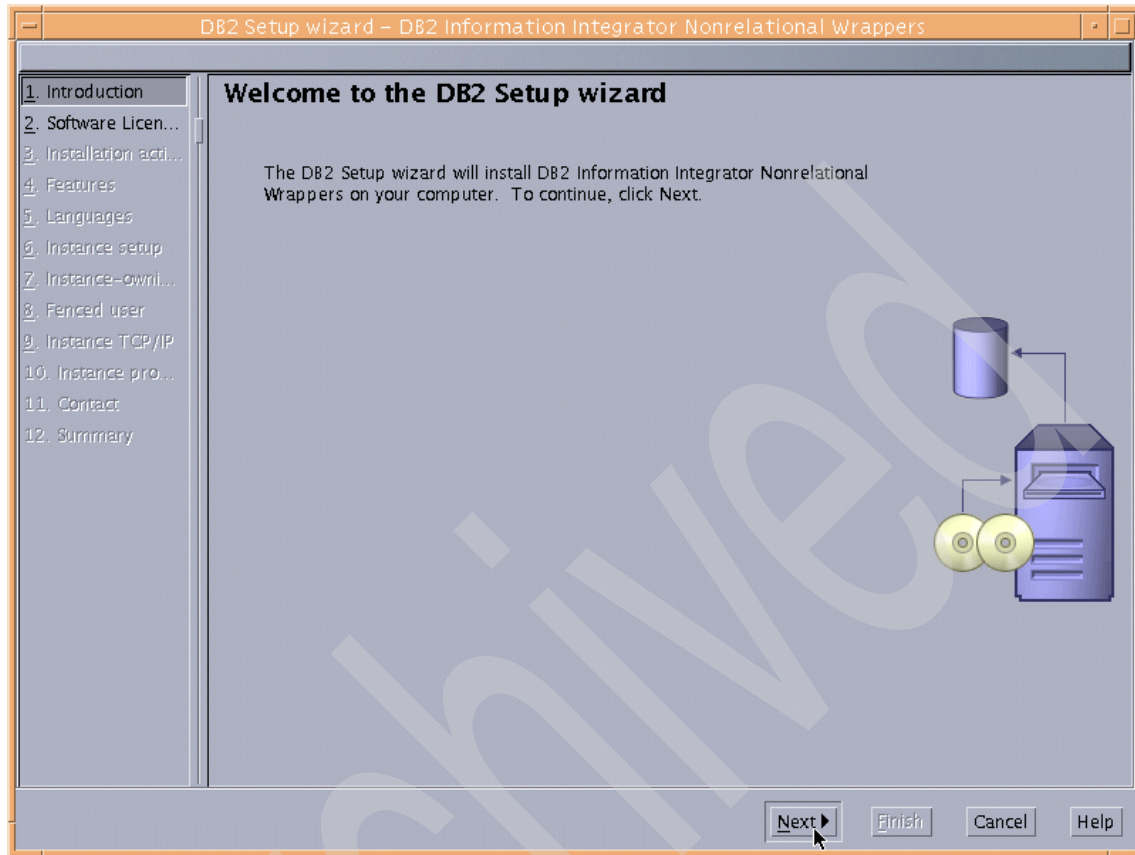


Figure 5-22 Non relational wrappers setup - Introduction

Click **Next** to continue.

- Non relational Wrappers - DB2 Setup Wizard - Software License Agreement  
Please read the license agreement before you proceed. You need to **Accept** the Agreement as shown in Figure 5-23.

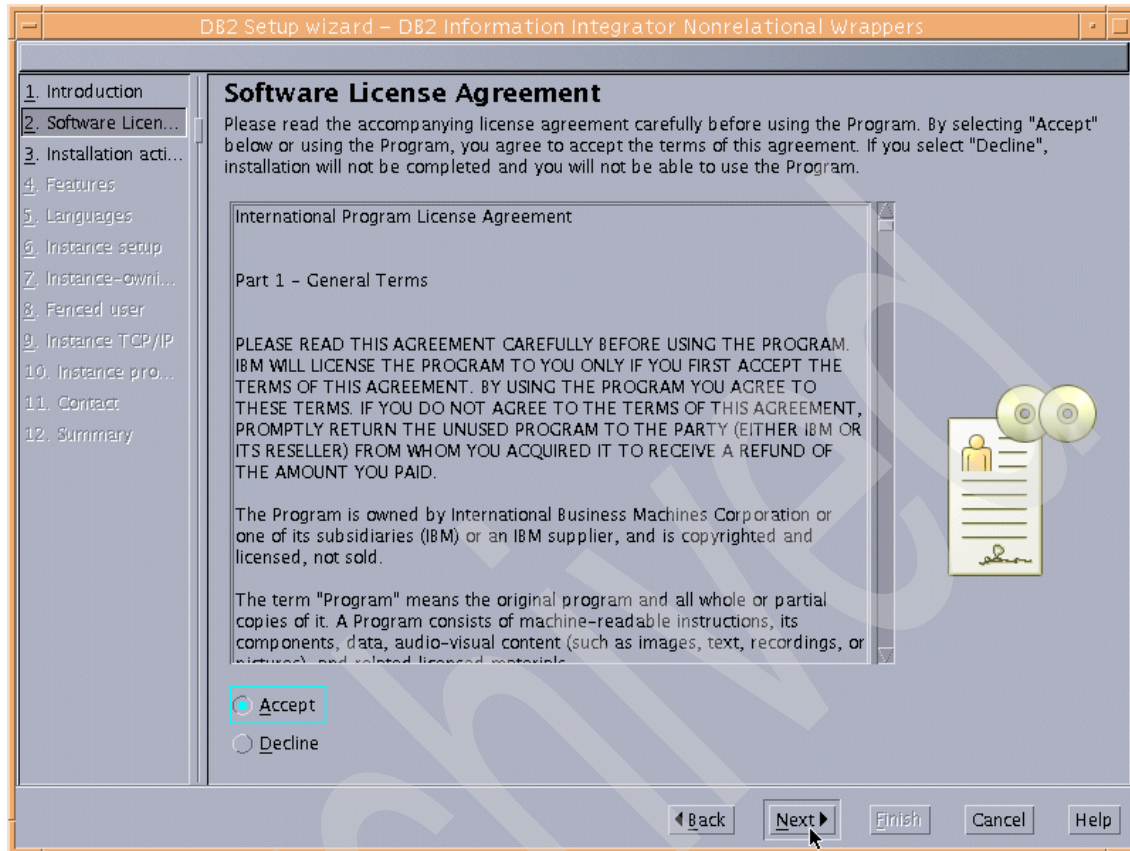


Figure 5-23 Non relational wrappers setup - SW License Agreement

Click **Next** to continue.

- Non relational Wrappers - DB2 Setup Wizard - Select installation action

Select the check box for installing the non relational wrappers on your machine as shown in Figure 5-24.

Select **Save your settings** in a response file if you want to record your interactions with the installation process in a text file. Take a look at 5.2.2 “Installation response file examples” on page 157 to see a response file.

**Tip:** You can automate the setup of DB2 Information Integrator by giving a response file as an argument for iisetup.bin.

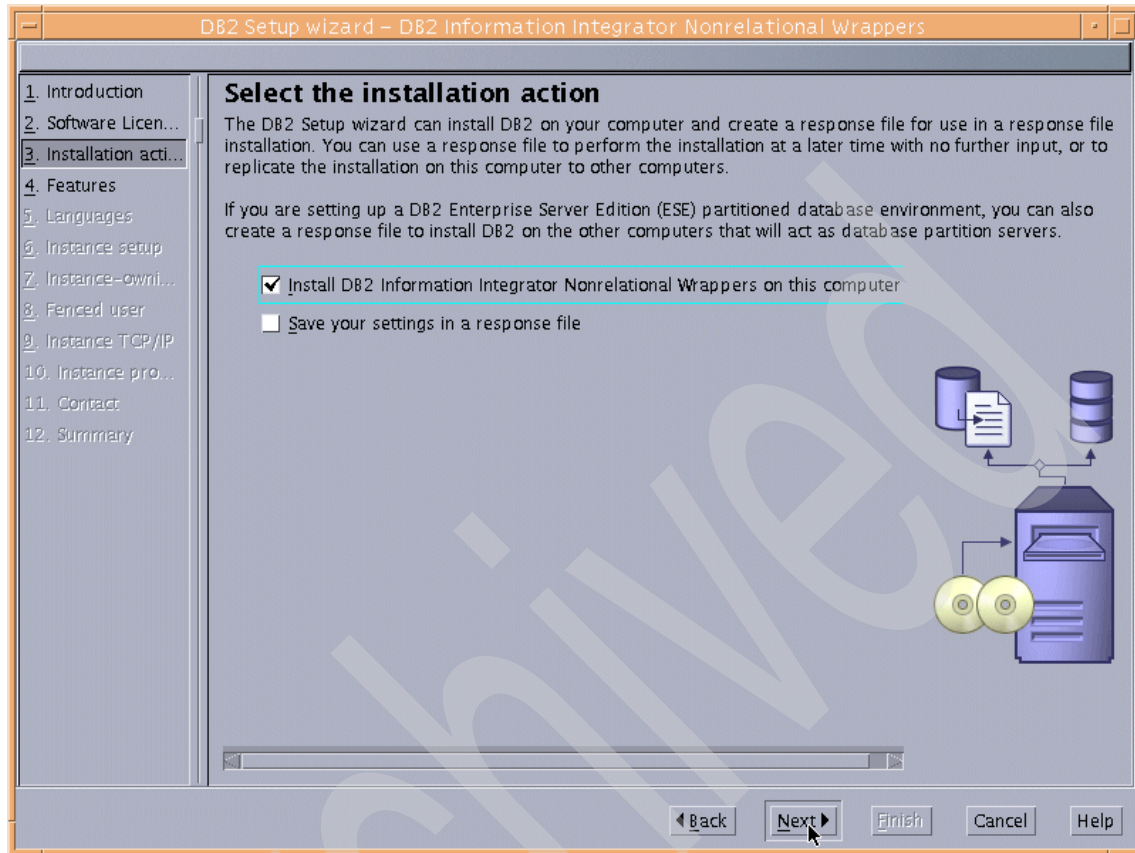


Figure 5-24 Non relational wrappers setup - Select install actions

Click **Next** to continue.

- Non relational Wrappers - DB2 Setup Wizard - Features to install  
Select the data sources you want to connect to from DB2 Information Integrator as shown in Figure 5-25.



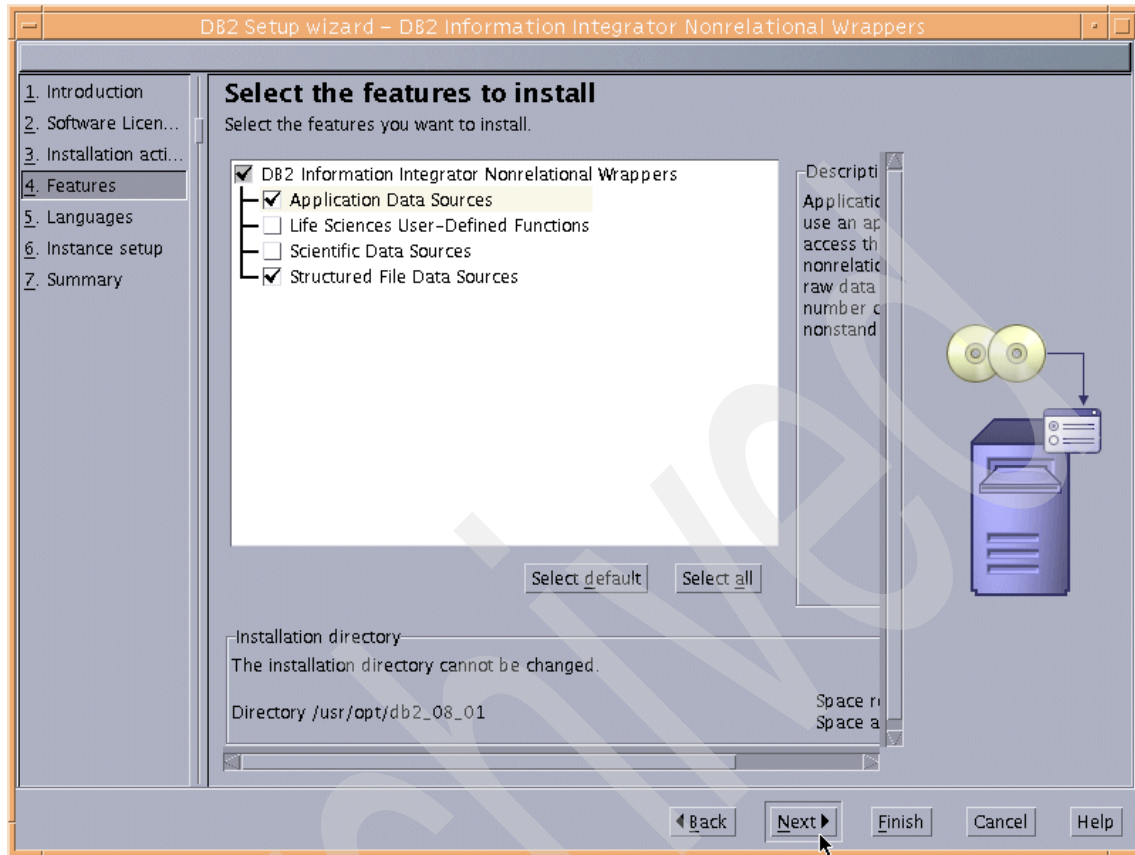


Figure 5-25 Non relational wrappers setup - Features to install

Click **Next** to continue.

► Non relational Wrappers - DB2 Setup Wizard - Languages

Install and uninstall any languages available from the Available languages and Selected languages list, by selecting and moving the setup wizard back and forth with the respective buttons as displayed in Figure 5-26.

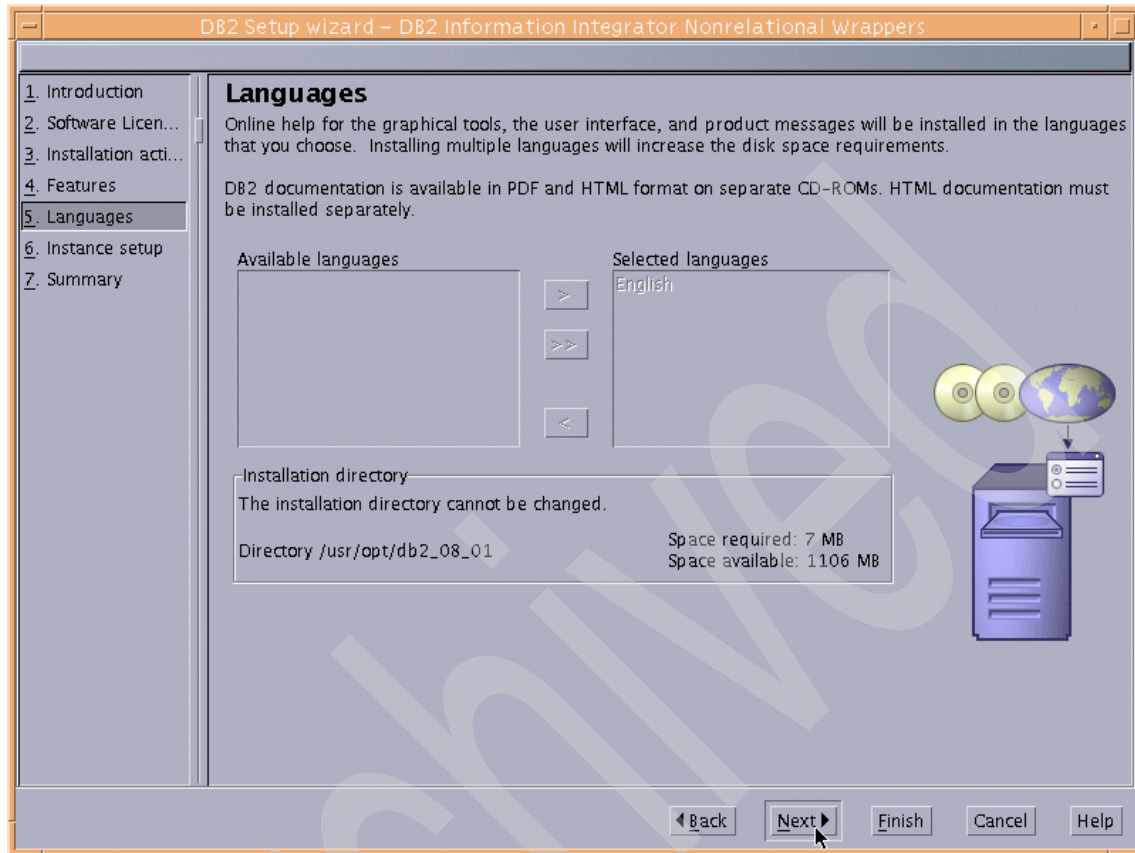


Figure 5-26 Non relational wrappers setup - Languages

Click **Next** to continue.

► Non relational Wrappers - DB2 Setup Wizard - Setup DB2 instance

Figure 5-27 offers different possibilities if you want to install non relational wrappers in new respective existing instances. Choose **Create a DB2 instance** 32-bit or 64-bit to create a new DB2 instance on your server. To install non relational wrappers in an existing instance, choose **Configure new function for an existing DB2 instance** and select the instance. If you want to defer the installation of non relational wrappers, choose **Do not create a DB2 instance**. Or, if you want to install non relational wrappers on another instance, run **db2i setup** after the installation of DB2 Information Integrator.

Please note the referenced Table 5-1 on page 124.

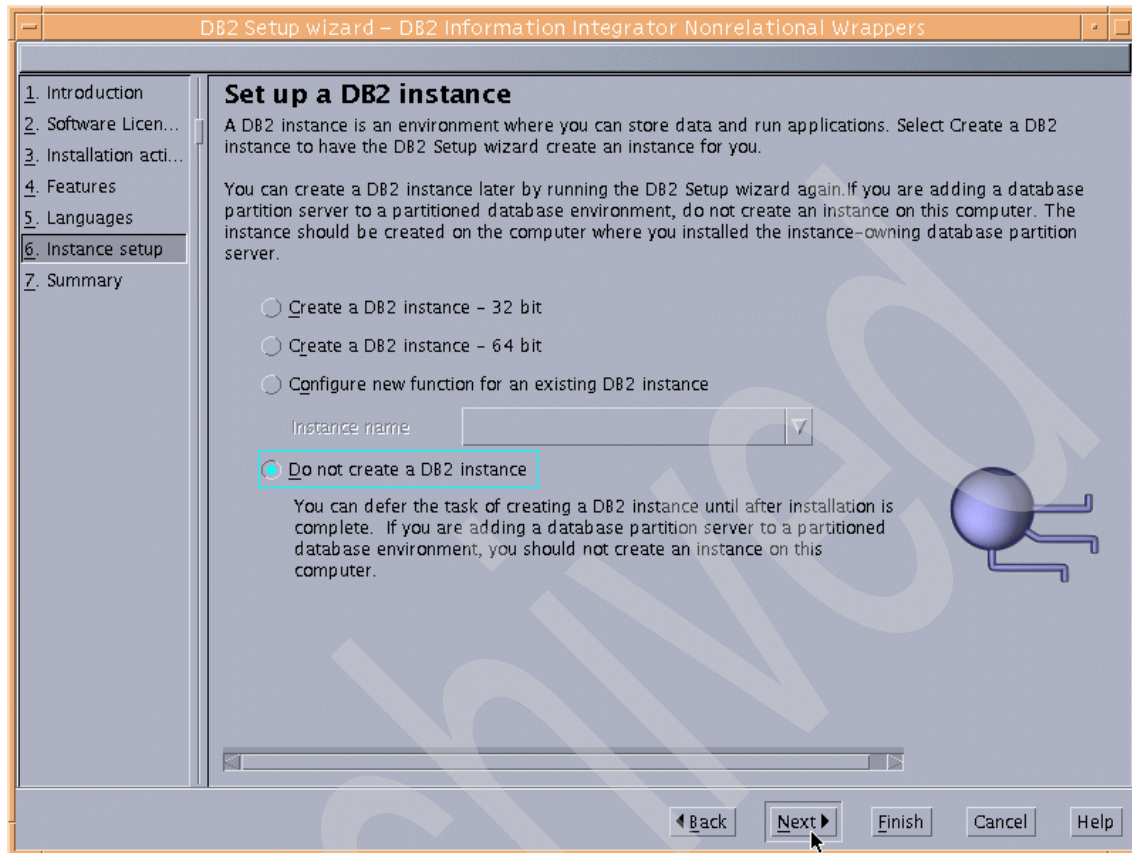


Figure 5-27 Non relational wrappers setup - Setup DB2 instance

Click **Next** to continue.

► Non relational Wrappers - DB2 Setup Wizard - Summary

Figure 5-28 displays your settings during the installation process, and provides the possibility to check inputs.



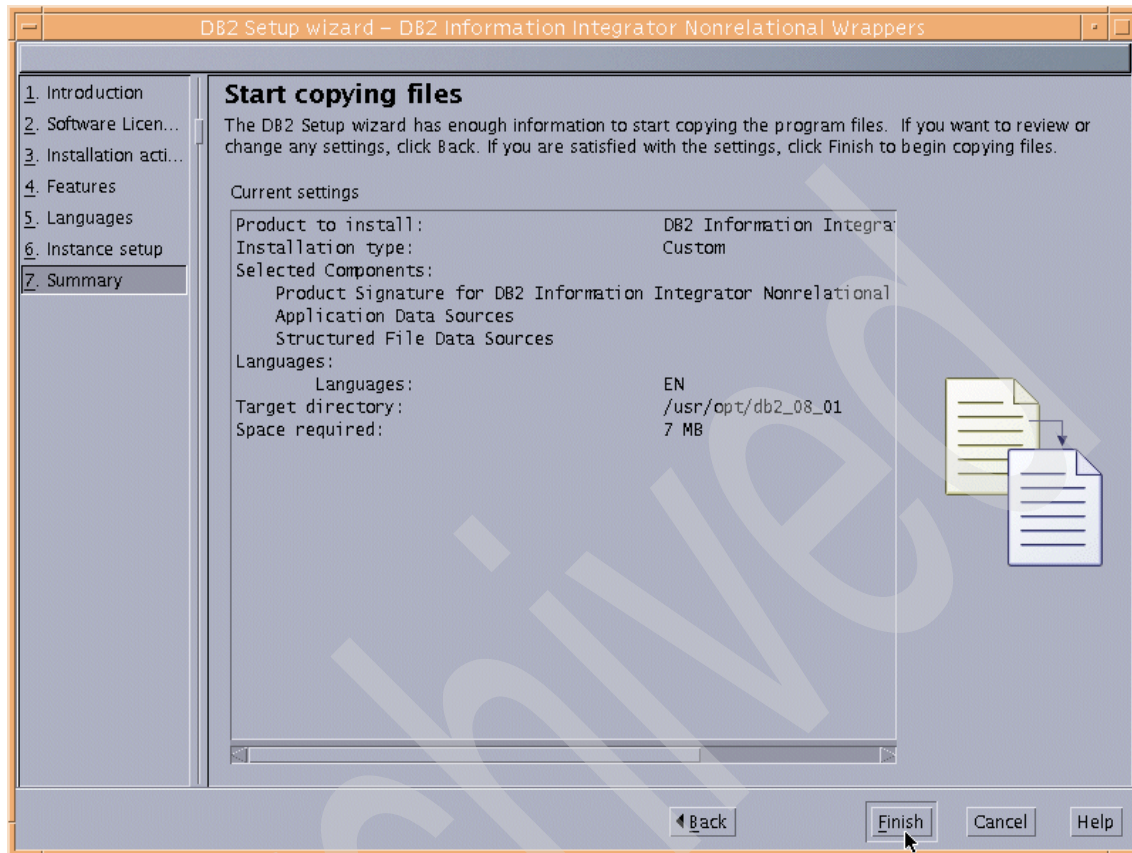
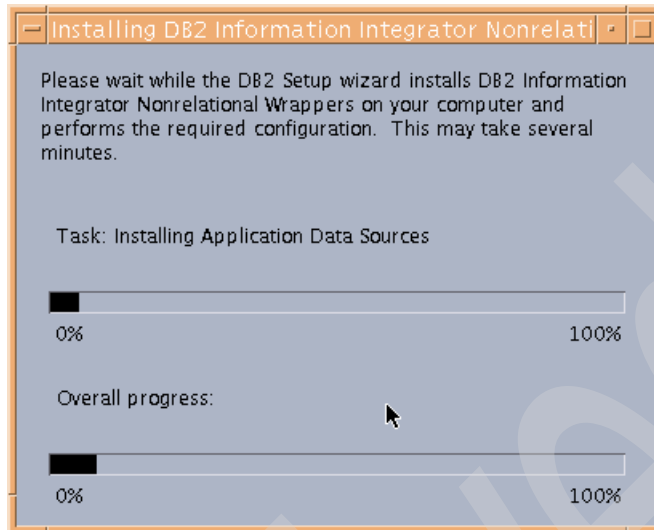


Figure 5-28 Non relational wrappers setup - Summary

Click **Next** to continue.

- Non relational Wrappers - DB2 Setup Wizard - Installation progress

Figure 5-29 shows the actual installation of all selected non relational wrappers that is performed. Depending on which instance you chose to update, the installation procedure will update your instance.



*Figure 5-29 Non relational wrappers setup - Installation progress*

After installing all selected relational wrappers, the next dialog will be launched automatically.

► Non relational Wrappers - DB2 Setup Wizard - Summary

Check the success of the installation by viewing the **Status report** displayed in Figure 5-30. Scroll through the list and be aware should any failure occur. Action for failures: Consult the documentation, or refer to 5.10 “Troubleshooting” on page 235.



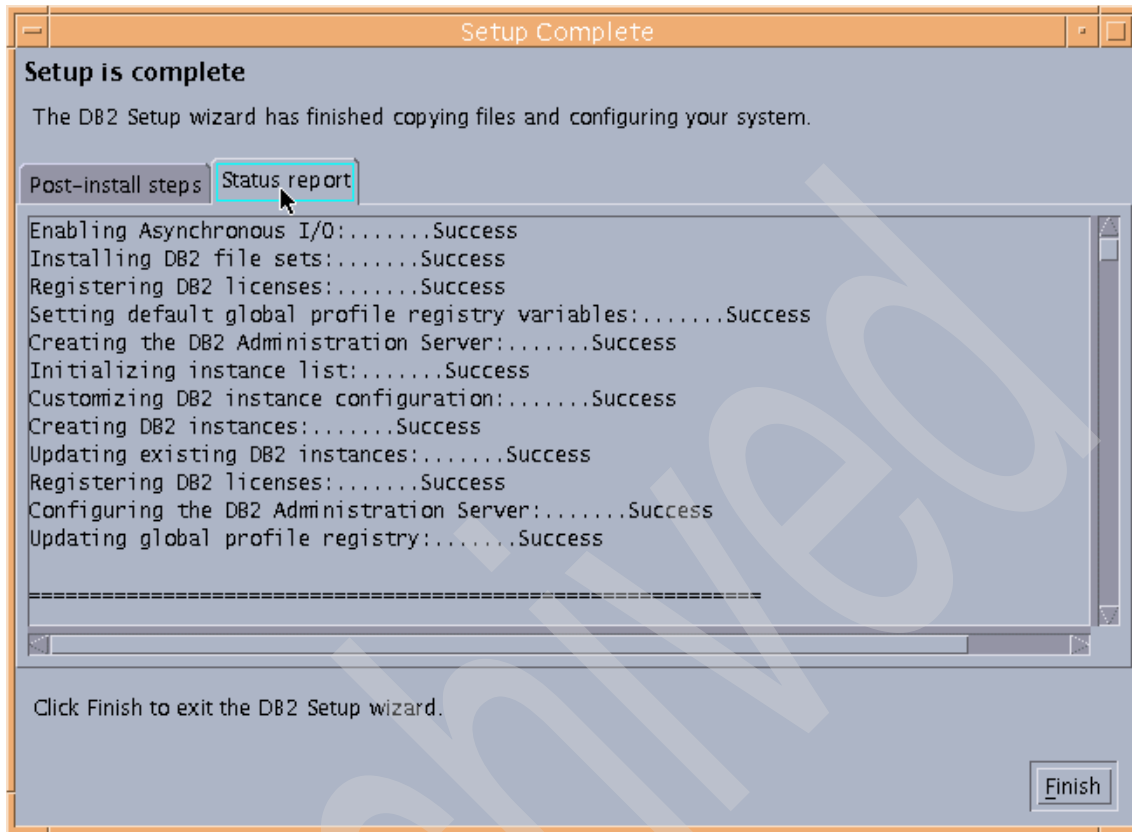


Figure 5-30 Non relational wrappers setup - Status

Click **Finish** to proceed.

► Non relational Wrappers - DB2 Setup Wizard - Success

With Figure 5-31 the installation of the non relational wrappers is finished.

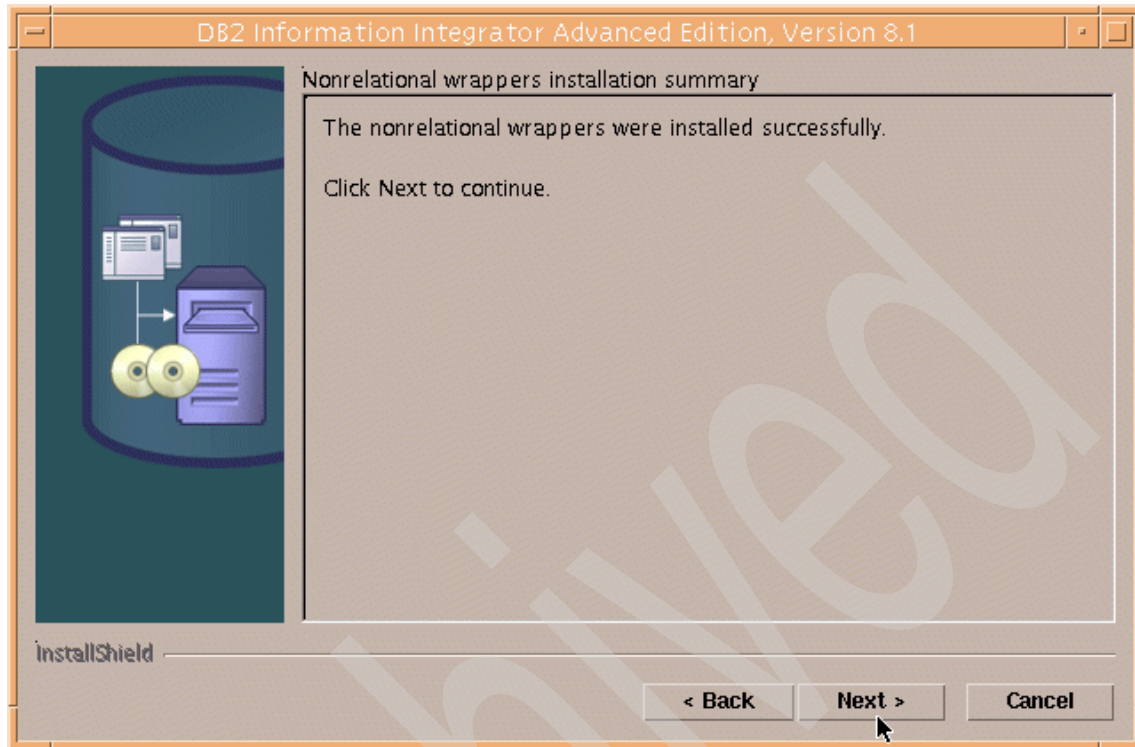


Figure 5-31 Non relational wrappers setup - Complete

Click **Next** to continue.

- DB2 Information Integrator Installation finished

Figure 5-32 shows the last screen in the installation process of DB2 Information Integrator.

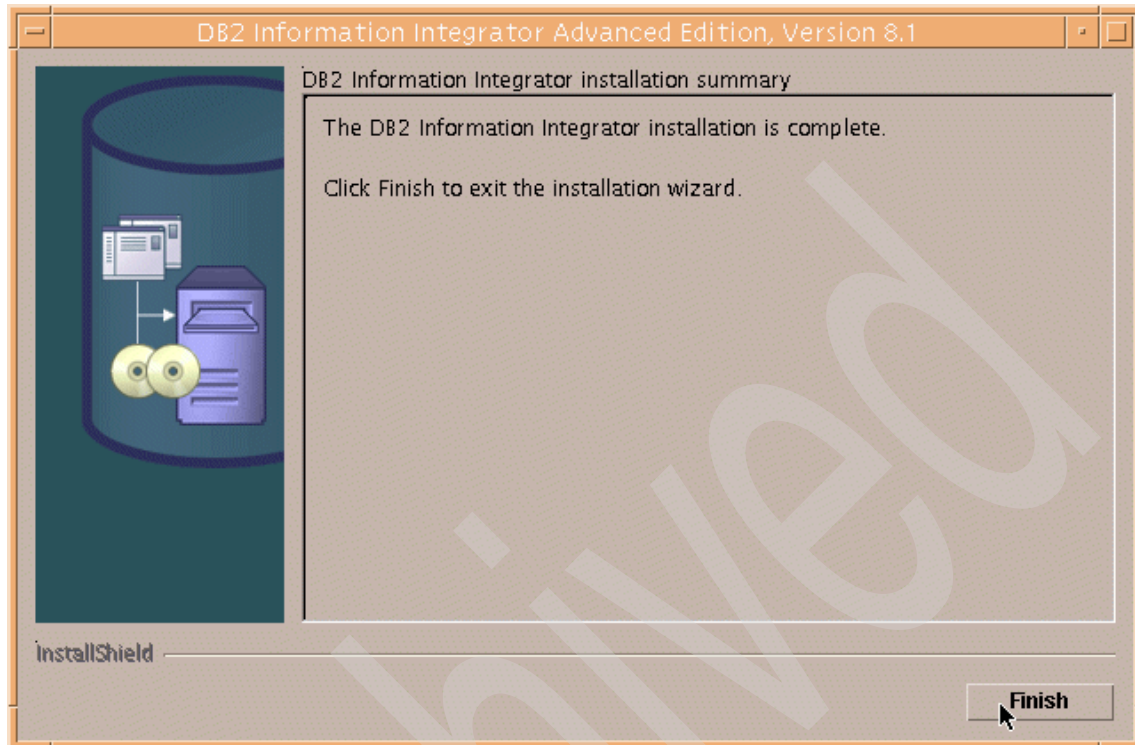


Figure 5-32 DB2 Information Integrator installation - Complete

Click **Finish** to end the installation.

## 5.2.2 Installation response file examples

Installation response files can be generated during the installation of DB2 Information Integrator to enable you to repeat an identical installation on another machine. Every user interaction is stored in the file using tags. The installation program uses these tags to perform an automatic installation based on the response file. Example 5-1 shows the response file we created during the installation of our relational wrappers.

### Example 5-1 Response file - Installation of relational wrappers

```
*-----
* Generated response file used by the DB2 Setup wizard
* generation time: 7/14/03 6:29 PM
*-----
* Product Installation
LIC_AGREEMENT      = ACCEPT
```

```

PROD          = II_RELATIONAL_WRAPPERS
INSTALL_TYPE  = CUSTOM
COMP          = CUSTOM_DATA_SOURCE_SUPPORT
COMP          = ORACLE_DATA_SOURCE_SUPPORT
COMP          = SQL_SERVER_DATA_SOURCE_SUPPORT
COMP          = ODBC_DATA_SOURCE_SUPPORT
*-----
*  Das properties
*-----
* -----
*  Instance properties
* -----
INSTANCE      = inst1
inst1.TYPE    = ese
* Instance-owning user
inst1.NAME    = db2fed32
inst1.UID     = 115
inst1.GROUP_NAME = db2grp1
inst1.HOME_DIRECTORY = /home/db2fed32
inst1.AUTOSTART = YES
inst1.AUTHENTICATION = SERVER
inst1.FEDERATED = YES
* Oracle data source wrapper information
inst1.ORACLE_HOME = /oracle9ir2
* SqlServer data source wrapper information
inst1.SQLSERVER_ODBC_LIB_PATH = /datadirect/odbc4.2
inst1.SQLSERVER_ODBC_INI_PATH = /home/db2fed32/.odbc.ini
*-----
*  Installed Languages
*-----
LANG          = EN

```

---

Example 5-2 shows the response file that we created during the installation of our non relational wrappers.

---

*Example 5-2 Response file - Installation of non relational wrappers*

---

```

*-----
* Generated response file used by the DB2 Setup wizard
* generation time: 7/15/03 4:37 PM
*-----
*  Product Installation
LIC_AGREEMENT = ACCEPT
PROD          = II_NONRELATIONAL_WRAPPERS
INSTALL_TYPE  = CUSTOM
COMP          = LSDC_APPLICATIONS_WRAPPER
COMP          = LSDC_STRUCTURED_FILES_WRAPPER
*-----
*  Das properties

```

```

* -----
* -----
* Instance properties
* -----
INSTANCE      = inst1
inst1.TYPE     = ese
* Instance-owning user
inst1.NAME     = db2fed32
inst1.UID      = 115
inst1.GROUP_NAME = db2grp1
inst1.HOME_DIRECTORY = /home/db2fed32
inst1.AUTOSTART = YES
inst1.AUTHENTICATION = SERVER
inst1.FEDERATED = YES
* -----
* Installed Languages
* -----
LANG          = EN

```

---

### 5.2.3 Installation hints, tips, and techniques

We have grouped some tips related to the DB2 Information Integrator installation:

- ▶ If you have not already installed DB2 ESE 8.1 and applied FixPak 2, the iisetup dialog on the DB2 Information Integrator installation CD will detect this and indicate that DB2 ESE 8.1.2 is a pre-requisite. If DB2 ESE 8.1 has not been installed, you will need to unmount the Information Integrator CD, and mount the DB2 ESE 8.1 installation CD to install DB2 ESE 8.1.
- ▶ The DB2 ESE 8.1 install CD that comes with DB2 Information Integrator contains DB2 ESE 8.1.2 (FixPak 2 level): If you install DB2 ESE 8.1 from this CD, you will not have download and apply DB2 UDB 8.1 FixPak 2.
- ▶ You cannot run multiple installers concurrently.
- ▶ If the installer does not display, clear out the temporary directory. Maybe there are still some files from previous cancelled or failed installation processes. Directories, you might look for, are:
  - UNIX: /tmp/ii
  - Windows: %TEMP%\ii
- ▶ If you have problems with the license registration, the reason might be the Java permissions settings. If so, a manual license registration will become necessary (you will be warned with a message) before the installation can be continued.

- ▶ Messages of linking data source client software into DB2 installation directory are stored in: /usr/opt/db2\_08\_01/lib/djxlinkAAAA.out where AAAA indicates the data sources like Oracle, Informix, etc.
- ▶ These considerations apply to HP-UX only:
  - There is no long filename support.
  - The following steps are necessary to mount the CD:
    - i. Check, if the pfs\_mountd and pfsd processes are running:  
*ps -eaf*
    - ii. If the processes are not running, start it with:  
*/usr/sbin/pfs\_mountd &*  
*/usr/sbin/pfsd4 &*
    - iii. To get the device name, perform:  
*ioscan -fknC disk*
    - iv. To mount the device, perform:  
*/usr/sbin/pfs\_mount <device\_name> <mountpoint>*  
*Example: /usr/sbin/pfs\_mount /dev/dsk/c2t2d0 /mnt*
  - To unmount the device, execute following step:  
*/usr/sbin/pfs\_umount <mountpoint>*  
*Example: /usr/sbin/pfs\_umount /mnt*
- ▶ If you are facing any problem or errors installing DB2 Information Integrator, make sure you enable error logging, with: (depending on the platform):
  - Windows: *iiSetup.exe -is:log <log filename> -debug*
  - UNIX: *./iiSetup.bin -is:log <log filename> -debug*
    - *iisetaup\_debug.log* is created in the temporary directory. (/tmp, %TEMP%)
    - log filename is the complete path and filename
- ▶ djxlink (UNIX only):
  - DB2 Information Integrator needs to be linked with the non-DB2 relational data source client software. *djxlink* creates the wrappers from the wrapper input libraries and the data source client software.
  - DB2 Information Integrator install process can do the link if the data source client software is already installed, and the system administrator knows the location of the data source client in the file system:
    - Path to the Oracle base directory ORACLE\_HOME. This is the Oracle Client directory that contains sub-directories /lib /bin and /network as well as others.

- Path to DataDirect Connect sub-directory /lib (DJX\_LOAD\_LIBRARY\_PATH)
- After installing FixPaks, or if the install did not do the link, the system administrator (root) should use the **djxlink** shell script, located at: /usr/opt/db2\_v8\_01/bin on AIX, to perform the following steps:
  - Write information, warning, and error messages to /usr/opt/db2\_08\_01/lib/djxlink.out
  - Create wrappers in the directory: /usr/opt/db2\_08\_01/lib

Also, in that directory are **djxlink** scripts that create just the wrapper libraries for specify data sources.

For instance, **djxlinkOracle** creates just the wrapper libraries for use with Oracle. **djxlinkMssql** creates just the wrapper libraries for use with DataDirect Connect to access SQL Server:

#### – **djxlinkOracle**

To perform **djxlinkOracle**, execute the following steps:

- Log in as root.
- cd /usr/opt/db2\_08\_01/bin
- echo \$ORACLE\_HOME. If the value is correct proceed to next step

If the value is not set or is not correct:

- Export ORACLE\_HOME=<path of the Oracle Client base directory>
- ./djxlinkOracle

After running **djxlinkOracle**, and cd /usr/opt/db2\_08\_01/lib, check for:

- libdb2net8F.a - The fenced wrapper library for use with Oracle
- djxlinkOracle.out - Warning error messages from **djxlinkOracle**

If libdb2net8F.a was successfully created, use **ls -l** to check its permissions. It will have to be readable and executable by the DB2 instance owner. In our case, that is **db2fed32**.

#### – **djxlinkMssql**

**djxlinkMssql** must be run to create it. For the SQL Server wrapper that uses Data Direct Connect to access SQL Server, the install of the DB2 Information Integrator relational wrapper for Microsoft SQL Server on AIX provides:

- libdb2mssql3.a - Base wrapper library
- libdb2mssql3U.a - Unfenced wrapper library
- libdb2STMssql3F.a - Input library for linking with DataDirect Connect ODBC driver manager library to create the fenced wrapper library

- libdb2mssql3F.a - The fenced wrapper. This library interfaces with the DataDirect Connect ODBC driver manager created by the Information Integrator installation only if the link with the DataDirect Connect was successful during the install. If it was not successful, execute the **djxlinkMssql** script to link Information Integrator with DataDirect Connect, and create the fenced wrapper library.
- The wrapper libraries and input libraries are installed into /usr/opt/db2\_v8\_01/lib. The **djxlinkMssql** script is installed into /usr/opt/db2\_v8\_08/bin. To perform **djxlinkMssql**, execute the following steps:
  - i. Log in as root.
  - ii. cd /usr/opt/db2\_08\_01/bin
  - iii. Echo \$DJX\_LOAD\_LIBRARY\_PATH.  
If the value is correct proceed to the next step, if the value is not set or it is not correct:
  - iv. Export DJX\_LOAD\_LIBRARY\_PATH=<path to directory containing DataDirect ODBC driver manager libodbc.a>
  - v. ./djxlinkMssql  
After running **djxlinkMssql** and cd /usr/opt/db2\_08\_01/lib, check for:
  - vi. libdb2mssql3F.a - The fenced wrapper library for use with DataDirect
  - vii. djxlinkMssql.out - Warning error messages from **djxlinkMssql**  
If libdb2mssql3F.a was successfully created, use **ls -l** to check its permissions. It will have to be readable and executable by the DB2 instance owner. In our case, that is **db2fed32**.
- After the execution of **djxlink**, make sure that permissions of the wrapper libraries created are set to read and execute for the DB2 instance owner:
 

ID. ./iisetaup.bin

## 5.3 Applying installed wrappers to instances

After the installation of the wrappers for the different data sources is completed, we need to configure these wrappers by running the **db2isetup** utility out of the /usr/opt/db2\_08\_01/instance directory. If you do not have a DB2 instance available, you can create one during this installation. In this section, we describe how to set up the configuration of our relational wrappers for an existing DB2 instance environment.



For the installed relational wrappers, we need to configure the client parameter for the respective data source. For Oracle you need to specify the home directory of your local Oracle Client installation. For Microsoft SQL Server, you need to specify the home of the ODBC driver installation directory, and the path to the ODBC configuration file (.odbc.ini).

With the DB2 Instance Setup wizard windows, we configure the new functions on an existing DB2 Instance:

- ▶ Open AIX terminal window with root authority.
- ▶ `cd /usr/opt/db2_08_01/instance`
- ▶ Set DISPLAY environment variable
- ▶ `./db2isetup`

As you launch this command, you enter an Introduction screen, with some general comments about the Setup wizard.

Click **Next** to continue.

As shown in Figure 5-33, you need to decide whether you want to create a new DB2 instance, or configure an existing one. We decided to configure our existing DB2 instance **db2fed32**:

1. Click **Configure new function for an existing DB2 instance**.
2. Select **db2fed32** for the instance name.

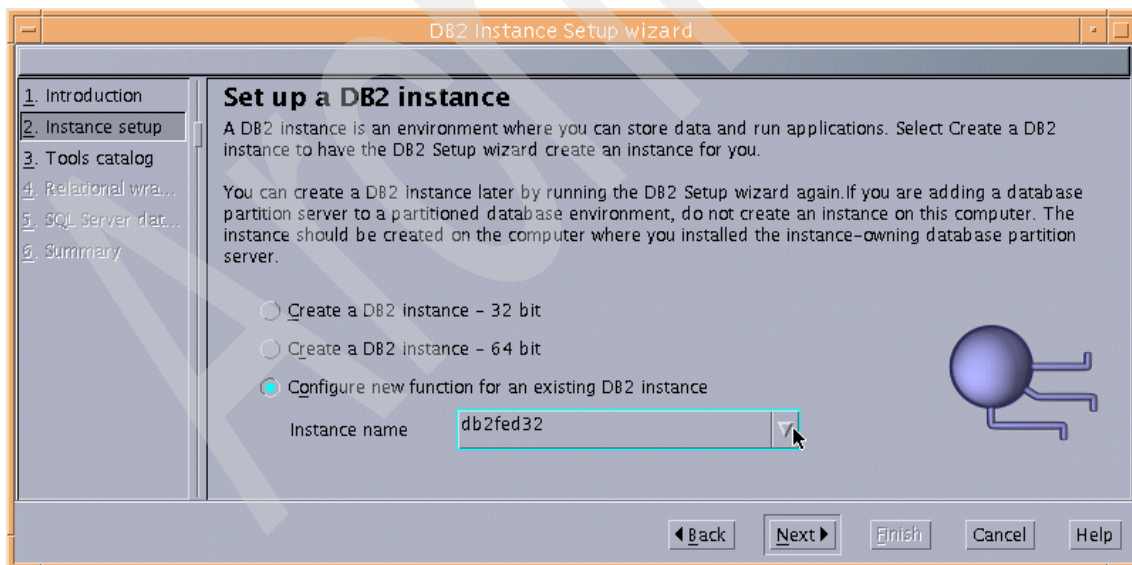


Figure 5-33 Configure the wrappers for the DB2 instance db2fed32

3. Click **Next** to continue.

This brings us to the screen **Prepare the DB2 tools catalog**, which we do not show here. Select **Do not prepare the DB2 tools catalog on this computer**.

4. Click **Next** to continue.

With the screen Relational wrappers set up, you decide whether you want to set up the selected instance to use the relational wrapper components or not. We decided to set up the instance for relational wrappers.

If you decide not to set up the instance to use the relational wrapper components, you can access to IBM data sources only - this is DB2 and Informix.

5. Click **Next** to continue.

See Figure 5-34 for details on how to configure the Oracle data source. Enter the directory of the Oracle Client installation directory.

The installation routine writes this value as parameter ORACLE\_HOME into the file <instance-home>/sqllib/cfg/db2dj.ini

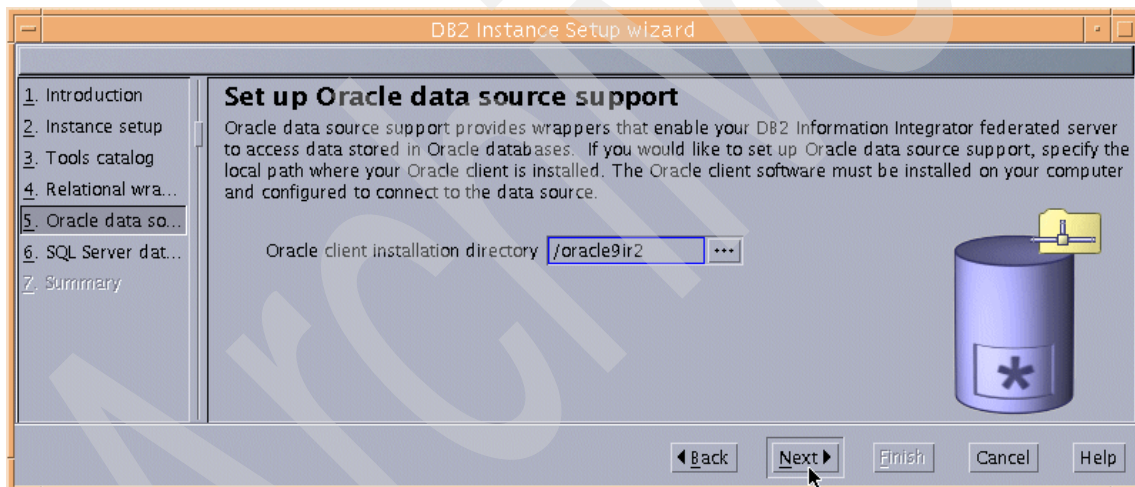


Figure 5-34 Oracle data source support

6. Click **Next** to continue.

You may get a Warning message like displayed in Figure 5-35, even when you entered the correct path to your Oracle installation home directory. This message is nothing to worry about, just double check with your Oracle Client installation to enter the correct value.

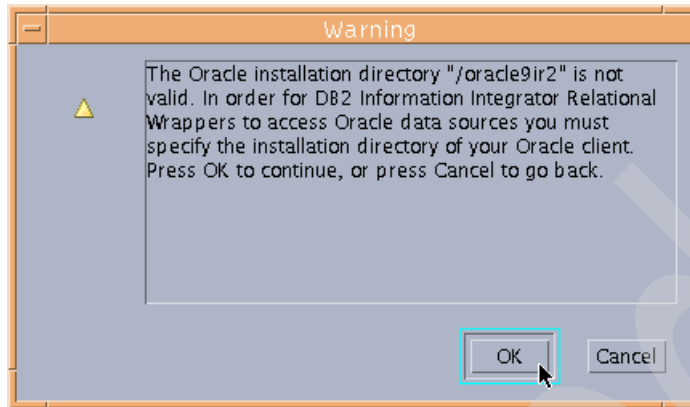


Figure 5-35 Oracle - Warning

7. Click **OK** to accept this message.

For details on configuring the Microsoft SQL Server wrapper, see Figure 5-36. To access Microsoft SQL Server, the SQL Server wrapper uses an ODBC driver. For our UNIX based DB2 Information Integrator installation, we used the DataDirect ODBC driver:

1. Enter the ODBC driver installation directory.
2. Enter the file path to .odbc.ini.
3. Optional: You can enter an ODBC trace directory.

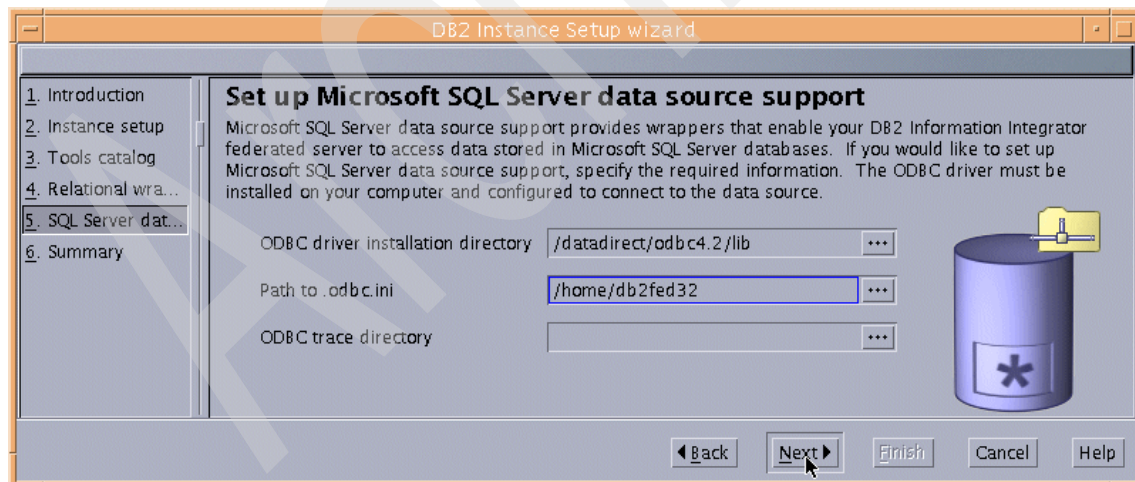


Figure 5-36 Microsoft SQL Server - Data source

4. Click **Next** to proceed.

The Summary shown in Figure 5-37 gives an overview of all your settings for configuring the relation wrappers for the selected instance. Check all your settings and step back if you need to correct any setting.

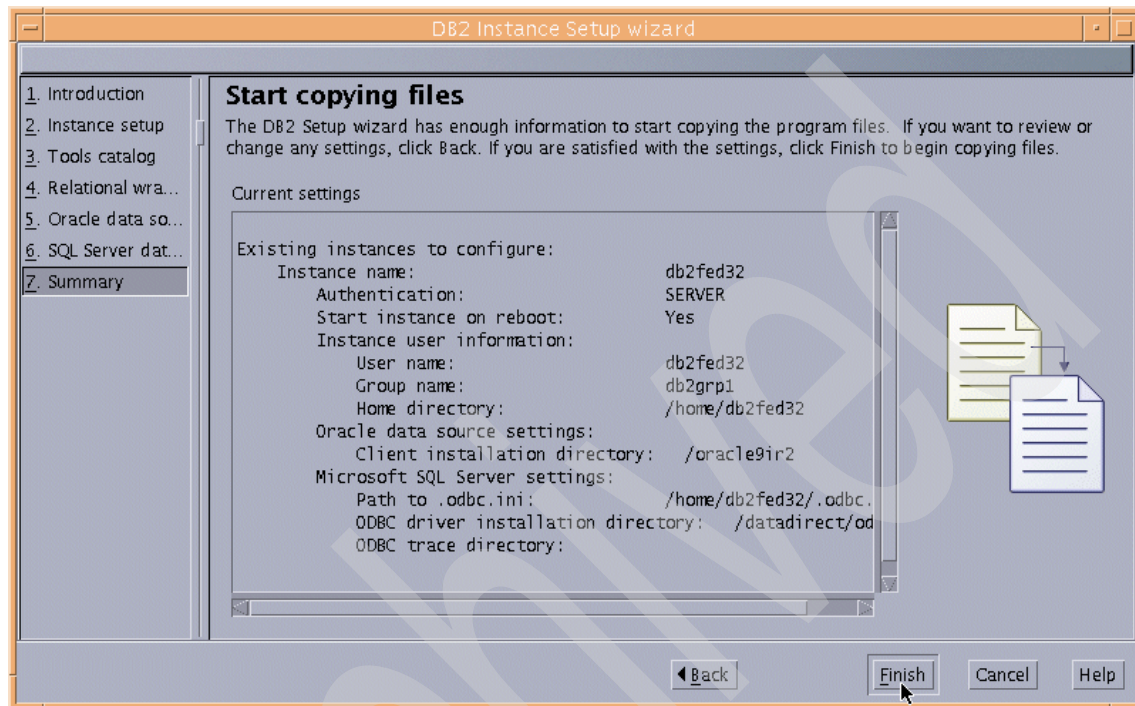


Figure 5-37 DB2 - Setup wizard

5. Click **Finish** to apply the settings to the instance.

Review Figure 5-38 to check the installation status report. For our installation, every step was successful except the last two: the configuration of the Oracle and the Microsoft SQL Server wrappers. View the shaded box below for how to resolve this situation.

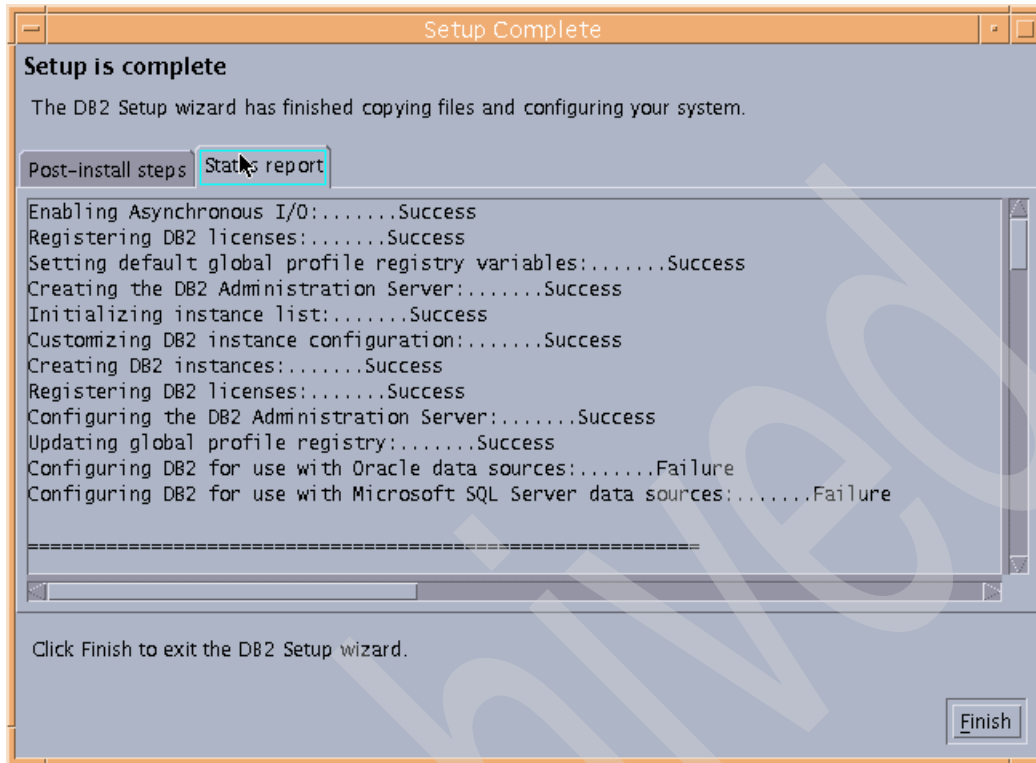


Figure 5-38 DB2 - Setup complete

6. Click **Finish** to end the installation.

**Important:** You may get two error messages during the setup. One in the section of Configuring DB2 for use with Oracle data sources, and the other on Configuring DB2 for use with Microsoft SQL Server data sources.

If you see these two failure messages, you need to perform the script **djxlink** for the data sources, which cannot be configured successfully.

**djxlink** performs a linkage editor command (**ld**) to bind libraries from a specific Oracle version or libraries from a specific ODBC DataDirect version with DB2 library files.

We describe below how to perform **djxlinkOracle** and **djxlinkMssql**.



### ***djxlinkOracle and djxlinkMssql linkage***

To perform **djxlinkOracle** and **djxlinkMssql** linkage, execute following steps:

1. Open AIX terminal window with root authority.
2. `cd /usr/opt/db2_08_01/bin`
3. For Oracle: `./djxlinkOracle` - This command might take a few seconds. You should get a succeed message.
4. For Microsoft SQL Server: `./djxlinkMssql` - This might take a few seconds. You should get a succeed message.
5. In the directory `/usr/opt/db2_08_1/lib`, check if you find new DB2 libraries and two output files:
  - `ls -l libdb2*net*.a`
  - `ls -l libdb2*mssql*.a`
  - `more djxlinkOracle.out`
  - `more djxlinkMssql.out`

## **5.4 Integrating Oracle 9i**

This chapter describes all steps necessary to integrate a data source based on Oracle 9i into a DB2 Federated Server, and to establishing a link between the table located on Oracle 9i and your federated system:

- ▶ Configuration information for Oracle 9i wrapper
- ▶ Creating the Oracle wrapper
- ▶ Creating the Oracle server
- ▶ Altering Oracle server definition and server options (optional)
- ▶ Creating Oracle user mappings
- ▶ Altering Oracle user mappings (optional)
- ▶ Creating Oracle nicknames

We use the Control Center on Windows platform to perform the necessary administration steps.

### **5.4.1 Configuration information for Oracle 9i wrapper**

The information in Table 5-2 is necessary to integrate the Oracle 9i data source:

*Table 5-2 Oracle information*

Parameter	Value
ORACLE_HOME	/home/oracle9ir2

Parameter	Value
ORACLE_SID	demo_on
Port	1521
User/Password	oracle/oracle
Schema	tpch

**Note:** Make sure you installed the Oracle Client on the federated server, and that you successfully configured and tested the connection to your Oracle server.

## Oracle wrapper libraries

For the Oracle wrapper, the install of the DB2 Information Integrator relational wrapper for Oracle on AIX provides:

- ▶ libdb2net8.a - Base wrapper library
- ▶ libdb2net8U.a - Unfenced wrapper library
- ▶ libdb2STnet8F.a - Input library for linking with Oracle Client to create the fenced wrapper library.
- ▶ libdb2net8F.a - The fenced wrapper. This library interfaces with the Oracle Client, created by the Information Integrator installation only if the link with the Oracle Client was successful during the install. If not, **djxlink0racle** must be run to create it.
- ▶ **djxlink0racle** - The script to link Information Integrator with the Oracle Client and created the fenced wrapper library.

The wrapper libraries and input libraries are installed into /usr/opt/db2\_v8\_01/lib.

The **djxlink0racle** script is installed into /usr/opt/db2\_v8\_08/bin

On Windows, the wrapper libraries use dynamic linking to find the Oracle Client. Linking is not required. The Information Integrator install provides the base, unfenced, and fenced libraries. They are installed into c:\Program Files\IBM\SQLLIB\bin. There is no input library for linking and no **djxlink0racle**.

## The Oracle wrapper and DB2 FixPaks

Updates to the Oracle wrapper are included in DB2 ESE 8.1 FixPaks.

On Windows, the FixPak updates the DB2 engine and all the files of the Oracle wrapper at the same time.

On UNIX, however, the FixPak updates only the base and 'U' Oracle wrapper libraries to the new FixPak level leaving the 'F' library at the old maintenance level. This is because the 'F' library is the one that is linked with the Oracle Client. The FixPak installation process unfortunately does not do this link. If there are updates to the Oracle wrapper in the FixPak, it will provide a new input library (libdb2STnet8F.a) for the link with the Oracle Client and possibly an updated **djx1ink0racle**. After the FixPak is applied, **djx1ink0racle** must be run by root to create a libdb2net8F.a that is the same level as the DB2 engine, and the base and 'F' wrapper libraries.

**Note:** If a FixPak is applied and **djx1ink0racle** is not run, what happens when nicknames are created or used is unpredictable.

## Oracle tnsnames.ora

The Oracle tnsnames.ora contains information the Oracle Client uses to connect to the Oracle server. The file is usually located in the /network/admin sub-directory of the Oracle Client.

There needs to be an entry for the Oracle server that we will be defining federated access to.

The name at the beginning of the entry is called the Oracle Network Service Name value, and is the value that will be used as the NODE setting of our Information Integrator server definition to the Oracle server.

Example 5-3 shows our tnsnames.ora file. The Network Service Name that we will use as the NODE setting in our Information Integrator server definition is highlighted.

*Example 5-3 The tnsnames.ora file*

---

```
ORAFED =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = lead)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = demo_on)
    )
  )
```

---



### db2dj.ini

As indicated in Table 4-1 on page 89, the DB2 instance owner's /sqlib/cfg/db2dj.ini file must contain the variable ORACLE\_HOME. Here we discuss the Oracle variables in db2dj.ini:

- ▶ ORACLE\_HOME - required. It indicates the Oracle Client's base directory.
- ▶ TNS\_ADMIN - optional. It indicates directory containing tnsnames.ora file. Only required if the tnsname.ora file is not in the default location, which is the Oracle Client's /network/admin sub-directory.
- ▶ ORACLE\_NLS - optional. See discussion in Chapter 8, "National language support" on page 293.
- ▶ ORACLE\_BASE - optional

## 5.4.2 Creating the Oracle wrapper

Here are the steps to create the Oracle wrapper:

1. Open the DB2 Control Center.
2. Expand your instance and your database to **FEDDB**.
3. Right-click **Federated Database Objects** for the database **feddb** and click **Create Wrapper**.
4. Choose the data source type and enter a unique wrapper name like shown in Figure 5-39. If your Oracle data source has Version 8i or higher, select **Oracle using OCI 8**, if not choose **Oracle using OCI 7**.
5. Click **OK**.

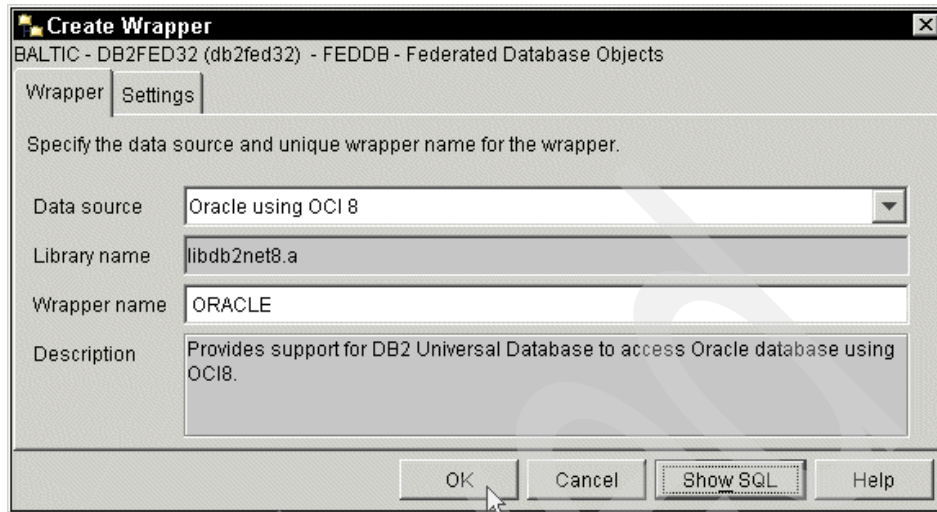


Figure 5-39 Oracle - Create Wrapper

Example 5-4 shows the command line version of creating the wrapper for your Oracle instance. Additionally, you may check your wrapper definition in the DB2 system catalogue with the two select statements included in the example.

Example 5-4 Oracle - Create wrapper statement

---

```
CONNECT TO FEDDB;
CREATE WRAPPER "ORACLE" LIBRARY 'libdb2net8.a';

SELECT * from SYSCAT.WRAPPERS;
SELECT * from SYSCAT.WRAPOPTIONS;
```

---

### 5.4.3 Creating the Oracle server

A server definition identifies a data source to the federated database. A server definition consists of a local name and other information about that data source server. Since we have just created the Oracle wrapper, we need to specify the Oracle server, from which you want to access data. For your wrapper defined, you can create several server definitions.

Steps to create an Oracle server:

1. Select the wrapper you created in the previous step - **ORACLE**.
2. Right-click **Servers** for wrapper **ORACLE** and click **Create**.

The server definition demands the following inputs, visualized in Figure 5-40:

- **Name:** The name of the server must be unique over all server definitions available on this federated database.
- **Type:** Select **ORACLE**
- **Version:** Select **8** or **9**. If you use Oracle wrapper using OCI 7, select the right version of your Oracle data source server.

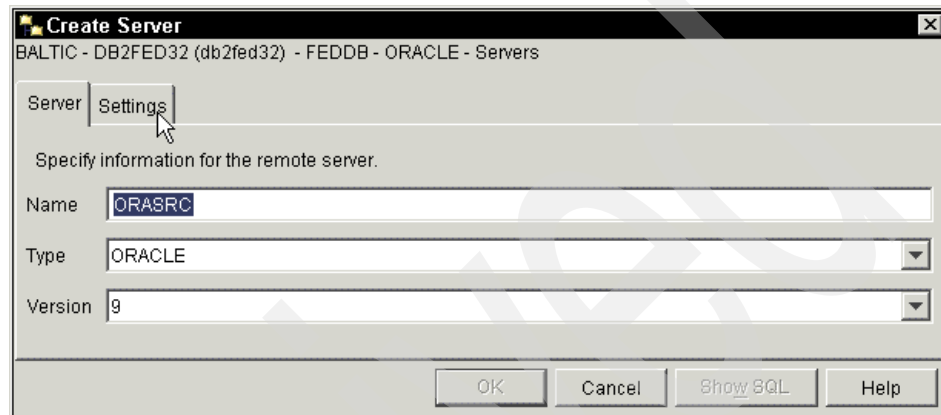


Figure 5-40 Oracle - Create Server dialog

Switch to the **Settings** menu to complete your server edition. For server settings visualized in Figure 5-41, some input fields required definition, some are optional. Required are the first two fields: Node and Password

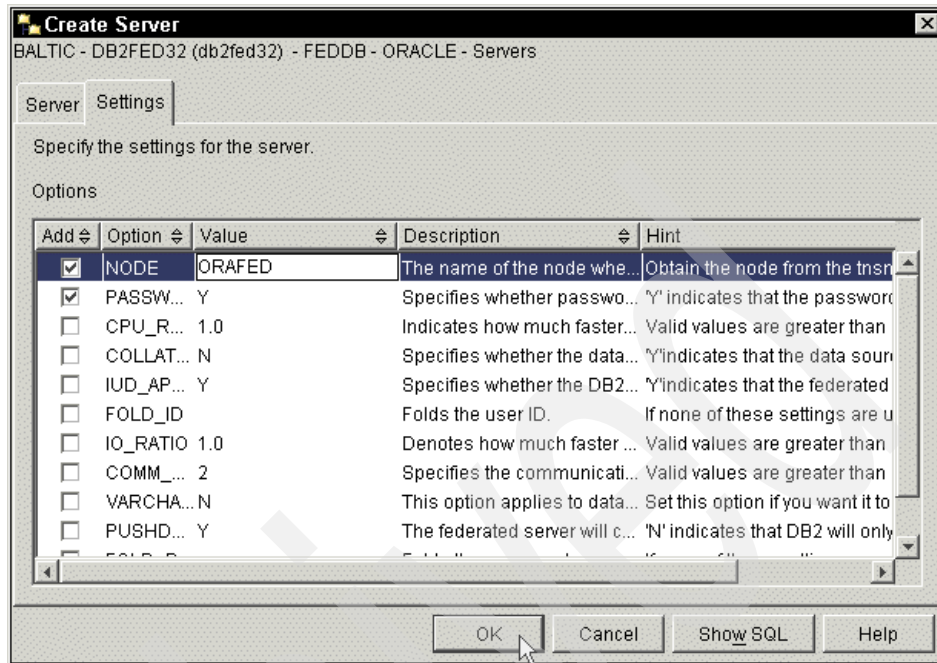


Figure 5-41 Oracle - Create Server - Settings

Server options are used to describe a data source server. Oracle servers have the following set of options as listed in Table 5-3. You can set the options at creation time of your server, or modify these settings afterwards. Please read the chapter 5.4.4 “Altering Oracle server definition and server options” on page 176 for more details.

Table 5-3 Oracle server options

Connectivity	Node	It identifies the entry name (Network Service Name) in Oracle <b>tnsnames.ora</b> file. Case sensitive.
	Password	<b>Default=Y:</b> Include the password on connections to Oracle
	Fold_ID / Fold_PW	<b>Default</b> is wrapper dependent. <b>Y</b> - connect four times with user ID/password with all combinations of lower and uppercase. <b>N</b> (recommended) - connect once with user ID/password exactly as specified <b>U</b> - connect once with user ID/password in uppercase <b>L</b> - connect once with user ID/password in lower case

<b>Pushdown</b>	Collating_Sequence	It specifies whether the data source uses the same default collating sequence as the federated database. This affects the pushdown of operations on character columns. <b>Default=N</b> <b>Y:</b> both use same collating sequence. ORDER BY can be pushed down without compromising integrity of result <b>N:</b> both use different collation sequence. ORDER BY cannot be pushed down <b>I:</b> case-insensitive. ORDER BY, DISTINCT, WHERE= cannot be pushed down
	Pushdown	<b>Default: Y:</b> the SQL operations are pushed down to data sources based on decision of pushdown analyzer and optimizer.
<b>Optimization</b>	CPU_RATIO	<b>Default: 1.0:</b> specifies the ratio of the DB2 Information Integrator server CPU capacity against the data source CPU capacity
	IO_RATIO	<b>Default: 1.0:</b> specifies the ratio of the DB2 Information Integrator server IO rate against the data source IO rate
	COMM_RATE	<b>Default: 2:</b> specifies the effective data rate of network to data source in MB per second.
<b>Other</b>	IUD_APP_SVPT_ENFORCE	<b>Default=Y:</b> should the DB2 federated system use save-points in multi-update transactions?
	VARCHAR_NO_TRAILING_BLANKS	<b>Default=N:</b> This option applies to variable character data types that do not pad the length with trailing blanks. Set this option to <b>Y</b> , when none of the columns contains trailing blanks. If only some of the VARCHAR columns contain trailing blanks, you can set an option with the ALTER NICKNAME statement.

The following Example 5-5 shows the command line version of creating the server for your Oracle instance. Additionally, you may check your server definition in the DB2 federated system catalog with the two select statements listed below.

*Example 5-5 Oracle - Create server statement*

---

```
CONNECT TO FEDDB;
CREATE SERVER ORASRC TYPE ORACLE VERSION '9' WRAPPER "ORACLE" OPTIONS( ADD NODE
'ORAFED', PASSWORD 'Y');
```

```
SELECT * from SYSCAT.SERVERS;
SELECT * from SYSCAT.SERVEROPTIONS;
```

---

## 5.4.4 Altering Oracle server definition and server options

You may modify a server definition when you:

- Upgrade a data source to a new version:  
`alter server ORASRC version 9`
- Want to change server options to different values

Server options are set to values that persist over successive connections to the data source. The values are stored in the federated system catalog. Here is an example to activate (add), set and deactivate (drop) a server option:

```
alter server ORASRC options (add IO_RATIO '2.0')
alter server ORASRC option (set IO_RATIO '3.0')
alter server ORASRC option (drop IO_RATIO)
```

To set a server option value temporarily, use the `SET SERVER OPTION` statement. This statement overrides the server option value in the server definition for the duration of a single connection to the federated database. The settings are not stored in the federated system catalog. An example is:

```
set server option COMM_RATE TO '3' for server ORASRC
```

**Tip:** The server options shown in Table 5-4 are not available through the Control Center. You must set these options through command line.

Table 5-4 Oracle additional server options

DB2_MAXIMAL_PUSHDOWN	It specifies the primary criteria for the optimizer in choosing the access plan. The optimizer can choose between cost optimization and the user requirement to perform as much as possible query processing by the remote data source. 'Y': choose the plan with most query operations to be pushed down to the data sources. 'N': choose the plan with minimum cost.
PLAN_HINTS	Hints are statement fragments that provide extra information for the Oracle optimizer. 'Y': Enabled 'N': Disabled

## 5.4.5 Creating Oracle user mappings

When the federated server needs to access the data source server, it needs to establish the connection with it first. With the user mapping, you define an association from a federated server user ID to an Oracle user ID and password that Information Integrator will use in connections to the Oracle server on behalf

of the federated user. An association must be created for each user who will be using the federated system. In our case, we only use the user ID db2fed32.

Here are the steps to create an Oracle server:

1. Select the server we created in the previous step - **ORASRC**.
2. Right-click **User Mappings** for server **ORASRC** and click **Create**.

Figure 5-42 lists all the users IDs available on your federated system. Select the user who will be the sender of your distributed requests to the Oracle data source. We selected the owner of our federated server instance, **db2fed32**.

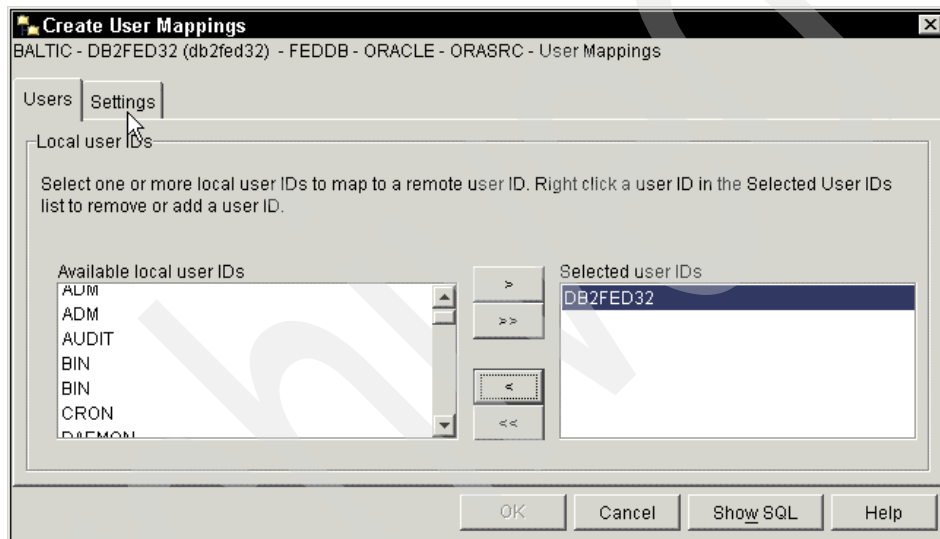


Figure 5-42 Oracle - Create User Mappings

Switch to settings as shown in Figure 5-43, to complete the user mapping. You need to identify the username and password to enable the federated system to connect to our Oracle data source.

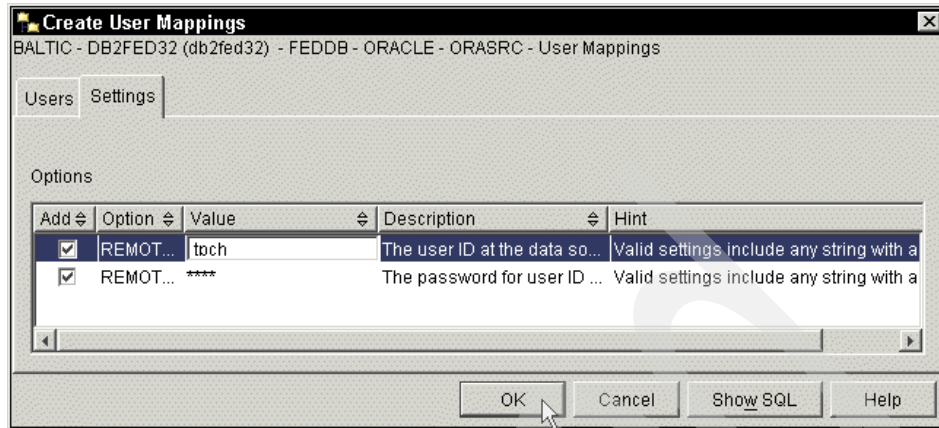


Figure 5-43 Oracle - Create User Mappings - Settings

Example 5-6 shows the command line version of creating the user mapping for your Oracle instance. Additionally, you may check your user mapping definition in the DB2 federated system catalog with the select statement listed at the end.

Example 5-6 Oracle - Create user mapping statement

```
CONNECT TO FEDDB;
CREATE USER MAPPING FOR "DB2FED32" SERVER "ORACLE9" OPTIONS ( ADD REMOTE_AUTHID
'tpch', ADD REMOTE_PASSWORD '*****' ) ;

SELECT * from SYSCAT.USEROPTIONS;
```

## 5.4.6 Altering Oracle user mappings

You can use the ALTER USER MAPPING statement to modify a user mapping. The statement is used to change the authorization ID or password that is used at the Oracle data source:

```
alter user mapping for db2fed32 SERVER ORASRC options
(set remote_authid 'paolo')
alter user mapping for db2fed32 server ORASRC options
(set remote_password 'newpass')
```

**Note:** The REMOTE\_AUTHID and REMOTE\_PASSWORD user options are case sensitive unless you set the FOLD\_ID and FOLD\_PW server options to 'U' or 'L' in the CREATE SERVER statement.

User mappings can also be altered in the DB2 Control Center.



## 5.4.7 Creating Oracle nicknames

After having set up the Oracle wrapper, the server definition and the user mapping to our Oracle database, we finally need to create the actual link to a table located on our remote database as a nickname.

When you create a nickname for an Oracle table, catalog data from the remote server is retrieved and stored in the federated global catalog.

Steps to create a DB2 Oracle nickname:

1. Select the server **ORASRC**.
2. Right-click **Nicknames** for the server **ORASRC** and click **Create**.

You see a dialog displayed in Figure 5-44 showing up. You have two possibilities to add a nickname. Either you click **Add** to manually add a nickname by specifying local and remote schema and table identification, or you use the **Discover** functionality, which we describe here.

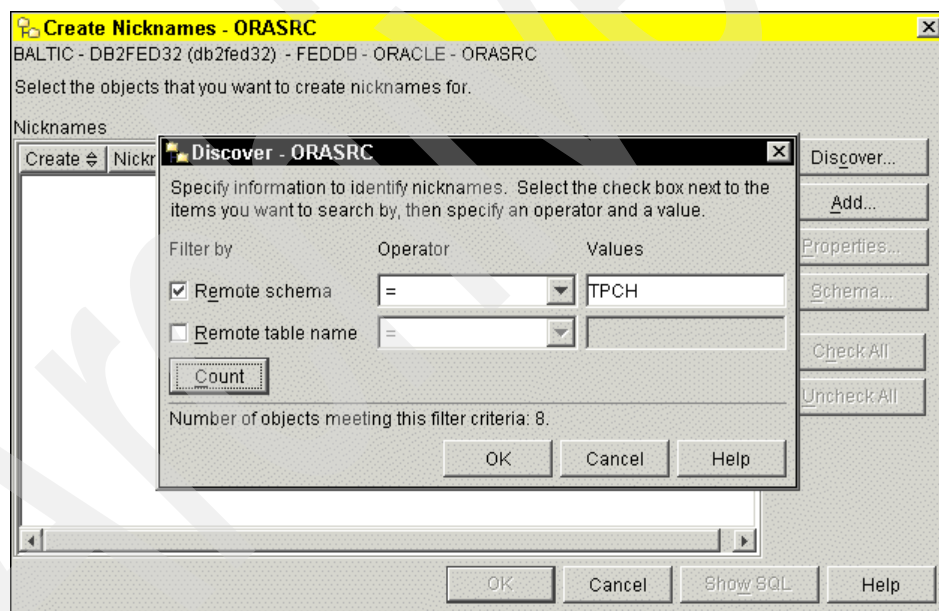


Figure 5-44 Oracle - Create Nicknames

*Discover* your remote Oracle data source by specifying either a **Remote Schema** or a specific **Remote Table Name** that you want to create a nickname for.

Choose a **filter** method, an **operator**, and the comparing **value**, and click **Count**. Upon successful access to our Oracle database, the number of objects which

meet the filter criteria will be displayed. Adjust your filter criteria if you find the number is too few or many. Click **OK** to list the actual objects as a list in the dialog, as displayed in Figure 5-45.

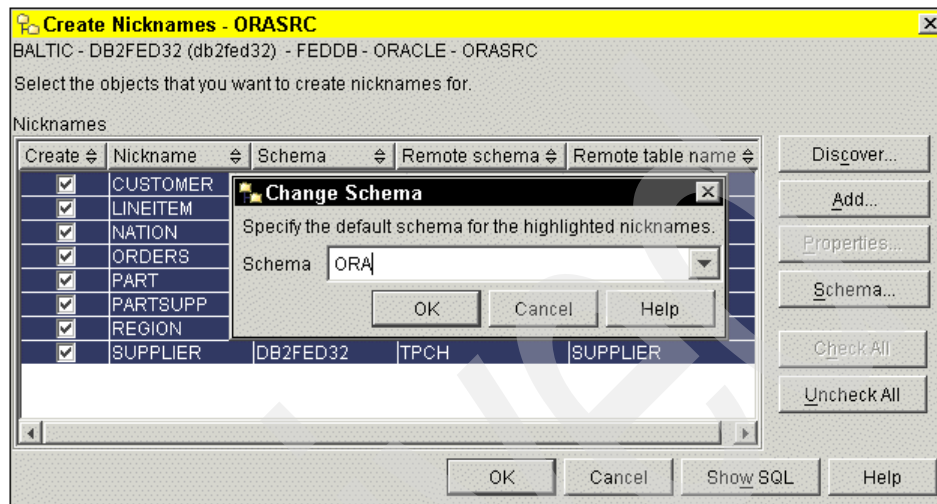


Figure 5-45 Oracle - Create Nicknames - Change Schema

If you use the Discover filter, the default schema for the entries added to the Create Nickname window is the user ID that is creating the nicknames. In our case, that is DB2FED32. We want to change this schema to adhere to our naming convention.

Use the **schema** button to change the local schema for your Oracle nicknames.

**Tip:** We recommend that you use the same schema name for all Oracle nicknames.

Example 5-7 shows the command line version of creating the nicknames for your Oracle instance. Additionally, you may check the nickname definition in the DB2 system catalog with the select statements listed in the example.

#### Example 5-7 Oracle - Create nickname statements

```
CONNECT TO FEDDB;
CREATE NICKNAME ORA.CUSTOMER FOR ORACLE9.TPCH.CUSTOMER;
CREATE NICKNAME ORA.LINEITEM FOR ORACLE9.TPCH.LINEITEM;
CREATE NICKNAME ORA.NATION FOR ORACLE9.TPCH.NATION;
CREATE NICKNAME ORA.ORDERS FOR ORACLE9.TPCH.ORDERS;
CREATE NICKNAME ORA.PART FOR ORACLE9.TPCH.PART;
CREATE NICKNAME ORA.PARTSUPP FOR ORACLE9.TPCH.PARTSUPP;
```

```

CREATE NICKNAME ORA.REGION FOR ORACLE9.TPCH.REGION;
CREATE NICKNAME ORA.SUPPLIER FOR ORACLE9.TPCH.SUPPLIER;

SELECT * from SYSCAT.TABLES WHERE TABSCHEMA='ORASRC';
SELECT * from SYSCAT.TABOPTIONS WHERE TABSCHEMA='ORASRC';
SELECT * from SYSCAT.COLUMNS WHERE TABSCHEMA='ORASRC';
SELECT * from SYSCAT.COLOPTIONS WHERE TABSCHEMA='ORASRC';
SELECT * from SYSCAT.INDEXES WHERE TABSCHEMA='ORASRC';
SELECT * from SYSCAT.INDEXOPTIONS WHERE TABSCHEMA='ORASRC';
SELECT * from SYSCAT.KEYCOLUSE WHERE TABSCHEMA='ORASRC';

```

---

### 5.4.8 Altering Oracle nicknames

Use the ALTER NICKNAME statement to modify the federated database representation of a data source. Use this statement to:

- ▶ Change local data type of a column:  

```
alter nickname ORA.REGION alter column r_name local type varchar (50)
```
- ▶ Change local column name:  

```
alter nickname ORA.REGION alter column r_name local name "R_REGIONNAME"
```

## 5.5 Integrating Microsoft SQL Server 2000

This section describes all the steps necessary to integrate a data source based on Microsoft SQL Server 2000 into a DB2 Federated Server, and actually establish a link between the table located on SQL Server and your federated server:

- ▶ Microsoft SQL Server client configuration
- ▶ Creating the Microsoft SQL Server wrapper
- ▶ Creating the Microsoft SQL Server server
- ▶ Altering Microsoft SQL Server definition and server options (optional)
- ▶ Creating Microsoft SQL Server user mappings
- ▶ Altering Microsoft SQL Server user mappings (optional)
- ▶ Creating Microsoft SQL Server nicknames
- ▶ Altering Microsoft SQL Server nicknames (optional)

We use the Control Center on the Windows platform to perform the necessary administration steps.

## 5.5.1 Microsoft SQL Server client configuration

The information in Table 5-5 is necessary to integrate Microsoft SQL Server data source.

Table 5-5 Microsoft SQL Server information

Parameter	Value
Server	radon.almaden.ibm.com®
DBName	SQLServDB
Port	1433
User/Password	sqlserv/sqlserv

**Note:** Make sure you installed the DataDirect ODBC driver on the federated server, and that you successfully configured and tested the connection to your Microsoft SQL Server!

### Microsoft SQL Server wrapper libraries

For Microsoft SQL Server, the install of the DB2 Information Integrator Relational Wrapper for SQL Server on AIX provides:

- ▶ libdb2mssql3.a - Base wrapper library
- ▶ libdb2mssql3U.a - Unfenced wrapper library
- ▶ libdb2STmssql3F.a - Input library for linking with DataDirect Connect ODBC to create the fenced wrapper library.
- ▶ libdb2mssql3F.a - The fenced wrapper  
This library interfaces with the DataDirect Connect; created by the Information Integrator installation only if the link with DataDirect Connect was successful during the install. If not, **djxlinkMssql** must be run to create it.
- ▶ **djxlinkMssql** - Script to link Information Integrator with DataDirect Connect and create the fenced wrapper library.

The wrapper libraries and input libraries are installed into /usr/opt/db2\_v8\_01/lib.

The **djxlinkMssql** script is installed into /usr/opt/db2\_v8\_08/bin

On Windows, the wrapper libraries use dynamic linking to find the SQL Server ODBC driver. Linking is not required. The Information Integrator install provides the base, unfenced, and fenced libraries. They are installed into c:\Program Files\IBM\SQLLIB\bin. There is no input library for linking and no **djxlinkMssql**.

## The Microsoft SQL Server wrapper and DB2 FixPaks

Updates to the SQL Server wrapper are included in DB2 ESE 8.1 FixPaks.

On Windows, the FixPak updates the DB2 engine and all the files of the SQL Server wrapper at the same time.

On UNIX, however, the FixPak updates only the base and 'U' SQL Server wrapper libraries to the new FixPak level leaving the 'F' library at the old maintenance level. This is because the 'F' library is the one that is linked with DataDirect Connect. The FixPak installation process unfortunately does not do this link. If there are updates to the SQL Server wrapper in the FixPak, it will provide a new input library (libdb2STMssql3F.a) for the link with DataDirect Connect and possibly an updated **djx1inkMssql**. After the FixPak is applied, **djx1inkMssql** must be run by root to create a libdb2mssql3F.a that is the same level as the DB2 engine and the base and 'F' wrapper libraries.

**Note:** If a FixPak is applied and **djx1inkMssql** is not run, what happens when nicknames are created or used is unpredictable.

### .odbc.ini file

On AIX, the .odbc.ini file contains information the DataDirect Connect ODBC driver for Microsoft SQL Server uses to connect to the SQL Server server. The file can be anywhere on the system. The ODBC\_INI variable in the db2dj.ini file will tell Information Integrator to find the .odbc.ini file it is to use. It is recommended that a copy of the .odbc.ini file containing an entry for the SQL Server server be placed in the DB2 instance owner's home directory.

There needs to be an entry for the SQL Server server that we will be defining federated access to.

The name of the entry is called the Data Source Name value, and is the value that will be used as the **NODE** setting of our Information Integrator server definition to the SQL Server server.

Example 5-8 shows a .odbc.ini entry for SQL Server. The Data Source Name that we will use as the **NODE** setting in our Information Integrator server definition is highlighted.

#### *Example 5-8 Entry .odbc.ini for SQL Server*

---

RADONSQL=MS SQL Server 2000

[RADONSQL]

Driver=/opt/odbc/lib/libivmsss18.so

Description=MS SQL Server on RADON

### Notes:

Other parameters can be included in the .odbc.ini entry for the SQL Server server. The example shows the minimum required for Information Integrator to use the entry to connect to the SQL Server server.

For Information Integrator on Windows, the entry for the SQL Server server needs to be in Windows ODBC Data Source Administration; the entry needs to be a system DSN.

### db2dj.ini

As indicated in Table 4-1, the DB2 instance owner's /sqllib/cfg/db2dj.ini file must contain the variable ODBC\_INI and DJX\_LOAD\_LIBRARY\_PATH. Here is discussion of the variables for SQL Server in db2dj.ini.

Entries are only required on UNIX. No entries are required in db2dj.ini on Windows for Information Integrator to connect a SQL Server server:

- ▶ DJX\_ODBC\_LIBRARY\_PATH - required  
Indicates location of the DataDirect Connect Driver Manager (libodbc.a) and SQL Server ODBC driver. Example:  
DJX\_LOAD\_LIBRARY\_PATH=/opt/odbc/lib
- ▶ ODBC\_INI - required.  
Indicates the full path to the .odbc.ini file that Information Integrator is to use.  
Example ODBC\_INI=/home/db2fed32/.odbc.ini

**Note:** If the .odbc.ini file is in the DB2 instance owner's home directory (like in the example), you may be tempted to use \$HOME in the specification. Do not do this. This will cause errors, such as DB2 crashes.

### db2set

On UNIX and Linux, two db2set variables must be set when using the Information Integrator Relational Wrapper for SQL Server. These variables are not required on Windows:

- ▶ db2set DB2ENVLIST=LIBPATH
- ▶ db2set DB2LIBPATH=<path of the directory containing the DataDirect Connect driver manager (libodbc.a) and SQL Server ODBC driver>

```
db2set DB2ENVLIST=LIBPATH
db2set DB2LIBPATH=/opt/odbc/lib
```

## 5.5.2 Creating the Microsoft SQL Server wrapper

Here are the steps:

1. Expand your instance and database.
2. Right-click **Federated Database Objects** for database **feddb** and click **Create Wrapper**.
3. Choose data source type **MicrosoftSQL** and enter a unique wrapper name.
4. Click **OK**.

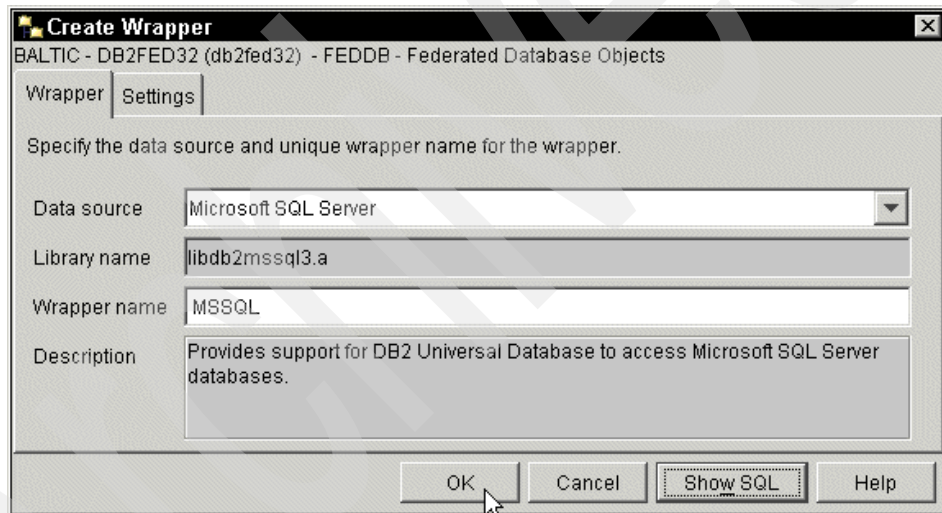


Figure 5-46 Microsoft SQL Server - Create Wrapper dialog

The following Example 5-10 shows the command line version of creating the wrapper for your Microsoft SQL Server instance. Please check the wrapper definition with the two DB2 system catalog tables listed.

### Example 5-10 SQL Server - Create wrapper statements

```
CONNECT TO FEDDB;
CREATE WRAPPER "MSSQL" LIBRARY 'libdb2mssql3.a';

SELECT * from SYSCAT.WRAPPERS;
```

```
SELECT * from SYSCAT.WRAPOPTIONS;
```

---

### 5.5.3 Creating the Microsoft SQL Server server

A server definition identifies a data source to the federated database. A server definition consists of a local name and other information about that data source server. Since we have just created the Microsoft SQL wrapper, we need to specify the SQL Server, from which you want to access data. For your wrapper defined, you can create several server definitions.

Steps to create an Microsoft SQL Server:

1. Select the wrapper you created in the previous step - **MSSQL**.
2. Right-click **Servers** for wrapper **MSSQL** and click **Create**.

The server definition demands following inputs, visualized in Figure 5-47:

- ▶ **Name:** The name of the server must be unique over all server definitions available on this federated database.
- ▶ **Type:** Select MSSQLSERVER
- ▶ **Version:** Select 6.5, 7.0 or 2000



Figure 5-47 Microsoft SQL Server - Create Server

Switch to the Settings menu to complete your server definitions. See the description below and displayed in Figure 5-48.



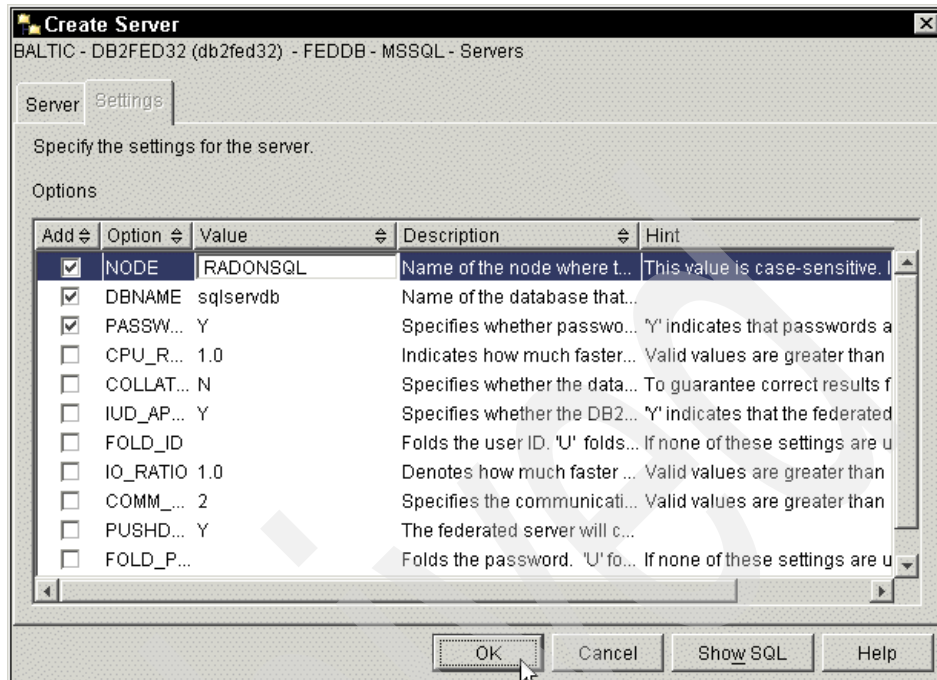


Figure 5-48 Microsoft SQL Server - Create Server - Settings

For a Microsoft SQL Server you need to specify the first three options, Node, DBName and Password, in order to entirely define the connection. All other server options are optional.

Server options are used to describe a data source server. Microsoft SQL Server has the following set of options as listed in Table 5-6. You can set these options at server creation time, or modify these settings afterwards. Please read the Chapter Altering Microsoft SQL Server definition and server options for more details.

Table 5-6 Microsoft SQL Server - Server options

<b>Connectivity</b>	Node	For Information Integrator on Windows, indicates a System DSN name in Windows ODBC Data Source administration. This parameter is case sensitive. For Information Integrator on UNIX, indicates a Data Source Name in the .odbc.ini file. This parameter is case sensitive.
	DBName	Database name at the SQL Server server. Case sensitive.
	Password	<b>Default=Y:</b> Include the password on connections to SQL Server
	Fold_ID / Fold_PW	<b>Default</b> is wrapper dependent. <b>Y</b> - connect four times with user ID/password with all combinations of lower and uppercase. <b>N</b> (recommended) - connect once with user ID/password exactly as specified <b>U</b> - connect once with user ID/password in uppercase <b>L</b> - connect once with user ID/password in lower case
<b>Pushdown</b>	Collating_Sequence	It specifies whether the data source uses the same default collating sequence as the federated database. This affects the pushdown of operations on character columns. <b>Default=N</b> <b>Y:</b> both use same collating sequence. ORDER BY can be pushed down without compromising integrity of result <b>N:</b> both use different collation sequence. ORDER BY cannot be pushed down <b>I:</b> case-insensitive. ORDER BY, DISTINCT, WHERE= cannot be pushed down
	Pushdown	<b>Default: Y:</b> the SQL operations are pushed down to data sources based on decision of pushdown analyses and optimizer.
<b>Optimization</b>	CPU_RATIO	<b>Default: 1.0:</b> specifies the ratio of the DB2 Information Integrator server CPU capacity against the data source CPU capacity
	IO_RATIO	<b>Default: 1.0:</b> specifies the ratio of the DB2 Information Integrator server IO rate against the data source IO rate
	COMM_RATE	<b>Default: 2:</b> specifies the effective data rate of network to data source in MB per second.
<b>Other</b>	IUD_APP_SVPT_ENFORCE	<b>Default=Y:</b> should the DB2 federated system use save-points in multi-update transactions? Y indicates that the federated server will not allow INSERT, UPDATE and DELETES on nicknames if SQL Server does not support save points. --FIXME-- does SQL Server supports save points?

Example 5-11 shows the command line version of creating the wrapper for your Microsoft SQL instance. Additionally, you may check your server definition in the DB2 system catalogue with the two select statements listed below.

*Example 5-11 Microsoft SQL Server - Create server statement*

---

```
CONNECT TO FEDDB;  
CREATE SERVER MSSQL2000 TYPE MSSQLSERVER VERSION '2000' WRAPPER "MSSQL"  
OPTIONS( ADD NODE 'RADONSQL', DBNAME 'sqlservdb', PASSWORD 'Y');  
  
SELECT * from SYSCAT.SERVERS;  
SELECT * from SYSCAT.SERVEROPTIONS;
```

---

## 5.5.4 Altering Microsoft SQL Server definition and server options

You may modify a server definition when you:

- ▶ Upgrade a data source to a new version:  

```
alter server MSSQL2000 version 6.5  
alter server MSSQL2000 version 2000
```
- ▶ Want to change server options to different values

Server options are set to values that persist over successive connections to the data source. The values are stored in the federated system catalog. Here is an example to activate (add), set and deactivate (drop) a server option:

```
alter server MSSQL2000 options (add IO_RATIO '2.0')  
alter server MSSQL2000 options (set IO_RATIO '3.0')  
alter server MSSQL2000 options (set IO_RATIO)
```

To set a server option value temporarily, use the SET SERVER OPTION statement. This statement overrides the server option value in the server definition for the duration of a single connection to the federated database. The settings are not stored in the federated system catalog. An example is:

```
set server option COMM_RATE TO '3' for server MSSQL2000
```

The server option DB2\_MAXIMAL\_PUSHDOWN, outlined in Table 5-7, specifies the primary criteria for the optimizer in choosing the access plan.

**Tip:** The server option shown in Table 5-7 is not available through the Control Center, but setting this option to Y might help you in analyzing your federated queries.

Table 5-7 Microsoft SQL Server additional server options

<b>DB2_MAXIMAL_PUSHDOWN</b>	Default: 'N' The optimizer can choose between cost optimization and the user requirement to perform as much as possible query processing by the remote data source. 'Y': choose the plan with most query operations to be pushed down to the data sources. 'N': choose the plan with minimum cost.
-----------------------------	---

### 5.5.5 Creating Microsoft SQL Server user mappings

When the federated server needs to access the data source server, it needs to establish the connection with it first. With the user mapping you define an association from a federated server user ID and password to an SQL Server user ID and password. An association must be created for each user that will be using the federated system. In our case, we only use one user ID, db2fed32.

Steps to create a user mapping definition:

1. Select the server we created in the previous step - **MSSQL2000**.
2. Right-click **User Mappings** for server **MSSQL2000** and click **Create**.

Figure 5-49 lists all users IDs available on your federated system. Select the user who will be the sender of your distributed requests to the Microsoft SQL data source. We select the owner of our federated server instance, **db2fed32**.

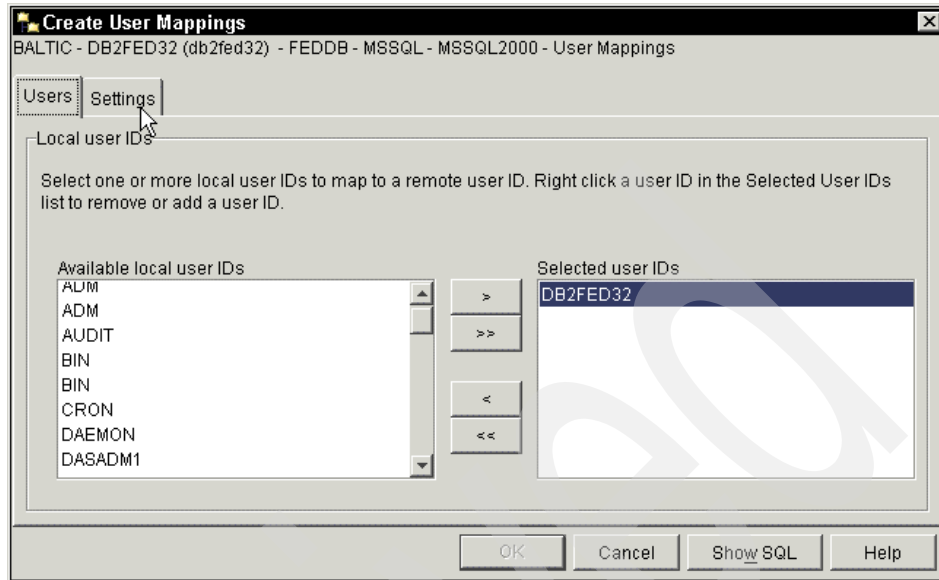


Figure 5-49 Microsoft SQL Server - Create user mappings

Switch to **Settings** as shown in Figure 5-50, to complete the user mapping. You need to identify the username and password to enable the federated system to connect to our DB2 Microsoft SQL data source.

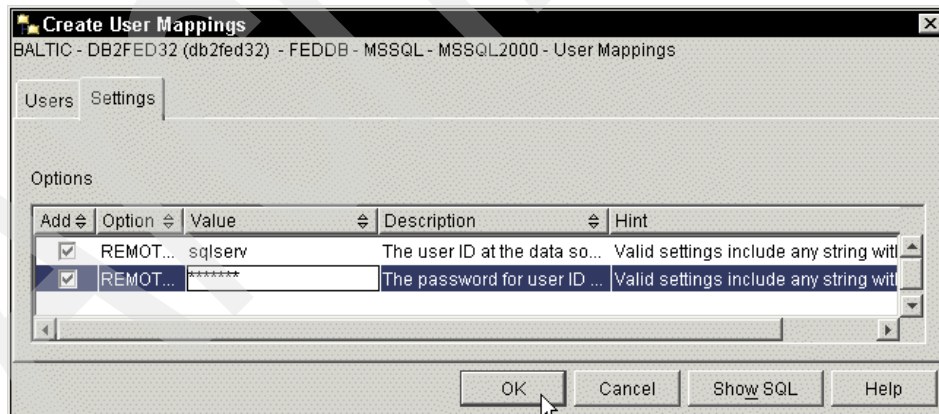


Figure 5-50 Microsoft SQL Server - Create user mappings - Settings

Example 5-12 shows the command line version to create the user mapping for your Informix instance. Additionally, you may check your user mapping definition in the DB2 system catalog with the SELECT statements listed below.

```
CONNECT TO FEDDB;  
CREATE USER MAPPING FOR "DB2FED32" SERVER "MSSQL2000"  
OPTIONS ( ADD REMOTE_AUTHID 'sqlserv', ADD REMOTE_PASSWORD 'sqlserv') ;  
  
SELECT * from SYSCAT.USEROPTIONS;
```

---

## 5.5.6 Altering Microsoft SQL Server user mappings

Use the ALTER USER MAPPING statement to modify a user mapping. It is used to change the authorization ID or password that is used at the Microsoft SQL data source:

```
alter user mapping for db2fed32 SERVER DEMO options  
    (set remote_authid 'mssql')  
alter user mapping for db2fed32 SERVER DEMO options  
    (set remote_password 'sanjose')
```

**Note:** The REMOTE\_AUTHID and REMOTE\_PASSWORD user options are case sensitive unless you set the fold\_ID and FOLD\_PW server options to U or L in the CREATE SERVER statement.

User mapping can also be altered in the DB2 Control Center.

## 5.5.7 Creating Microsoft SQL Server nicknames

After setting up the Microsoft SQL wrapper, the server definition, and the user mapping to our Microsoft SQL Server database, we finally need to create the actual link to a table located on our remote database as a nickname.

When you create a nickname for a Microsoft SQL Server table, catalog data from the remote server is retrieved and stored in the federated global catalog.

Steps to create a Microsoft SQL nickname:

1. Select the server **MSSQL2000**.
2. Right-click **Nicknames** for server **MSSQL2000** and click **Create**.

You see a dialog displayed in Figure 5-51 showing up. You have two possibilities to add a nickname. Either you click **Add** to manually add a nickname by specifying local and remote schema and table identification, or you use the **Discover** functionality, which we describe here.

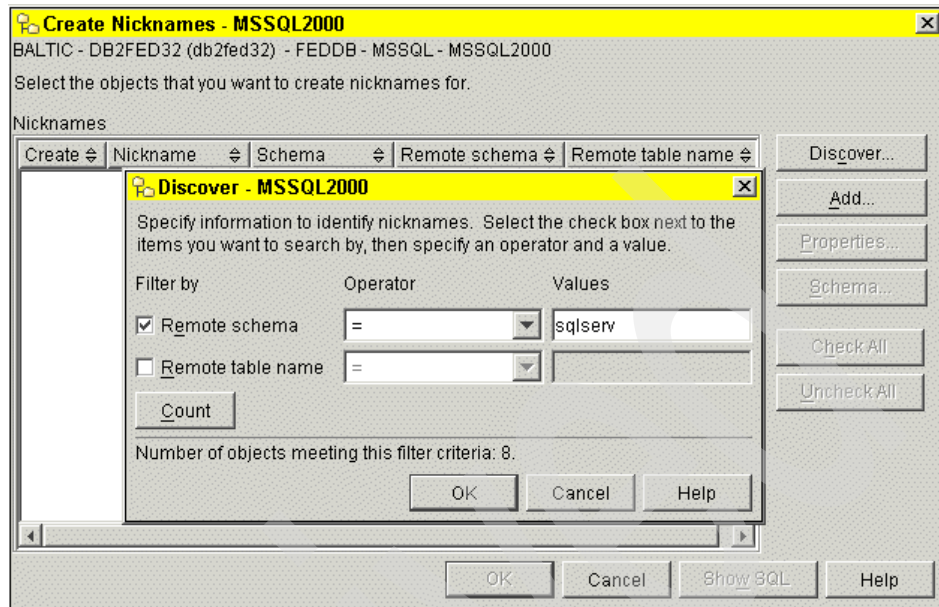


Figure 5-51 Microsoft SQL Server - Discover

Discover your remote Microsoft SQL Server data source by specifying either a **remote schema** or a specific **remote table** name you want to create a nickname for.

Choose a **filter** method, an **operator** and the comparing **value**, and click **Count**. Upon successful access to our Microsoft SQL Server database, the number of objects which meet the filter criteria will be displayed. Adjust your filter criteria if you find the number is too few or many. Click **OK** to list the actual objects in the dialog, as displayed in Figure 5-52.

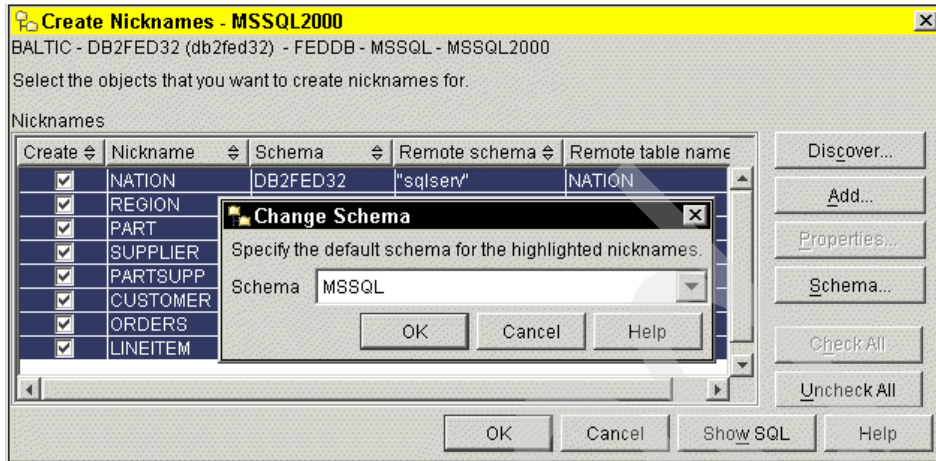


Figure 5-52 Microsoft SQL Server - Create nicknames - Change schema

If the Discover filter is used to add entries to the Create Nickname window, the default schema will be the user ID that is creating the nicknames. In our case, that is DB2FED32. We want to change this schema to adhere to our naming convention.

Use the **Schema** button to change the local schema for your Microsoft SQL Server nicknames.

**Tip:** We recommend that you use the same schema name for all DB2 Microsoft SQL nicknames.

Example 5-13 shows the command line version of creating the nicknames for your DB2 Microsoft SQL instance. Additionally, you may check the nickname definition in the DB2 system catalog with the select statements listed in the example.

Example 5-13 Microsoft SQL Server - Create nickname statements

```
CONNECT TO FEDDB;
CREATE NICKNAME MSSQL.NATION FOR MSSQL2000."sqlserv".NATION;
CREATE NICKNAME MSSQL.REGION FOR MSSQL2000."sqlserv".REGION;
CREATE NICKNAME MSSQL.PART FOR MSSQL2000."sqlserv".PART;
CREATE NICKNAME MSSQL.SUPPLIER FOR MSSQL2000."sqlserv".SUPPLIER;
CREATE NICKNAME MSSQL.PARTSUPP FOR MSSQL2000."sqlserv".PARTSUPP;
CREATE NICKNAME MSSQL.CUSTOMER FOR MSSQL2000."sqlserv".CUSTOMER;
CREATE NICKNAME MSSQL.ORDERS FOR MSSQL2000."sqlserv".ORDERS;
CREATE NICKNAME MSSQL.LINEITEM FOR MSSQL2000."sqlserv".LINEITEM;
```



```
SELECT * from SYSCAT.TABLES WHERE TABSCHEMA='MSSQL';
SELECT * from SYSCAT.TABOPTIONS WHERE TABSCHEMA='MSSQL';
SELECT * from SYSCAT.COLUMNS WHERE TABSCHEMA='MSSQL';
SELECT * from SYSCAT.COLOPTIONS WHERE TABSCHEMA='MSSQL';
SELECT * from SYSCAT.INDEXES WHERE TABSCHEMA='MSSQL';
SELECT * from SYSCAT.INDEXOPTIONS WHERE TABSCHEMA='MSSQL';
SELECT * from SYSCAT.KEYCOLUSE WHERE TABSCHEMA='MSSQL';
```

---

### 5.5.8 Altering Microsoft SQL Server nicknames

Use the ALTER NICKNAME statement to modify the federated database representing of a data source. Use this statement to:

- ▶ Change local data type of a column:  
`alter nickname MSSQL.REGION alter column r_name local type varchar (50)`
- ▶ Change local column name:  
`alter nickname MSSQL.REGION alter column r_name local name "R_REGIONNAME"`

## 5.6 Integrating XML data source

This section describes all steps necessary to integrate a data source based on XML data source into a DB2 Federated Server, and actually establishes a link between the XML file and your federated server:

- ▶ Introduction
- ▶ Configuration information for XML wrapper
- ▶ Creating the XML wrapper
- ▶ Creating the XML server
- ▶ Creating XML nicknames
- ▶ Altering XML nicknames (optional)

We use the Control Center on the Windows platform to perform the necessary administration steps.

### 5.6.1 Introduction

DB2 Information Integrator is a powerful tool to map XML documents into the relational world of DB2 without the need to load and unload data.

We can use this wrapper in these cases:

- ▶ The user desires to join XML data with other data (relational data or other nicknames).

- ▶ The user desires to keep the original XML intact (to avoid replicating data or Data that may change often or is composed on the fly).

The XML document may come from one of several sources:

- ▶ Individual text file (file path for a single XML file - ex: /home/db2user/mydata.xml)
- ▶ Directory of text files (directory path for multiple XML files - ex: /home/db2user/myxmldata/)
- ▶ Document stored in a column (table name and column name for XML data stored in column)
- ▶ Access through a URI (URI for remote XML file - ex: <http://www.mysite.com/mydata.xml>)

DB2 Information Integrator can process SQL statements that query data in a XML file as if it were contained in an ordinary relational table or view. This process is illustrated in Figure 5-53.

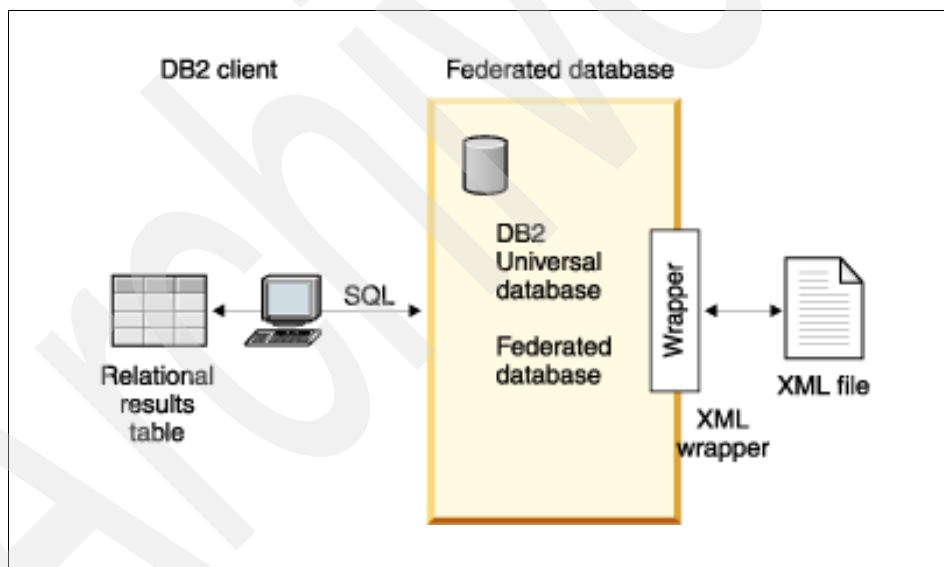


Figure 5-53 The XML data source

The XML wrapper can support the following data types:

- ▶ INTEGER
- ▶ SMALLINT
- ▶ FLOAT
- ▶ REAL
- ▶ DECIMAL

- ▶ CHAR
- ▶ VARCHAR
- ▶ DATE

CHAR or VARCHAR will can be declared as the data type for all nickname columns that are mapped to XML elements and attributes.

If the values for an XML element are attribute are valid numbers, then an appropriate DB2 numeric data type from the above list can be declared for the nickname column that is mapped to this element attribute.

If the value for the XML element or attribute are valid XML date values, then the DB2 DATE data type can be declared for the nickname column that is mapped to this element or attribute.

You must check this points before to decide to use or not XML wrapper:

- ▶ The complexity of the XML document
- ▶ The size of the XML document (limited by the available virtual memory)
- ▶ XML files must be accessible to the DB2 Information Integrator server (same machine or on a network accessible shares/mounts).
- ▶ INSERT, UPDATE, DELETE are not supported

## 5.6.2 Configuration information for XML wrapper

The information in Table 5-8 is necessary to integrate XML data source:

*Table 5-8 XML information*

Parameter	Value
AIX Federated Server	baltic.almaden.ibm.com
Local or Shared disk	/exchange/xml

On remote side, nothing needs to be configured, only the XML document needs to be accessible from the federated server.

The user ID that runs DB2 (in our case db2fed32) needs to be able to open in read the XML file.

## 5.6.3 Creating the XML wrapper

From the Create Wrapper window, specify the wrapper information:

On the Wrapper page, set the following server options:

- For the Data source field, select **XML**.
- For the Wrapper name field, type XML.

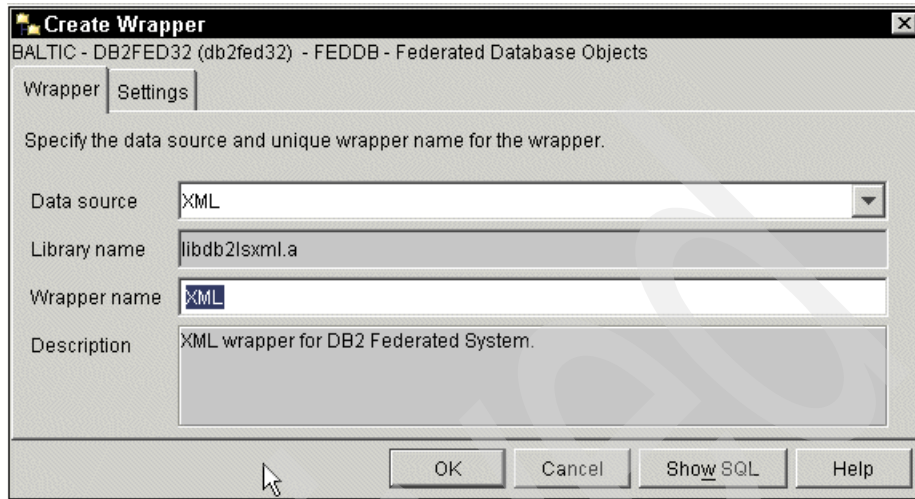


Figure 5-54 XML - Create Wrapper

We list in Example 5-14 the create wrapper statement:

Example 5-14 XML - Create wrapper statement

---

```
CONNECT TO FEDDB
CREATE WRAPPER XML LIBRARY 'libdb2lxml.a'
```

---

### 5.6.4 Creating the XML server

For XML, the Information Integrator server definition is required though it does not have any useful information such as the version and connection information in the server definitions for relational data sources.

From the Create Server window, specify the server information:

- For the **Name** field, type XMLSERV

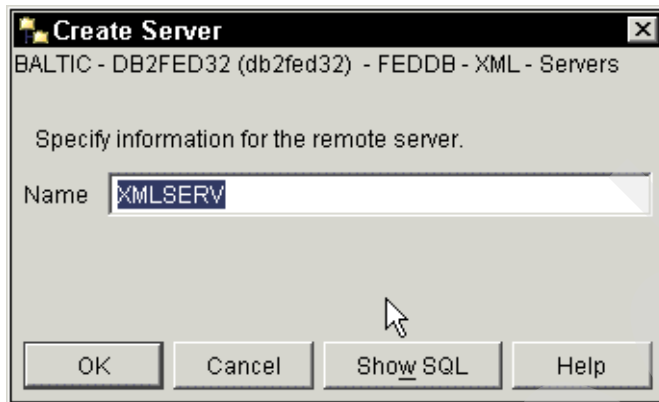


Figure 5-55 XML - Create Server

In Example 5-15 is the create server statement.

*Example 5-15 XML - Create server statement*

---

```
CONNECT TO FEDDB
CREATE SERVER XMLSERV WRAPPER XML
```

---

**Note:** For the XML server, we cannot update, add, or drop server options.

### 5.6.5 Creating XML nicknames

When you use the Control Center's Discover filter for XML (see Figure 5-56):

- ▶ An XML schema file or an XML document file can be specified in the first field of the window. If an XML schema file is specified in this field, then specify the XML document file for which the nickname is to be created in the Local file field in the middle of the window. The schema file specified in the top field must be appropriate for the document file specified in the middle File path field.
- ▶ The Discover filter can declare the columns of the nicknames to be created. It reads the XML schema file or the XML document file and creates an appropriate nickname definition with elements and attributes in the XML schema or document mapped to columns of the nickname. The Discover filter has a default behavior for declaring the data types of the nickname columns.
- ▶ If an XML schema is specified in the top field, then the Discover filter will use information in the XML schema to find out if any elements and attributes should be mapped to DB2 numeric or date data types; and the remaining

“string” columns will be mapped to DB2 varchar with the default length specified in the Default VARCHAR length field of the Discover window.

- If an XML file is specified in the top field of the Discover filter, then all elements and attributes of the document are mapped to DB2 VARCHAR data type, and the length is based on the value in the Default VARCHAR length field in the Discover window.

The Discover window does *not* create the nicknames. It only populates the Create Nickname window with entries; the nicknames are not created until you click **OK** in the Create Nickname window.

**Discover - XMLSERV**

BALTIC - DB2FED32 (db2fed32) - FEDDB - XML - XMLSERV

Specify the input file and and top-level element for the XML data source nicknames. You can also specify wrapper options and DDL options.

XML or schema input file  
Z:\xmlRegion.xsd

XML file top-level element  
region

XML wrapper options

Document type FILE

☒ Specify data source

Local file

Local directory

☐ Provide data source at query time

☐ Validate input document

DDL statement options

Default VARCHAR length 48

Table suffix \_NN

Primary key suffix \_ID

Foreign key suffix \_FID

Local schema DB2DAT

OK Cancel Help

Figure 5-56 XML - Create Server - Discover

After the Discover window has populated the Create Nickname window with entries for nicknames to be created, and before you click **OK** in this window to create the nicknames, you can change the data type specifications for the nickname columns before the nicknames are created. On the Create Nickname window, highlight a particular entry, and click **Change**. This will open the Change Nickname window displaying the columns of the nickname and showing the data type and length specification. You can highlight a column and click **Change** to open the Change column window. In that window, you can change the data type and length for the column.

**Note:** In our case, we use Samba Server to map the UNIX file system to a Windows Network Drive since we use DB2 Control Center on Windows.

On the XML schema file, you must define the element name as your XML file, since it is case sensitive (as shown in Example 5-16).

*Example 5-16 XML document*

---

```
<?xml version='1.0'?>
<regions>
  <region>
    <r_regionkey>0</r_regionkey>
    <r_name>AFRICA</r_name>
    <r_comment>special Tiresias about the furiously even dolphins are furi
    </r_comment>
  </region>
  <region>
    <r_regionkey>1</r_regionkey>
    <r_name>AMERICA</r_name>
    <r_comment>even, ironic theodolites according to the bold platelets wa
    </r_comment>
  </region>
</regions>
```

---

And, in Example 5-17.

*Example 5-17 XML - Define element*

---

```
?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by Mauricio A.
Hernandez (IBM Almaden Research Center) -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xsd:element name="region">
    <xsd:annotation>
      <xsd:documentation>Comment describing your root
element</xsd:documentation>
```

```

        </xsd:annotation>

        <xsd:complexType name="Region">
            <xsd:sequence>
                <xsd:element name="r_regionkey" type="xsd:integer"/>
                <xsd:element name="r_name" type="xsd:string"/>
                <xsd:element name="r_comment" type="xsd:string"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

</xsd:schema>

```

---

The XML or schema input file must be reachable from your Windows station. In our case these files are stored on the AIX server and reachable for DB2 Information Integrator.

For the **Specify data source** option, enter the XML file stored on the server side as shown in Figure 5-57. You can browse the Server file system by clicking the ... button.



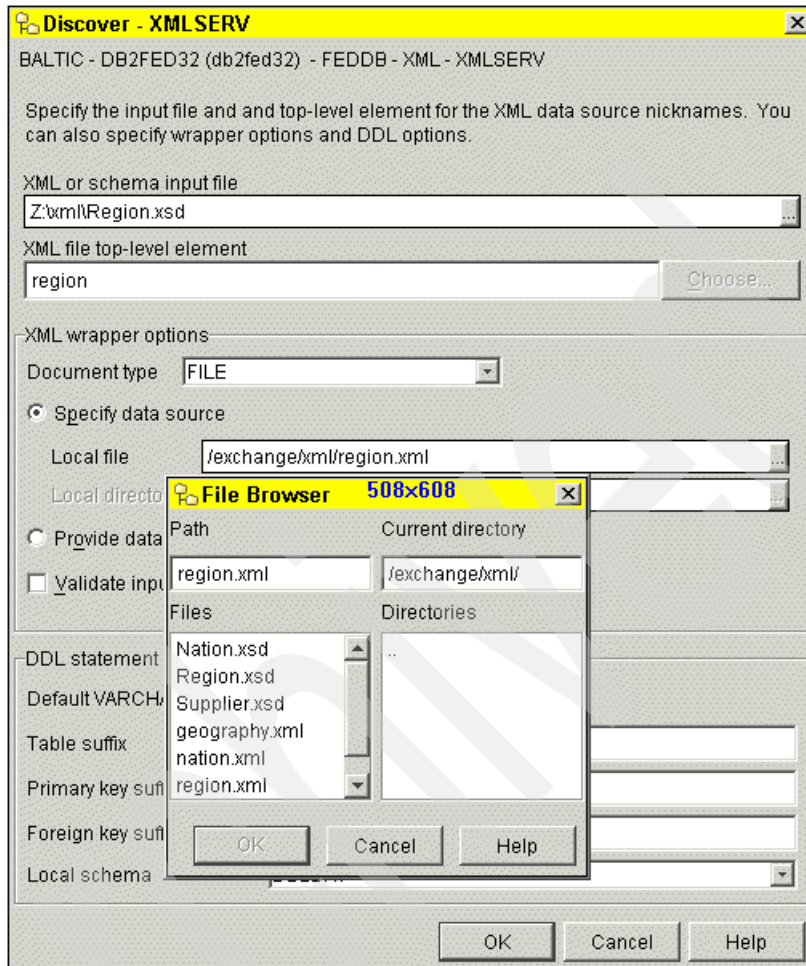


Figure 5-57 XML - File Browser

- For the **Local schema** option, type XML as shown Figure 5-58.

**Discover - XMLSERV**

BALTIC - DB2FED32 (db2fed32) - FEDDB - XML - XMLSERV

Specify the input file and top-level element for the XML data source nicknames. You can also specify wrapper options and DDL options.

XML or schema input file

XML file top-level element

XML wrapper options

Document type

☒ Specify data source

Local file

Local directory

☐ Provide data source at query time

☐ Validate input document

DDL statement options

Default VARCHAR length

Table suffix

Primary key suffix

Foreign key suffix

Local schema

Figure 5-58 XML - Discover

- You need to check or modify all Columns definition as shown in Figure 5-59.

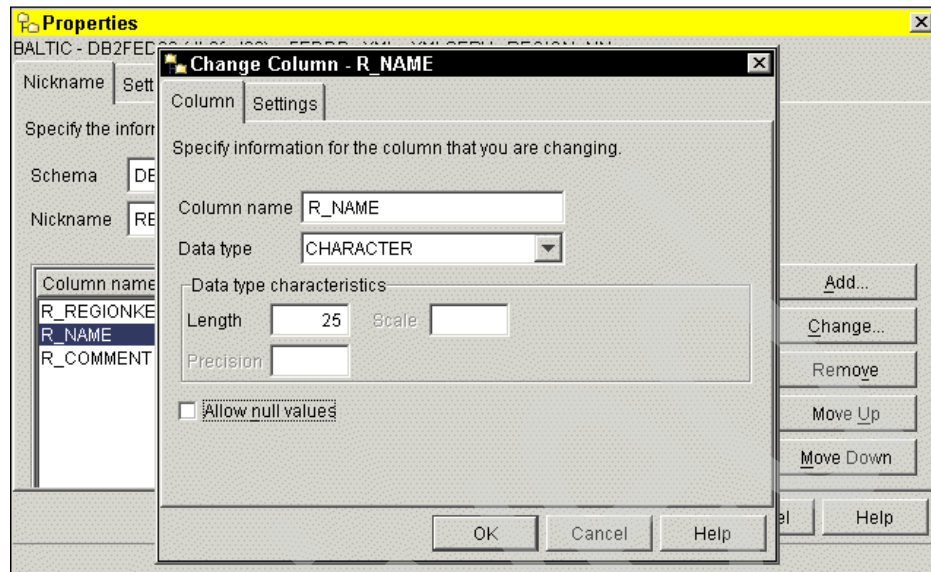


Figure 5-59 XML - Change column

- Check the **REGION** Nickname definition and click the **OK** button to create the nickname (see Figure 5-60).

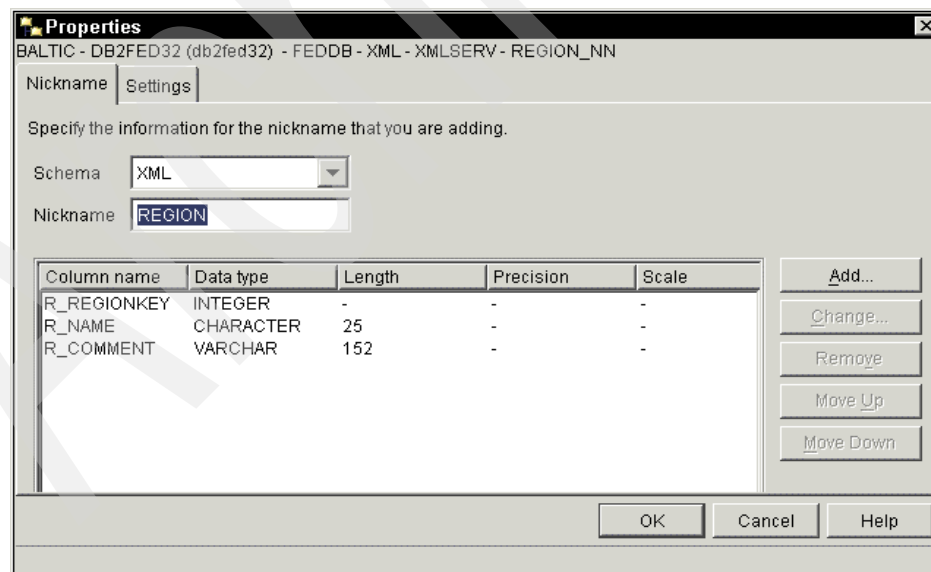
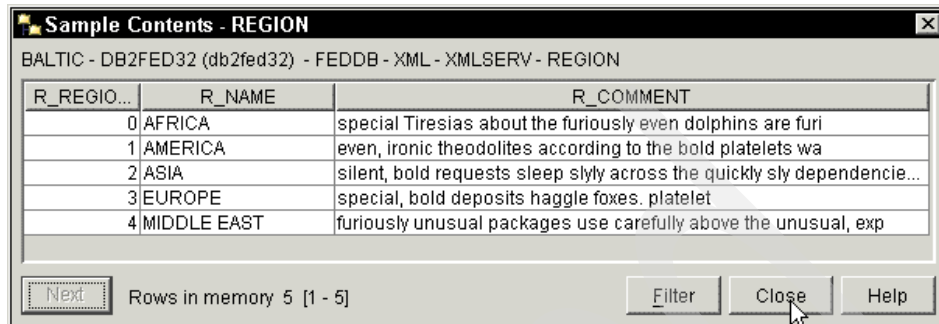


Figure 5-60 XML - Properties

- You can check the result by right-clicking **Sample Content** on the **REGION** nickname (see Figure 5-61).



R_REGIONKEY	R_NAME	R_COMMENT
0	AFRICA	special Tiresias about the furiously even dolphins are furi
1	AMERICA	even, ironic theodolites according to the bold platelets wa
2	ASIA	silent, bold requests sleep slyly across the quickly sly dependence...
3	EUROPE	special, bold deposits haggle foxes. platelet
4	MIDDLE EAST	furiously unusual packages use carefully above the unusual, exp

Figure 5-61 XML - Sample contents

In Example 5-18, we list the nickname definitions for three of the eight tables derived from the XML document.

*Example 5-18 XML - Create nickname statements*

```
CONNECT TO FEDDB
CREATE NICKNAME XML.REGION (
  R_REGIONKEY INTEGER NOT NULL OPTIONS(XPATH './r_regionkey/text()'),
  R_NAME CHARACTER (25)NOT NULL OPTIONS(XPATH './r_name/text()'),
  R_COMMENT VARCHAR (152)NOT NULL OPTIONS(XPATH './r_comment/text()')
  FOR SERVER "XMLSERV"
  OPTIONS(XPATH '//region',FILE_PATH '/exchange/xml/region.xml')
CREATE NICKNAME XML.NATION (
  N_NATIONKEY INTEGER NOT NULL OPTIONS (XPATH './n_nationkey/text()'),
  N_NAME CHAR(25)NOT NULL OPTIONS (XPATH './n_name/text()'),
  N_REGIONKEY INTEGER NOT NULL OPTIONS (XPATH './n_regionkey/text()'),
  N_COMMENT VARCHAR(152)NOT NULL OPTIONS (XPATH './n_comment/text()')
  FOR SERVER "XMLSERV"
  OPTIONS(XPATH '//nation',FILE_PATH '/exchange/xml/nation.xml')
CREATE NICKNAME XML.SUPPLIER (
  S_SUPPKEY INTEGER NOT NULL OPTIONS (XPATH './s_suppkey/text()'),
  S_NAME CHAR(25)NOT NULL OPTIONS (XPATH './s_name/text()'),
  S_ADDRESS VARCHAR(40)NOT NULL OPTIONS (XPATH './s_address/text()'),
  S_NATIONKEY INTEGER NOT NULL OPTIONS (XPATH './s_nationkey/text()'),
  S_PHONE CHAR(15)NOT NULL OPTIONS (XPATH './s_phone/text()'),
  S_ACCTBAL DOUBLE NOT NULL OPTIONS (XPATH './s_acctbal/text()'),
  S_COMMENT VARCHAR(101)NOT NULL OPTIONS (XPATH './s_comment/text()')
  FOR SERVER "XMLSERV"
  OPTIONS(XPATH '//supplier',FILE_PATH '/exchange/xml/supplier.xml')
```

The FILE\_PATH option is only used in the top-level nickname, and it defines where the data comes from.

If you do not define the FILE\_PATH, the document file name may come as a parameter, and you need to specify this information when you are writing queries as shown in Example 5-19.

*Example 5-19 XML - FILE\_PATH not defined*

---

```
CREATE NICKNAME XML.REGION_FILE (  
    doc VARCHAR(100) OPTIONS(DOCUMENT 'FILE'),  
    R_REGIONKEY INTEGER NOT NULL OPTIONS(XPATH './r_regionkey/text()'),  
    R_NAME CHARACTER (25) NOT NULL OPTIONS(XPATH './r_name/text()'),  
    R_COMMENT VARCHAR (152) NOT NULL OPTIONS(XPATH './r_comment/text()'))  
    FOR SERVER "XMLSERV"  
    OPTIONS(XPATH '//region')  
  
SELECT r.r_name FROM xml.region_file r  
    WHERE r.doc = '/exchange/xml/region.xml'  
    AND r.r_regionkey = 2  
  
CREATE NICKNAME XML.REGION_URL (  
    doc VARCHAR(100) OPTIONS(DOCUMENT 'URI'),  
    R_REGIONKEY INTEGER NOT NULL OPTIONS(XPATH './r_regionkey/text()'),  
    R_NAME CHARACTER (25) NOT NULL OPTIONS(XPATH './r_name/text()'),  
    R_COMMENT VARCHAR (152) NOT NULL OPTIONS(XPATH './r_comment/text()'))  
    FOR SERVER "XMLSERV"  
    OPTIONS(XPATH '//region')  
  
SELECT r.r_name FROM xml.region_url r  
    WHERE r.doc = 'http://www.mysite.com/region.xml'  
    AND r.r_regionkey = 2
```

---

The XPATH option for each nickname defines the context for the nickname data. In the example, the XPATH option for the XML.REGION nickname is '//region'. This tells the XML wrapper to read elements/attributes in their relation to the <region> element to get the data for the columns of this nickname. from the point of all occurrences.

The XPATH option for each column defines the element (or attribute) that contains the column data. The value is appended to the XPATH value for the nickname to find the element or attribute for a particular nickname column. In the example, the XPATH option for the XML.REGION nickname is //region and the XPATH option for the column R\_REGIONKEY is ./r\_regionkey/text(). The two values are appended so that the XML wrapper finds the <r\_regionkey> element under the element <region> to get the values for the R\_REGIONKEY column of the nickname.

Since DB2 is a cost engine, we add some parameters for the XML wrapper:

- ▶ **INSTANCE\_PARSE\_TIME** (for root-nicknames only): Time required in milliseconds to parse the XML document
- ▶ **XPATH\_EVAL\_TIME**: Time required in milliseconds to evaluate a nickname XPath expression

You can use the default values or modify it at CREATE NICKNAME time to optimize your queries.

You can also set the **DB2\_DJ\_COMM** profile variable to load the wrapper during the database startup:

```
db2set DB2_DJ_COMM='libdb21xml.a'
```

### Known problem and workaround

When an XML document's size or complexity exceeds a threshold, the memory requirement to process a query will exceed the virtual storage available. When this occurs, the query will terminate and the XML wrapper returns:

```
SQL0901N  The SQL statement failed because of a non-severe system
error.Subsequent SQL statements can be processed.  (Reason "Unspecified
exception while parsing input document".)  SQLSTATE=58004
```

To avoid this problem, you can set a new **STREAMING** nickname option for the XML wrapper (available starting with DB2 V8.1 FixPak 3). The **STREAMING** option (see Example 5-20) specifies whether the XML source document is separated into logical fragments that correspond to the node that matches the XPath expression of the nickname. The XML wrapper then parses and processes the XML source data fragment by fragment, reducing the total memory required to read the document. You can specify streaming for any XML source document (FILE, DIRECTORY, URI, or COLUMN). This option is accepted only for columns of the root nickname (the nickname that identifies the elements at the top level of the XML document). The default streaming value is NO.

**Note:** Do not set the **STREAMING** parameter to YES if you also set the **VALIDATE** parameter to YES. If you set both parameters to YES, you will receive an error message.

We list in Example 5-20 the nickname statement with **STREAMING** option.

```
CREATE NICKNAME XML.REGION (  
    R_REGIONKEY INTEGER NOT NULL OPTIONS(XPATH './r_regionkey/text()'),  
    R_NAME CHARACTER (25) NOT NULL OPTIONS(XPATH './r_name/text()'),  
    R_COMMENT VARCHAR (152) NOT NULL OPTIONS(XPATH './r_comment/text()'))  
FOR SERVER "XMLSERV"  
OPTIONS(XPATH '//region',  
    FILE_PATH '/exchange/xml/region.xml', STREAMING 'YES')
```

---

## 5.6.6 Altering XML nicknames

You can use the ALTER NICKNAME statement to modify the federated database representation of a data source. Use this statement to:

- ▶ Change local data type of a column:  
`alter nickname XML.REGION alter column r_name local type varchar (25)`
- ▶ Change local column name:  
`alter nickname XML.REGION alter column r_name local name R_REGIONNAME`
- ▶ Add the streaming option (since DB2 V8.1 **FixPak 3**):  
`alter nickname XML.REGION options (ADD STREAMING 'YES')`
- ▶ Drop the streaming option:  
`alter nickname XML.REGION options (DROP STREAMING)`

## 5.7 Integrating table-structured files

In this section we describe how to integrate table-structured (flat) files data sources in the DB2 Information Integrator environment. We do so by showing the following steps:

1. Introduction
2. Table-structured file configuration information
3. Creating the table-structured file wrapper
4. Creating the table-structured file server
5. Creating table-structured file nicknames
6. Altering table-structured file nicknames

### 5.7.1 Introduction

The table-structured (or flat file) wrapper is a powerful tool to map table-structured files into the relational world of DB2 Information Integrator without the need to load/unload data.

A table-structured file has a regular structure consisting of a series of records, where each record contains the same number of fields, separated by an arbitrary delimiter. Null values are represented by two delimiters next to each other.

Table-structured files can be sorted or unsorted but we recommend to use sorted files.

We can use the wrapper in the following cases:

- ▶ the user desires to join table-structured files data with other data (relational data or other nicknames)
- ▶ the user desires to keep the original table-structured files intact (to avoid replicating data or data that may change often or is composed on the fly)

DB2 Information Integrator can process SQL statements that query data in a table-structured file as if it were contained in an ordinary relational table or view. This enables data in a table-structured file to be joined with relational data or data in other table-structured files. This process is illustrated in Figure 5-62.

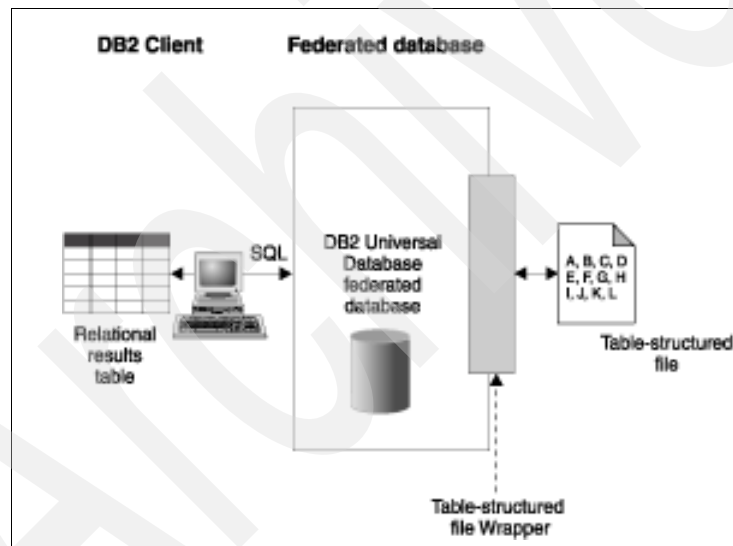


Figure 5-62 Table-structured file data source

The wrapper can support these data types:

- ▶ CHARACTER
- ▶ VARCHAR
- ▶ INTEGER
- ▶ SMALLINT
- ▶ DOUBLE



- ▶ REAL
- ▶ DECIMAL

CHARACTER and VARCHAR can be declared as the data type any column of a nickname for a flat file.

INTEGER, SMALLINT, DOUBLE, REAL, and DECIMAL can be declared as the data type for a column of nickname for a flat file if the value in all occurrences of that field in every record of the file meet the criteria for the values of this data type.

You must verify the following points before deciding to use table-structured files wrapper:

- ▶ Data files must be accessible to the DB2 Information Integrator server: they must reside on the same machine or on a network accessible shares/mounts.
- ▶ The user ID that runs DB2 (in our case, 'db2fed32') must be able to open and read the file.
- ▶ The data schema for the data file (number of column and columns data types) must be consistent throughout the file.
- ▶ The data can only be sorted on one column in ascending order.
- ▶ INSERT, UPDATE, DELETE are not supported.

## 5.7.2 Table-structured file configuration information

The information in Table 5-9 is necessary to integrate table-structured files data source.

*Table 5-9 Table-structured file - Information*

Parameter	Value
AIX Federated Server	baltic.almaden.ibm.com
Local or Shared disk	/exchange/flatfiles

Since we are on a UNIX platform, table-structured files must be converted to a UNIX format (FTP in ASCII mode).

**Note:** Do not put the column delimiter at the end of each row. Samba server does not convert files from Windows to UNIX, and from UNIX to Windows.

### 5.7.3 Creating the table-structured file wrapper

From the Create Wrapper window, specify the wrapper information:

On the Wrapper page, set the following server options:

1. For the **Data source** field, select the table-structured files.
2. For the **Wrapper name** field, type FLATFILES

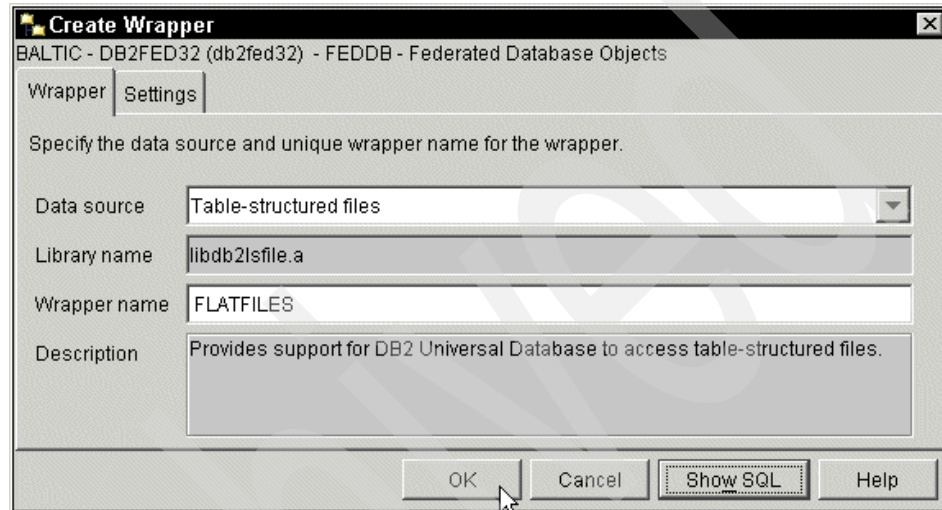


Figure 5-63 Table-structured file - Create Wrapper

In Example 5-21 we list the wrapper statement.

Example 5-21 Table-structured file - Create wrapper statement

---

```
CONNECT TO FEDDB
CREATE WRAPPER "FLATFILES" LIBRARY 'libdb2lsfile.a'
```

---

### 5.7.4 Creating the table-structured file server

For the table-structure (flat file) wrapper, the server definition is required though it does not contain any useful information, such as the version and connection information in a server definition for a relational data source.

From the Create Server window (see Figure 5-64) specify the server information:

- For the **Name** field, type FLATSERV

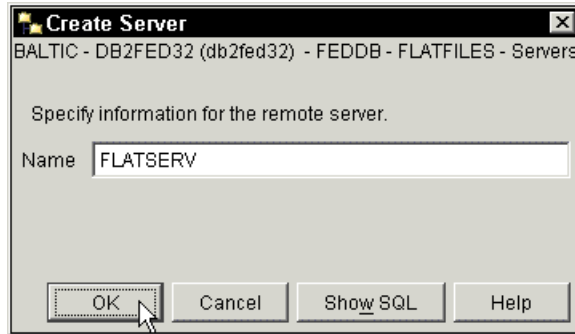


Figure 5-64 Table-structured file - Create Server

In Example 5-22 we list the server statement.

Example 5-22 Table-structured file - Create server statement

---

```
CONNECT TO FEDDB
CREATE SERVER FLATSERV WRAPPER "FLATFILES"
```

---

**Note:** For a table-structured file server, we cannot update, add, or drop server options.

### 5.7.5 Creating table-structured file nicknames

We assume that the contents of the file for which a nickname is being created, are as listed in Example 5-23.

Example 5-23 Contents of file 'region.tbl'

---

```
0|AFRICA|special Tiresias about the furiously even dolphins are fur
1|AMERICA|even, ironic theodolites according to the bold platelets wa
2|ASIA|silent, bold requests sleep slyly across the quickly sly dependencies
3|EUROPE|special, bold deposits haggle foxes. platelet
4|MIDDLE EAST|furiously unusual packages use carefully above the unusual, exp
```

---

We can see there are three fields:

- ▶ The first field contains only numbers (0 - 4), and they are unique in each record, and increase from one record to the next, and from the first record to the last in the file. This is a criteria for declaring the file as “sorted.”
- ▶ The second field contains characters.
- ▶ The third field contains characters.

The field delimiter is the character |

If we tested, we would find that each record ends with a line-feed, including the last record.

From the Create Nickname window (see Figure 5-65) click **Discover** to open the Discover Nickname window, and set the following:

- ▶ For the **Local schema** option, type FLAT
- ▶ For the **File extension** option, type tbl
- ▶ For the **Directory path** option, type /exchange/flatfiles

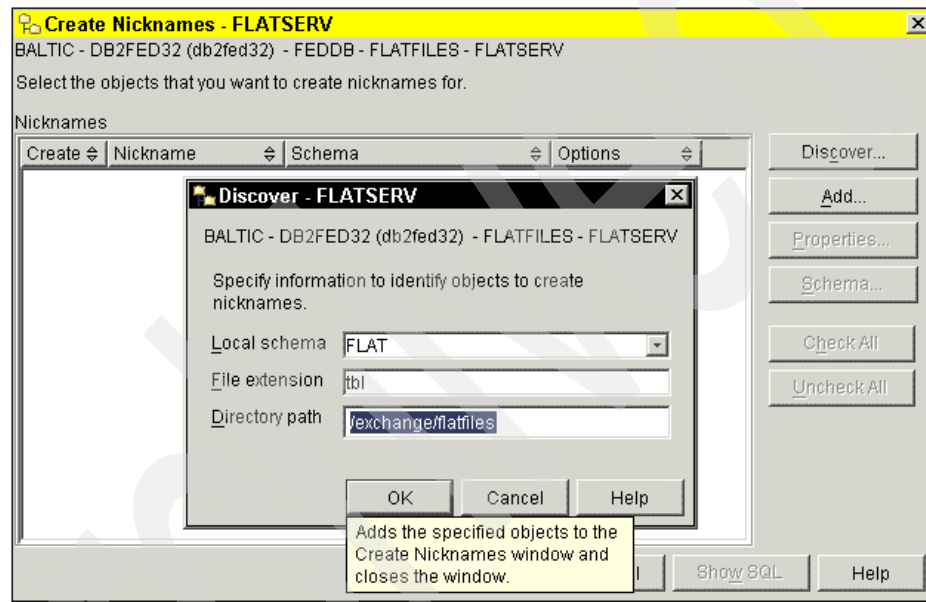


Figure 5-65 Table-structured file - Create nicknames

- ▶ Click **OK**, then a list of nicknames appears. We decided to customize **REGION**, and clicked the **Properties** button (see Figure 5-66).
- ▶ You must add columns and define names and data types.

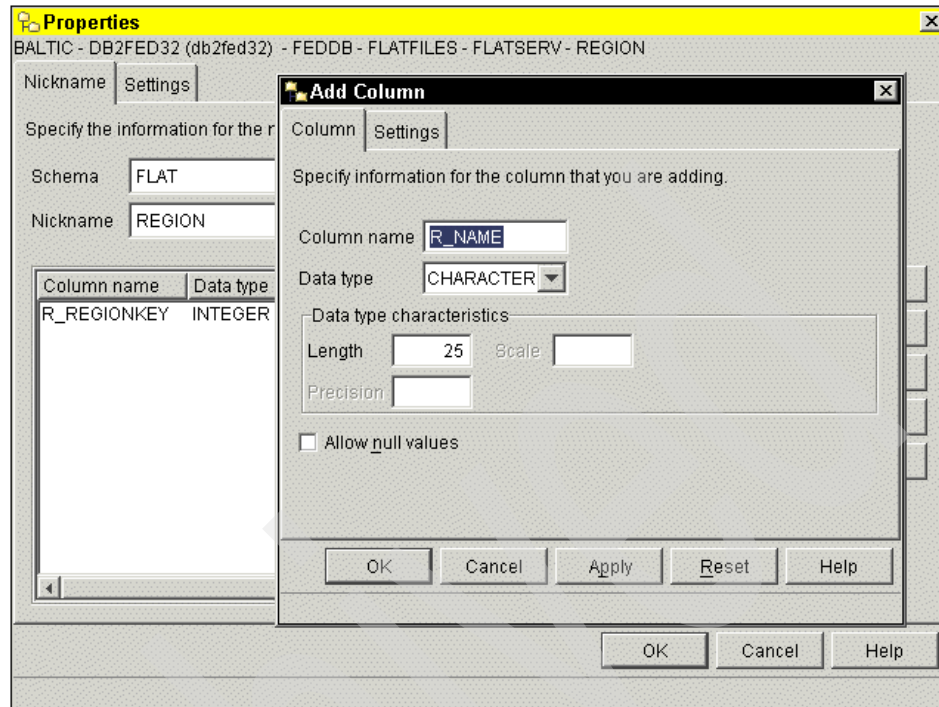


Figure 5-66 Table-structured file - Add column

- On the Setting page, shown in Figure 5-67, we set the **Column Delimiter** to | and the **File Path** to /exchange/flatfiles/region.tbl

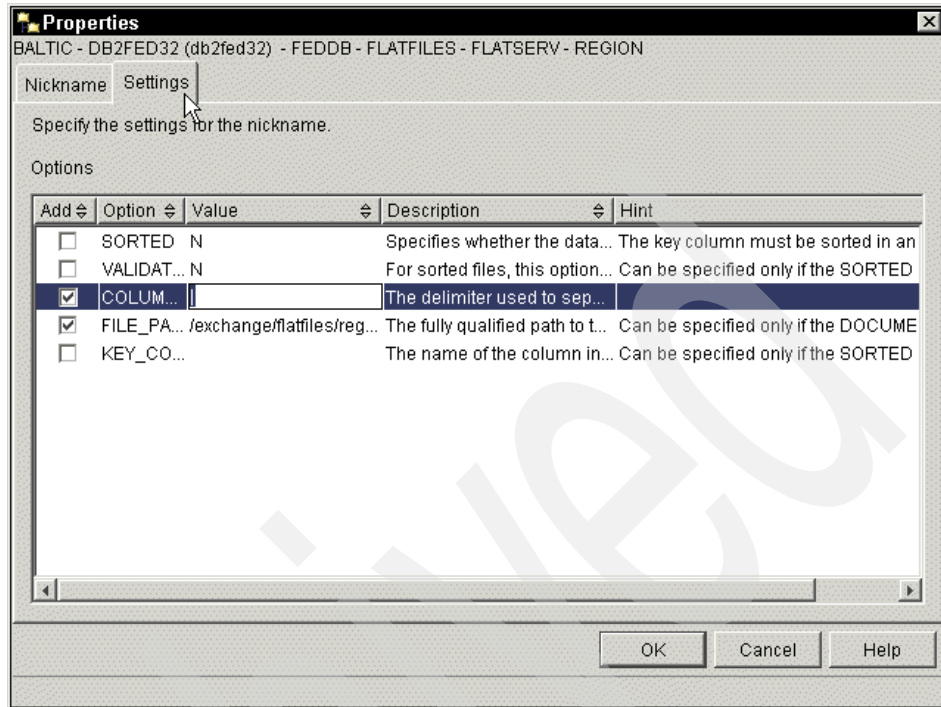


Figure 5-67 Table-structured file - Properties

- Click **OK**, then a list of nicknames appears. Then Click **OK** again to create the nickname in Figure 5-68.

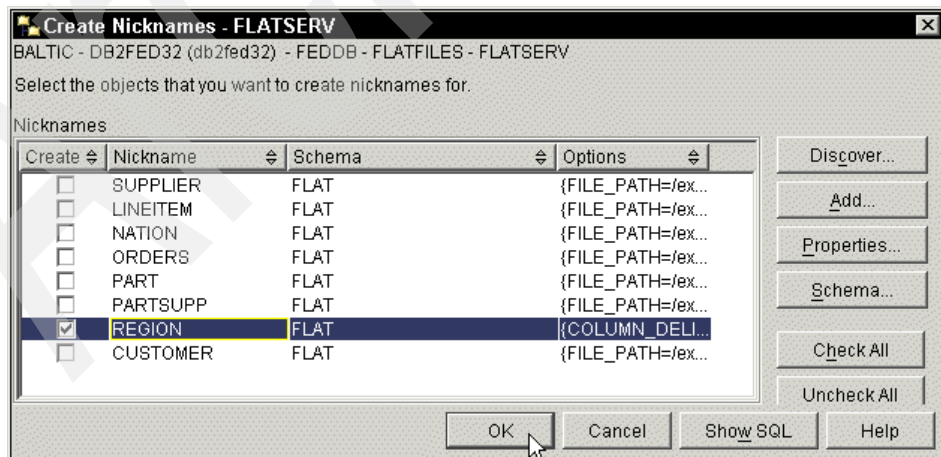


Figure 5-68 Table-structured file - Select objects for nicknames

- You can check the result in Figure 5-69. Right-click **Sample Content** on the **REGION** nickname:

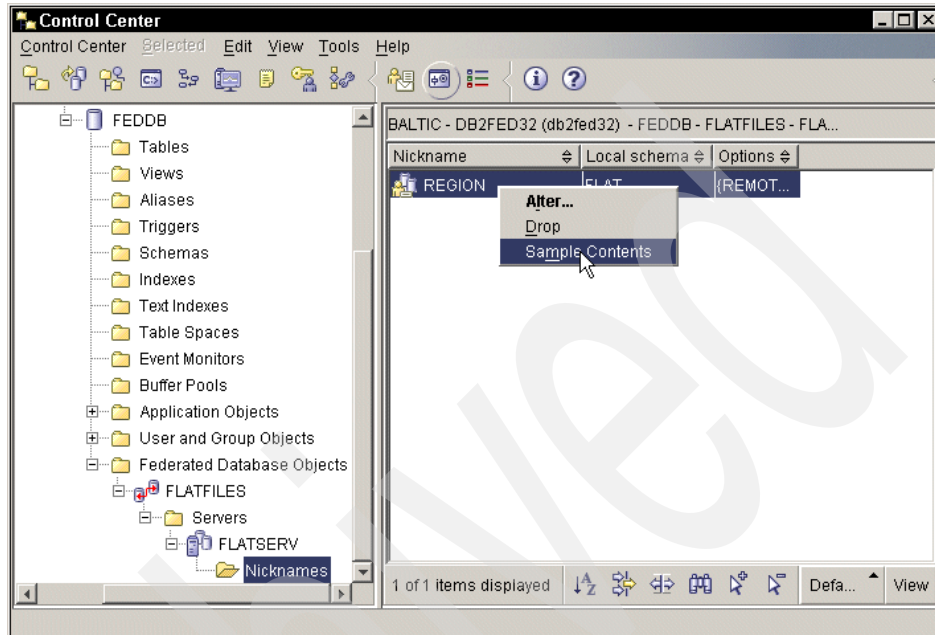


Figure 5-69 Control Center - Again nicknames

In Example 5-24 we list the nickname definitions for one of the eight tables derived from the table-structured files:

Example 5-24 Table-structured file - Create nickname statement

```
CONNECT TO FEDDB
CREATE NICKNAME FLAT.REGION (
  R_REGIONKEY INTEGER NOT NULL ,
  R_NAME CHARACTER (25) NOT NULL ,
  R_COMMENT VARCHAR (152) NOT NULL )
FOR SERVER "FLATSERV"
OPTIONS(COLUMN_DELIMITER '|' ,
  FILE_PATH '/exchange/flatfiles/region.tbl')
```

## 5.7.6 Altering table-structured file nicknames

Use the ALTER NICKNAME statement to modify the federated database representation of a data source. Use this statement to:

- Change the local data type of a column:

```
alter nickname FLAT.REGION alter column r_name local type varchar (25)
```

- Change local column name:

```
alter nickname FLAT.REGION alter column r_name local name "R_REGIONNAME"
```

## 5.8 Integrating Microsoft Excel

In this section we describe how to integrate data contained in the Microsoft Excel worksheet. We look at the following steps:

1. Introduction
2. Configuration information for ODBC wrapper
3. Setting Excel ODBC on Windows
4. Setting OpenLink client on AIX
5. Creating ODBC wrappers
6. Creating ODBC servers
7. Altering the ODBC server
8. Creating the ODBC nickname
9. Altering ODBC nicknames

### 5.8.1 Introduction

DB2 Information Integrator provides wrappers that access enterprise data and integrate it with data from other sources. One popular source of data is Microsoft Excel worksheets. Excel worksheets are used in many different types of businesses and contain a wide variety of data. DB2 Information Integrator provides two wrappers that you can use to access data in Excel worksheets: the Excel wrapper and the ODBC wrapper. Both of these wrappers enable you to access Excel worksheet data, but each wrapper has different capabilities. In this section we present the ODBC wrapper methods for accessing Excel data, and provide useful information that will help you decide which wrapper is appropriate for your specific needs.

For ODBC wrapper, you will not have to install any application software to access the worksheets since the Microsoft Excel driver is shipped with the Windows operating system. Additionally, you have to consider the Microsoft Excel application license requirements. If you use the Excel wrapper, the Microsoft Excel application must be installed on the federated server. You should review the Microsoft Excel license terms and conditions for specific information for your environment.

When you use the ODBC wrapper, the data types are determined by the Microsoft Excel driver. The Microsoft Excel driver maps the Microsoft Excel data types to ODBC data types. Then the Information Integrator ODBC wrapper maps



the ODBC data types (**ODBC 3.0** only) to DB2 data types. The DB2 data types for each column are stored in the federated database catalog table. Table 5-10 below shows the data type mappings.

*Table 5-10 Excel data type mapping*

Excel data type	ODBC data type	DB2 data type
CURRENCY	SQL_NUMERIC	DECIMAL/DOUBLE
DATETIME	SQL_TIMESTAMP	TIMESTAMP
LOGICAL	SQL_BIT	SMALLINT
NUMBER	SQL_DOUBLE	DOUBLE
TEXT	SQL_VARCHAR	VARCHAR

Because the ODBC wrapper supports many different data types, you can manually alter the data type mappings to other DB2 data types. Issue an ALTER NICKNAME statement to change the local data types. The list of data types that are available through the ODBC wrapper is comprehensive, and includes data types such as LOBs and other double-byte data types. However, there is the potential of running into a data type mismatch when you alter the local data type to something other than the original mapping. Additionally, issuing these ALTER NICKNAME statements can be time consuming when you have a large number of columns or nicknames. If you alter the local type of a nickname column, be sure that the new type will work for all the values for the corresponding column of the Excel spreadsheet.

The Excel wrapper supports four DB2 data types: DATE, DOUBLE, INTEGER, and VARCHAR. When you issue the CREATE NICKNAME statement, you specify one of these data types for each of the columns in the worksheet that you are accessing.

Using the ODBC wrapper, you can perform insert and update operations on the worksheet. The Microsoft Excel driver does not support delete operations. To delete data from the worksheet, you must open the worksheet directly to make the changes. The Excel wrapper is a read-only wrapper. To insert, update, or delete data from the worksheet, you must open the worksheet directly using Excel to make the changes.

The ODBC wrapper cannot access a worksheet when the workbook is opened by a user or an application (that is Excel) in exclusive (read and write) mode. However, if the ODBC wrapper opens the workbook before applications and users open the workbook, the applications and users can open the workbook in read-only mode. Since the Excel wrapper supports read-only operations, the

wrapper can access the worksheet when other applications and users *already have the workbook open in Exclusive mode*.

Using ODBC wrapper, you can access data from any of the worksheets within a workbook. The Microsoft Excel driver interprets the workbook as a database and each worksheet within the workbook as a table. Because the Excel wrapper interprets the workbook as a table, you can access only the first worksheet in the workbook.

When you use the ODBC wrapper to access Excel data, you are limited by what the Microsoft Excel driver supports. The Microsoft Excel driver is strict about the worksheet format. The driver assumes that the first non-blank row will always contain the column labels. If the first non-blank row contains data instead, the data in the first row is treated as the column labels. This results in losing the data in the first row whenever you access the data. You can overcome this limitation by modifying your worksheet and inserting a row of column labels into the worksheet. If your worksheet has several rows of titles or column labels, you must use a named range to explicitly designate the location of the data within the worksheet. The ODBC recognizes only one row of labels, the first row in the range. No blank rows are allowed between the labels and the data. The named range must include only one row of column labels.

When you use the Excel wrapper to access Excel data, the wrapper expects that there are no column labels or titles in the worksheet. However, if the worksheet does include column labels, you can use the RANGE option in the CREATE NICKNAME statement, and exclude all titles and column labels from the range.

Through the Microsoft Excel driver, the ODBC wrapper allows predicates and aggregate functions to be pushed down to the data source for processing. The driver can also handle joins of worksheets. Pushdown processing can improve performance. All of the data source rows are not returned back to DB2 when the query references a worksheet. Predicates and aggregate functions can be processed by the data source.

The Excel wrapper does not allow any functions or predicates to be pushed down to the data source. The predicates and functions must be processed by the federated server when the rows are returned. Processing predicates and functions at the federated server instead of pushing down the process to the data source can have a negative impact on performance.

For Excel data source, it is recommended to use the ODBC wrapper to get better performance and more DB2 Information Integrator features.

You must check these points before you decide to use or to not use Excel wrapper or ODBC wrapper with an Excel worksheet:

- The size of the Excel worksheet
- Insert/Update/Delete is not supported

In our case, since Information Integrator is on AIX where there is no Excel ODBC driver, nor Microsoft Excel, we need to install a multi-tier ODBC provider, or install a new DB2 Information Integrator on Windows since we have already decided to install DB2 Information Integrator on AIX. We decided to use the ODBC wrapper with an OpenLink Software solution, as shown in Figure 5-70. For more information on OpenLink Software products, and specifically MultiTier ODBC Client 5.0, go to the Web site:

<http://www.openlink.co.uk>

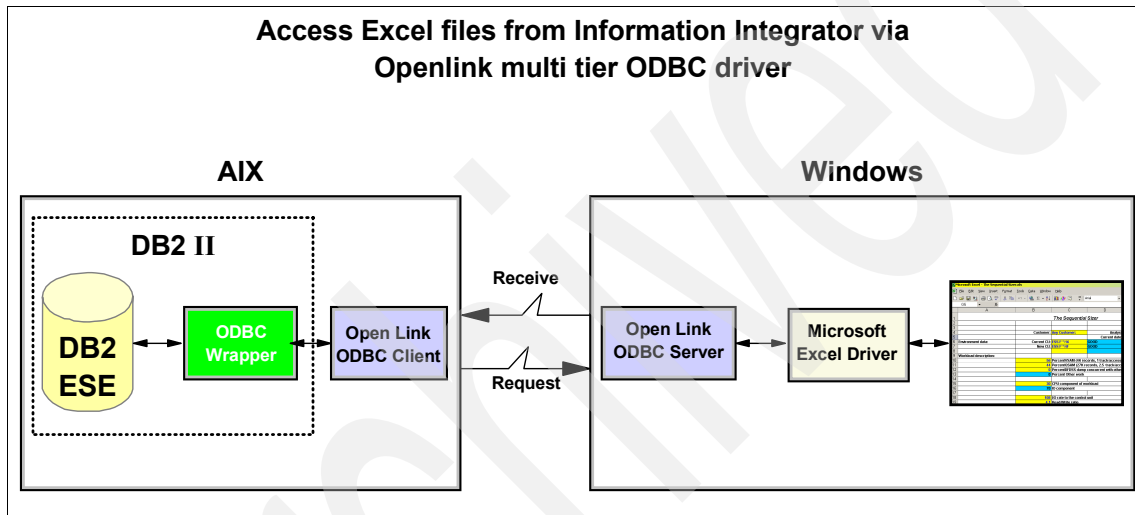


Figure 5-70 Excel files through OpenLink

## 5.8.2 Configuration information for ODBC wrapper

The information in Table 5-11 is necessary to integrate the Excel data source.

Table 5-11 Excel information

Parameter	Value
Windows Server	radon.almaden.ibm.com
ODBC Driver	Microsoft Excel Driver

**Note:** Excel files must be stored on a local disk, not on a shared disk.

The information in Table 5-12 is necessary to integrate OpenLink ODBC Server driver.

Table 5-12 OpenLink Server ODBC driver information

Parameter	Value
Windows Server	radon.almaden.ibm.com
Multi-tier ODBC Provider	OpenLink Server
Communication Port	5000
User/Password	admin/admin

The information in Table 5-13 is necessary to integrate the OpenLink ODBC client driver.

Table 5-13 OpenLink Client ODBC driver information

Parameter	Value
AIX Federated Server	baltic.almaden.ibm.com
Multi-tier ODBC Provider	OpenLink Client
User/Password	admin/admin
Administration Port	8000

### 5.8.3 Setting Excel ODBC on Windows

A 32-bit driver that is shipped with the Windows operating system. The Microsoft Excel driver is one of the Microsoft ODBC desktop database drivers. The Microsoft ODBC desktop database drivers are a Microsoft Jet-based set of ODBC drivers. The Microsoft Excel driver allows ODBC applications to access Excel worksheets using the ODBC interface.

To configure the Data Source Name (DSN):

1. Open the ODBC Data Source Administrator.
2. Click the **System DSN** tab.
3. Click **Add** to add a new DSN.
4. Select the driver: **Microsoft Excel Driver(\*.xls)**
5. Click the **Finish** button.
6. Fill in the Data Source Name field: **REGION**
7. Check Excel Version: **Excel 97-2000**
8. Click the **Select Workbook** button to select an Excel file location, then click **OK**.
9. Click the **OK** button to create your REGION DSN.

Repeat these steps for each workbook that you plan to access.

## 5.8.4 Setting OpenLink client on AIX

To configure the Data Source Name, it is recommended to use the Web Admin Interface with the form, not the Wizard interface

To use the Web Admin Interface, you must start the server by the command in Example 5-25.

*Example 5-25 Activating OpenLink server*

---

```
iodbc-admin-httpd.sh start
```

```
iodbc-admin-httpd.sh stop
```

---

When the Web Admin Client is started, use a browser (<http://localhost:8000/> (default)) to configure OpenLink (user admin/<passwd>):

1. Click **Client Components Administration -> Configure ODBC Data Sources by Form -> Configure ODBC Data Sources**
2. Click **Add** to add a new DSN.
3. Click **New** for "OpenLink Generic ODBC Driver"
4. You must fill in these fields:
  - a. Name: REGIONXLS
  - b. Database Name: REGION (this is the Workbook DSN name created on Windows)
  - c. Check Read Only connection
  - d. Server Type: **ODBC**
  - e. Host Name: radon.almaden.ibm.com:5000
5. Click the **Add** button to create your REGIONXLS DSN.
6. Click **Test DSN:REGIONXLS** to check the Connection.

This will add an entry for the REGIONXLS to the .odbc.ini file.

Repeat this step for each workbook that you plan to access.

## 5.8.5 Creating ODBC wrappers

From the Create Wrapper window, specify the wrapper information:

On the Wrapper page, set the following server options:

1. From the **Data source** field, select **ODBC**.
2. For the **Wrapper name** field, type ODBCXLS

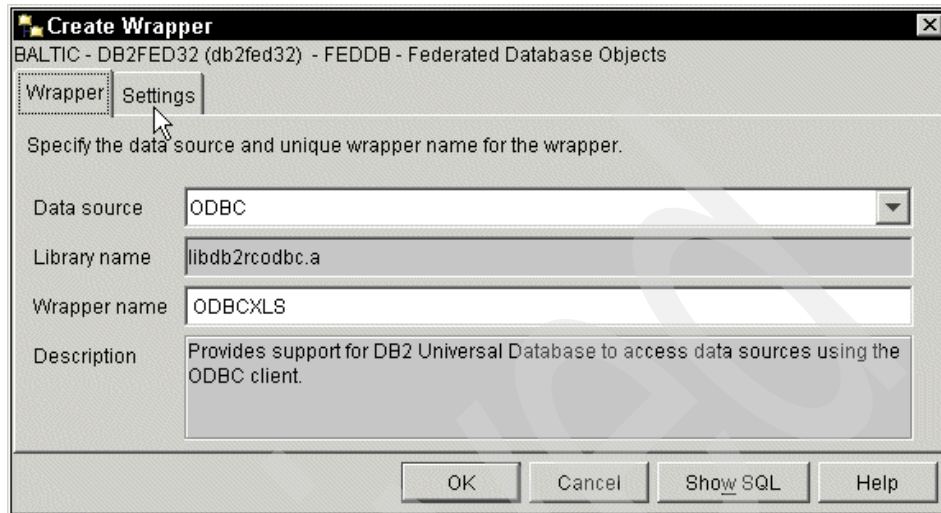


Figure 5-71 ODBC - Create wrapper

On the Settings pages, ensure that the DB2\_FENCED option is set to N (the default value). Set the MODULE option with the full library path for the ODBC driver.

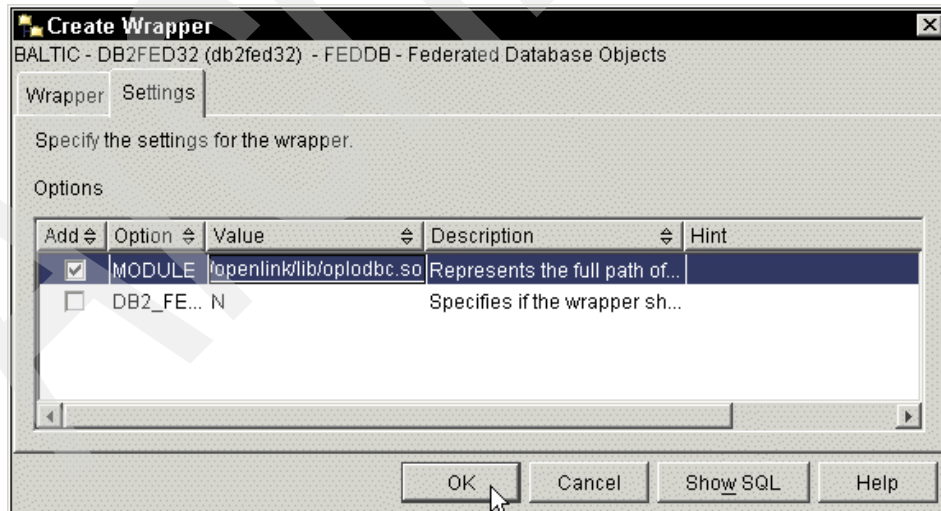


Figure 5-72 ODBC - Create wrapper - Settings

In Example 5-26 we list the wrapper statement.

```
CONNECT TO FEDDB
CREATE WRAPPER "ODBCXLS" LIBRARY 'libdb2rcodbc.a'
      OPTIONS( ADD MODULE'/openlink/lib/oplodbc.so')
```

## 5.8.6 Creating ODBC servers

You need to repeat this step for each workbook that you plan to access.

From the Create Server window (Figure 5-73), specify the server information:

On the Server page, set the following server options:

1. For the **Name** field, type REGIONSRV
2. For the **Type** field, select **ODBC**.
3. For the **Version** field, select **3.0**.

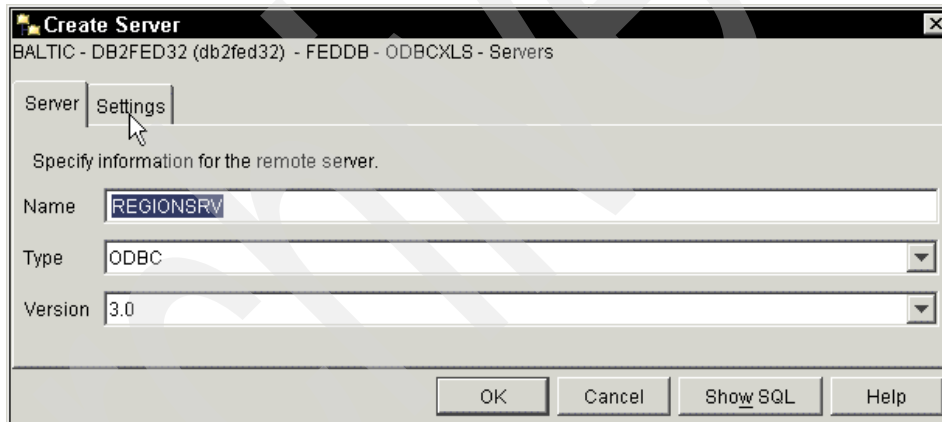


Figure 5-73 ODBC - Create server

On the Settings page (Figure 5-74) set the following server options:

1. For the **Node** option, type REGIONXLS in the Value column. This is the name of the ODBC data source you have configured with the OpenLink Admin Assistant.
2. For the **Password** option, select **N** in the Value column.
3. Click **OK** to create the server definition.

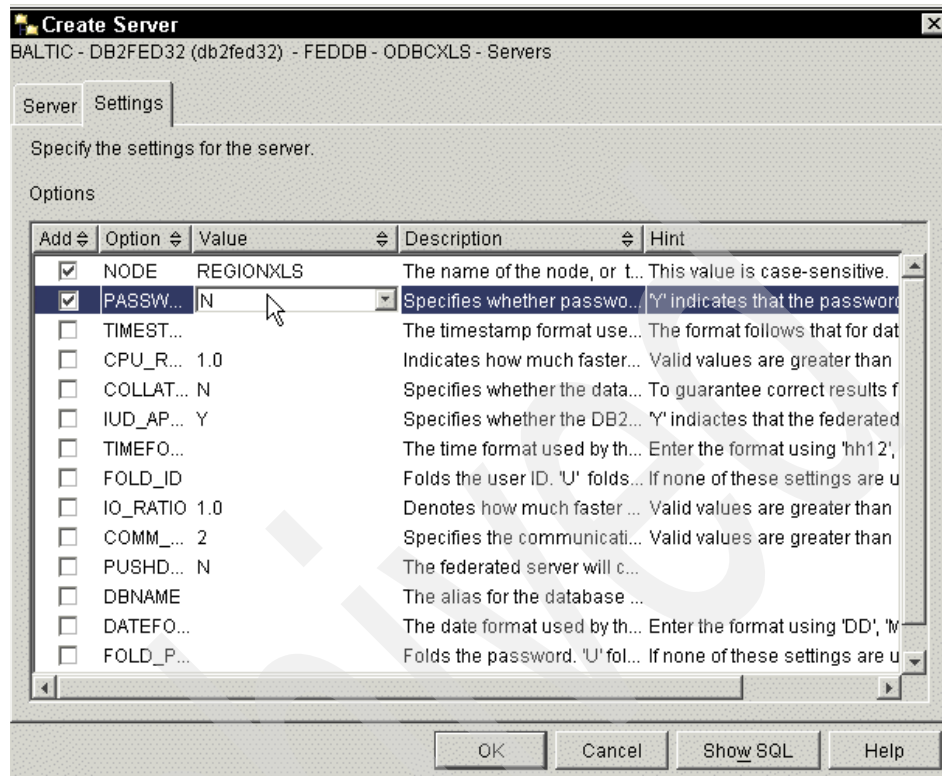


Figure 5-74 ODBC - Create server - Settings

The NODE value corresponds to the entry REGIONXLS added to the .odbc.ini file by the OpenLink client configuration. See 5.8.4 “Setting OpenLink client on AIX” on page 223.

In Example 5-27 we list the server statement.

*Example 5-27 ODBC - Create server statement*

```
CONNECT TO FEDDB
CREATE SERVER REGIONSRV TYPE ODBC VERSION '3.0' WRAPPER "ODBCXLS"
OPTIONS( ADD NODE 'REGIONXLS', PASSWORD 'N')
```

To take advantage of these capabilities, the following ODBC server options need to be set using the CREATE SERVER statement, or by using the ALTER SERVER statement:

- PUSHDOWN 'Y'
- DB2\_BASIC\_PRED 'Y'



- ▶ DB2\_ORDER\_BY 'Y'
- ▶ DB2\_GROUP\_BY 'Y'
- ▶ DB2\_COLFUNC 'Y'
- ▶ DB2\_SELECT\_DISTINCT 'Y'

## 5.8.7 Altering the ODBC server

You may modify a server setting when you want to change server options to different values.

Server options are set to values that persist over successive connections to the data source. The values are stored in the federated system catalog. Here is an example to activate (add), set, and deactivate (drop) a server option:

```
alter server ALLSRV options (add PUSHDOWN 'Y')
alter server ALLSRV options (set NODE 'NEWXLS')
alter server ALLSRV options (drop DB2_ORDER_BY)
```

To set a server options value temporarily, use the SET SERVER OPTION statement. This statement overrides the server options values in the server definition for the duration of a single connection to the federated database. The settings are not stored in the federated system catalog. An example is:

```
set server option DB2_GROUP_BY TO 'N' for server ALLSRV
```

**Tip:** The ODBC server options are not all available through the DB2 Control Center.

Additional server options for ODBC are listed in Table 5-14.

Table 5-14 ODBC additional server options

Server options	Description
DB2_GROUP_BY	GROUP BY is supported
DB2_ORDER_BY	ORDER BY is supported
DB2_BASIC_PRED	It allows '=', '<', '>' predicates
DB2_COLFUNC	It allows column functions
DB2_SELECT_DISTINCT	SELECT DISTINCT is supported

These server options with its settings override the default settings for attributes within the Information Integrator ODBC wrapper. The wrapper needs to work with any ODBC data sources, even ones with little SQL functionality; to avoid the occurrences of errors that occur from pushing down SQL operations and functions not supported by an ODBC data source, the settings for these

attributes in the wrapper are conservative. Since Excel supports some basic SQL functionality, we can set these server options to override the default settings in the ODBC wrapper; Information Integrator will then be able to pushdown simple WHERE clauses, ORDER BY, and GROUP BY clauses to Excel which should improve overall performance.

### 5.8.8 Creating the ODBC nickname

From the Create Nicknames window, click **Add** to open the Add Nickname window and set the following:

1. For the **Nickname schema** option, type XLS
2. For the **Nickname** option, type REGION. This is the Worksheet name.
3. For the **Remote table** option, type REGION\$ to read all cells.
4. Click **OK** to Add the Nickname definition.

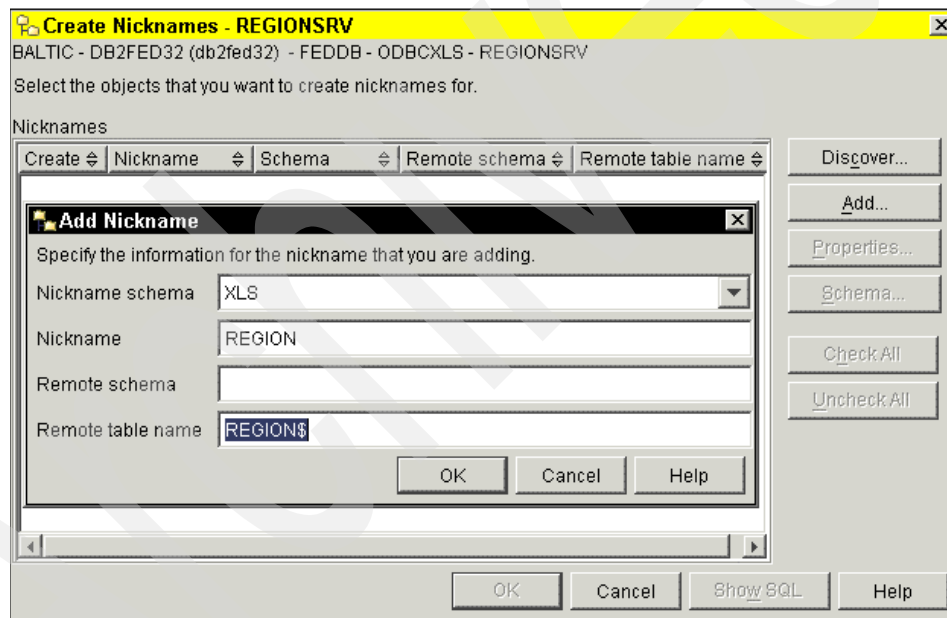


Figure 5-75 ODBC - Add nickname

5. The add function returns the nicknames based on the sheet names of the spreadsheet file. See Figure 5-76. To change the name of the nickname, highlight the nickname; you can click **Properties**.
6. Click **OK** to create the nickname.

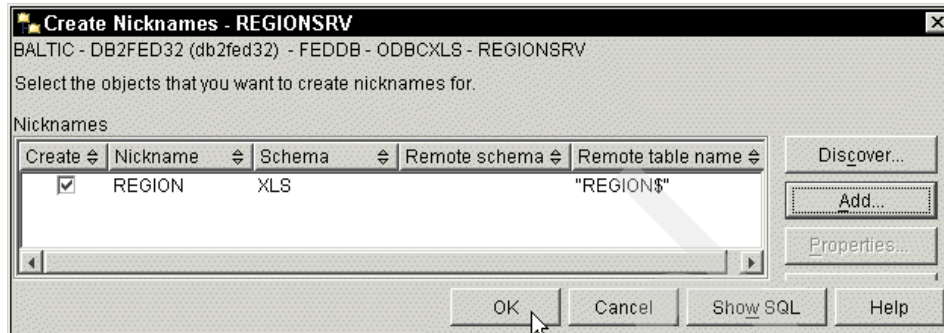


Figure 5-76 ODBC - Nickname created

7. View the sample contents of the REGION nickname in Figure 5-77.

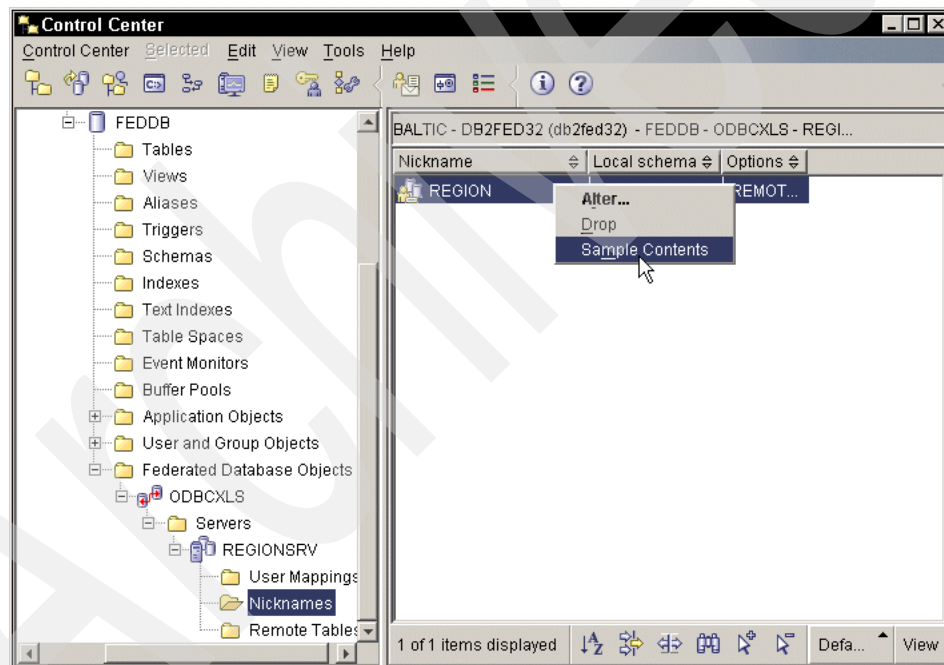


Figure 5-77 ODBC - Nickname contents

We list in Example 5-28 the nickname statement:

*Example 5-28 ODBC - Create nickname statement*

---

```
CONNECT TO FEDDB  
CREATE NICKNAME XLS.REGION FOR REGIONSRV."REGION$"
```

---

Note that in the Create Nickname statement, after the FOR keyword, only two parts are specified. We are used to seeing three parts after the FOR keyword in a Create Nickname statement. The first of the three is the Information Integrator server name. The second of the three is the remote schema, and the third is the remote object name. In the case of Excel and the ODBC interface into it, there is no schema for an Excel file or its contents. The workbook file itself must be declared as an ODBC data source, and the only objects within the file are the worksheets. There is no two-layer naming hierarchy such as in most relational DBMSs.

For this sample, we have one worksheet on the Excel file, but you can create several nicknames on the same Excel file by using different worksheets, and giving the worksheet name to create the nickname.

You must include one row of column labels in the worksheet. If you do not include one row of column labels in the worksheet, the first row of data will be treated as the column labels. Cells that contain formulas, such as SUM, will return the result of the formula and not the actual formula.

In Figure 5-78 we put on the first row the column labels and we have eight worksheets on the workbook (Excel file). With the same workbook, we can create one SERVER with eight NICKNAMES: REGION, NATION, CUSTOMER, SUPPLIER, LINEITEM, ORDERS, PART, and PARTSUPP.

	A	B	C
	N_NATIONKEY	N_NAME	N_REGIONKEY N_COMMENT
2	0	ALGERIA	0 final accounts wake quickly. special requ
3	5	ETHIOPIA	0 fluffily ruthless requests integrate fluffily. p
4	14	KENYA	0 ironic requests boost. quickly pending pint
5	15	MOROCCO	0 ideas according to the fluffily final pinto be
6	16	MOZAMBIQUE	0 ironic courts wake fluffily even, bold depos
7	1	ARGENTINA	1 idly final instructions cajole stealthily. regu
8	2	BRAZIL	1 always pending pinto beans sleep sil
9	3	CANADA	1 foxes among the bold requests
10	17	PERU	1 final, final accounts sleep slyly across the
11	24	UNITED STATES	1 blithely regular deposits serve furiously blit
12	8	INDIA	2 ironic packages should have to are slyly a
13	9	INDONESIA	2 unusual excuses are quickly requests. sly

Figure 5-78 Microsoft Excel - Workbook definition

We can see at the bottom of Figure 5-78, the workbook has eight worksheets: Region, Nation, Customer, Supplier, Lineitem, Orders, Part, and PartSupp. Excel responds to ODBC requests with an object name for each of these worksheets; the object names are REGION\$, NATION\$, CUSTOMER\$, SUPPLIER\$, LINEITEM\$, ORDERS\$, PART\$, and PARTSUPP\$ respectively.

We have created in the Information Integrator database (FEDDB) a server definition called ALLSRV that maps to this Excel file containing these eight worksheets.

We list in Example 5-29 the nickname statements for the eight tables derived from the Excel workbook:

```
CONNECT TO FEDDB
CREATE SERVER ALLRV TYPE ODBC VERSION '3.0' WRAPPER "ODBCXLS"
      OPTIONS( ADD NODE 'ALLXLS', PASSWORD 'N')
CREATE NICKNAME XLS.REGION$ FOR ALLSRV."REGION$"
CREATE NICKNAME XLS.NATION$ FOR ALLSRV."NATION$"
CREATE NICKNAME XLS.CUSTOMER$ FOR ALLSRV."CUSTOMER$"
CREATE NICKNAME XLS.SUPPLIER$ FOR ALLSRV."SUPPLIER$"
CREATE NICKNAME XLS.LINEITEM$ FOR ALLSRV."LINEITEM$"
CREATE NICKNAME XLS.ORDERS$ FOR ALLSRV."ORDERS$"
CREATE NICKNAME XLS.PART$ FOR ALLSRV."PART$"
CREATE NICKNAME XLS.PARTSUPP$ FOR ALLSRV."PARTSUPP$"
```

---

**Note:** When you create a nickname for an Excel workbook, you can rename the worksheet name, for example:

```
CREATE NICKNAME XLS.EMEA_REGION FOR ALL."REGION$"
```

### 5.8.9 Altering ODBC nicknames

Use the ALTER NICKNAME statement to modify the federated database representation of a data source. Use this statement to:

- ▶ Change local data type of a column:  

```
alter nickname XLS.REGION$ alter column r_name local type varchar (25)
```
- ▶ Change local column name:  

```
alter nickname XLS.REGION$ alter column r_name local name "R_REGIONNAME"
```

## 5.9 Maintaining

In this section we discuss maintenance and operational issues. We highlight the following topics:

- ▶ Applying FixPaks
- ▶ Updating nickname statistics
- ▶ Schema changes at data sources

### 5.9.1 Applying FixPaks

Our current environment, and the latest available version of DB2 Information Integrator right now works with DB2 ESE V8.1 FixPak 2.

As soon as DB2 V8.1 FixPak 3 becomes available, this FixPak will update all installed components of DB2 Information Integrator, but will not install any new components.

On UNIX, if you are using non-DB2 relational wrappers, like Informix, Oracle, Sybase, SQL Server, and Teradata, please pay attention to following considerations:

- ▶ The FixPak only updates the *wrapper input libraries*, so the wrappers are now at a different level from the engine.
- ▶ Run **djxlink** to create new wrappers that are at same level as the DB2 engine.
  - Stop DB2 before running **djxlink**
  - Back up the old wrapper library files, because new ones with same name will overwrite them.

**Note:** If you do not run **djxlink** after installing a DB2 FixPak to a DB2 Information Integrator installation on UNIX that includes relational wrappers, what occurs when nicknames are created or used is unpredictable, since the DB2 engine and the wrappers will be at different FixPak levels.

**djxlink** is not required (nor is it available) for DB2 Information Integrator on Windows. On Windows, the wrappers using dynamic linking to find the data source client libraries.

## 5.9.2 Updating nickname statistics

It is very important to have updated information on statistics and index information stored in a DB2 Information Integrator database catalog of nicknames. Nickname statistics and index information makes sure that the optimizer correctly picks the best plan for executing a SQL statement where a nickname is involved.

There are three ways to update nickname statistics:

- ▶ Drop/create nickname

At creation time of the nickname, statistics, and index information will be fetched from the data source. This mechanism assumes that statistics are updated at the data source, because the statistics information is just copied from the data source into the catalog information of the nickname. Basically, it depends upon the database administrator to update statistics on the data source.

- ▶ `get_stats`

Refer to 7.3 “Get Statistics utility: `get_stats`” on page 289.

- ▶ “Manually” execute SQL.

The idea of this manual approach is to derive the statistics of the data source using the count function on the nickname or directly on the data source, and update `SYSSTAT.TABLES` and `SYSSTAT.COLUMNS` with this information.

Additionally, you may need to add index records for nicknames using the **Create Index ... specification only** statement.

If you are required to create nicknames for views at data sources, instead of for tables, then you will have to use `get_stats` or the manual technique to put nickname statistics into the Information Integrator catalog. Also, since views do not have index information at the data sources, you will also have to use the `CREATE INDEX...SPECIFICATION ONLY` statement to add information to the Information Integrator catalog about which columns of the remote view are in indexes at the data source.

Also, creating nicknames with the non-relational wrappers (XML, table structure flat file) will not get nickname statistics. The statistics for these nicknames must be added manually, or using `get_stats`.

It is suggested that if you will create nicknames for views or for non-relational data sources, that you create scripts to create the nicknames, and include in the script's statements to add the index information and statistics to the nickname.

### 5.9.3 Schema changes at data sources

If a table or a view is changed (i.e. a column is added), where a nickname is referring to, you basically have to drop and recreate the nickname.

The recreation of this nickname causes DB2 Information Integrator to re-read the catalog at the data source to learn about the columns of the remote table.

### 5.9.4 Nicknames used in views and packages

Dropping a nickname invalidates any views or packages that reference the nickname.

If the nickname is created again, such as to get new statistics, or to add a column for a column added to a table at a data source:

- ▶ Re-create any views that use the nickname, so that the view will be valid again.
- ▶ Re-bind any packages that contain static SQL that references the nickname.



## 5.10 Troubleshooting

This section provides you with suggestions and solution approaches for frequently asked questions and common problems around DB2 Information Integrator. Here we list several information sources for solving problems, give hints on resolving errors when linking with data source clients (executing **djxlink**), and describe solutions for errors on defining and using federated objects.

### 5.10.1 Errors linking with data source clients

Throughout the whole troubleshooting section, we work with an example of Oracle data sources. However, **djxlink** is a general concept for linking all data sources with DB2. What you read here can also be applied to any data source other than Oracle.

#### What is djxlink doing?

The shell script **djxlink** creates the wrapper libraries, which are linked with the data source client libraries, the operating system libraries, and DB2 libraries. Furthermore, it searches through libraries on the system for symbols (APIs) that the wrapper needs to be able to find at runtime.

Places to search:

1. /usr/lib
2. DB2 directories
3. Subdirectories of the data source client software - defined by i.e. ORACLE\_HOME, INFORMIXDIR, etc.

#### Who needs to execute it?

Root

#### When does it need to be executed?

- ▶ The script is executed automatically at the end of DB2 Information Integrator installation.
- ▶ After you install client software for a new data source, the script for that specific data source is being executed automatically. This is if you additionally add a data source after you installed DB2 Information Integrator.
- ▶ After applying DB2 Information Integrator FixPaks

#### Which input does it need?

- ▶ /usr/opt/db2\_08\_01/djxlink\* (that is **djxlink0racle**)

- ▶ Wrapper input library - libdb2ST\* (i.e. /usr/opt/db2\_08\_01/lib/libdb2STnet8U.a, /usr/opt/db2\_08\_01/lib64/libdb2STnet8F.a)
- ▶ Data source client software
- ▶ Environment variables indicating the location of data source client software

### Which output is produced?

- ▶ djsxlink\*.out file in /usr/opt/db2\_08\_01/lib is produced (i.e. djsxlinkOracle.out). This file contains detailed information, warning, and error messages.
- ▶ Wrapper libraries - libdb2\*.a in /usr/opt/db2\_08\_01/lib
- ▶ and in the DB2 instance owner ~/sqllib/lib directories

### Heart of a djsxlink Oracle script

```
linknet8 () {
target=libdb2net8F.a # wrapper library being built
/bin/rm -f $target
/bin/ld -o $target \
    -eFencedWrapper_Hook\
    -bE:djsxlinkOracle.2 -bI:djsxlinkOracle.3 -bM:SRE\
    -M -K -lc -lld -lc_r -lpthreads -lodm /lib/crt0_r.o \
    -lC -lm -lc \
    -M -K -lc -lld -lc_r -lpthreads -lodm /lib/crt0_r.o \
    -lC -lm -lc \
    -L$ORACLE_HOME/lib -L$ORACLE_HOME/rdbms/lib \
    $ORACLE_HOME/rdbms/lib/defopt.o $ssdbaed_obj \
    $nautab_obj \
    $ORACLE_HOME/rdbms/lib/xaondy.o \
    -lc1ntsh \ # Main Oracle client library libclntsh.a
    libdb2STnet8F.a # Wrapper input library - supplied by II
1>>djsxlinkOracle.out 2>&1
}
```

#### Explanations:

**\$ORACLE\_HOME:** When you run **djsxlinkOracle** manually, you need to export this environment variable first. When doing DB2 Information Integrator Relational Connect Oracle wrapper install, ORACLE\_HOME will be set automatically:

- ▶ Input libraries:
  - They are mostly of the type lib\*.a
  - -l ... in a link actually identifies the file lib\*.a
  - Key Oracle input library is libclntsh.a (-lc1ntsh in the **djsxlinkOracle**), supplied by the Oracle Client.

- Wrapper input library for Oracle is libdb2STnet8F.a, supplied by DB2 Information Integrator
- ▶ **djxlink** error and warning messages:
  - Script writes to: /usr/opt/db2\_08\_01/lib/djxlink\*.out.
  - Warning message about the duplicate symbol is OK.
  - Warning message about the entry point not found is OK.

### **djxlink usual problem causes**

- ▶ Environment variable does not point to right location in the file system (i.e. ORACLE\_HOME).
- ▶ The wrong data source client software is installed, so it does not include the right input libraries.
- ▶ Changing names of libraries in different releases of the client software. Look out for messages like -lct.a not found, which means that the file libct.a cannot be found.
- ▶ Changing locations of symbols in APIs of client libraries

## **5.10.2 Errors when defining and using federated objects**

### **SQL1822N**

DB2 Information Integrator attempts to translate data source error codes into DB2 equivalents. Nevertheless, the data source error code is always available in the log file db2diag.log.

The error code SQL1822N stands for an error code on the data source for which DB2 Information Integrator does not have any translation to an equivalent DB2 UDB SQL code.

The first token in the message contains the error code of the vendor.

The additional tokens/texts may contain additional error descriptions of the vendor.

### **db2diag.log**

This log file is used by most wrappers to provide basic error information in the format as shown in Example 5-30.

*Example 5-30 Output of db2diag.log*

---

```
2003-07-17-13.27.45.093829 Instance:db2fed32 Node:000
PID:29336(db2sysc) TID:1 Appid:none
drda wrapper getinfo Probe:5 Database:FEDDB
```

ODBC error txt:	
0x2FF159FC : 5B49 424D 5D5B 434C 4920 4472 6976 6572	[IBM][CLI Driver
0x2FF15A0C : 5D20 5351 4C31 3031 334E 2020 5468 6520	] SQL1013N The
0x2FF15A1C : 6461 7461 6261 7365 2061 6C69 6173 206E	database alias n
0x2FF15A2C : 616D 6520 6F72 2064 6174 6162 6173 6520	ame or database
0x2FF15A3C : 6E61 6D65 2022 4954 534F 2220 636F 756C	name "ITS0" coul
0x2FF15A4C : 6420 6E6F 7420 6265 2066 6F75 6E64 2E20	d not be found.
0x2FF15A5C : 2053 514C 5354 4154 453D 3432 3730 350A	SQLSTATE=42705.

---

### 5.10.3 Information to gather

Information about wrappers, servers, and nicknames can be found in following system catalog tables:

SYSCAT.WRAPPERS  
SYSCAT.WRAPOPTIONS

SYSCAT.SERVERS  
SYSCAT.SERVEROPTIONS

SYSCAT.USEROPTIONS

SYSCAT.TABLES  
SYSCAT.TABOPTIONS  
SYSCAT.COLUMNS  
SYSCAT.COLOPTIONS  
SYSCAT.INDEXES  
SYSCAT.INDEXOPTIONS  
SYSCAT.KEYCOLUSE

#### Instance information

This information can be obtained with following tools and commands:

- ▶ **db2look**: Enhanced to extract information on federated objects
- ▶ **db2dj.ini** variables and setting:  
UNIX: [db2 instance owner]~/sqllib/cfg/db2dj.ini  
Windows: c:\Program Files\IBM\sqllib\cfg\db2dj.ini
- ▶ db2level
- ▶ db2set <variables>: db2set -all
- ▶ System environment variables: UNIX: **env**; Windows: **set**
- ▶ Database Manager Configuration: **db2 get dbm cfg**
- ▶ Database Configuration: **db2 get db cfg for [databasename]**



## Part 3

# Performance concepts with DB2 Information Integrator

In this part we provide a general tutorial on the performance concepts that are the basis for understanding how your queries are performing in a data federation environment. The chapters are:

- ▶ A performance tutorial
- ▶ Our performance tools



## A performance tutorial

In this chapter we describe how to start evaluating the performance of queries accessing a DB2 Federated Data system.

We look at:

- ▶ Federated query performance
- ▶ Tuning a query on a single data source
- ▶ Tuning a query on multiple data sources
- ▶ Performance and availability with MQTs
- ▶ Federated query performance checklist

## 6.1 Federated query performance

Probably the most significant inhibitor to the viability of federated technology is the issue of reasonable performance. IBM invests heavily in query optimization research and development. The DB2 Information Integrator optimizer takes into account standard statistics from source data (for example, cardinality or indexes), data server capability (such as join features or built-in functions), data server capacity, I/O capacity, and network speed. The query rewrite logic rewrites queries for more efficient processing. For example, it can convert a join of unions, which drives a tremendous amount of data traffic into a union of joins, which leverages query power at the data server and minimizes data traffic back to the federated server.

The pushdown analysis capability identifies which operations can be executed at the data server prior to returning results to the federated server. The optimizer can perform a nested loop join that queries a small table on one server and uses the results as query predicates to a large table on another. And the administrator can define materialized query tables (MQTs), which the optimizer can transparently leverage to satisfy user queries.

This section provides the basic concepts and functions of how a query is handled within the whole scheme of federated data and information integration, and then focuses on performance factors for federated queries. The intent is to provide enough information on query handling to understand the examples that will follow later in the book.

### 6.1.1 Performance factors

The first and obvious factors include the processing power of the local and the remote machines, as well as the bandwidth of the communication network between. But probably the most impacting factors can be the quality of the query execution plan at the federated server and on the remote sources.

The plans influence the number of interactions required between the federated server and the remote sources, and the amount of data that is moved. Data movement between the federated server and the remote source is a key factor.

The amount of data being moved depends mainly on two factors:

- The amount of processing and filtering that can be pushed down to the remote data sources

If there are some filtering predicates in the where-clause, and the remote source is able to apply those predicates, then the federated server pushes down these predicates to the remote server to reduce the amount of data that needs to be shipped back.



- The data placement among multiple sources

If you join two tables and they are both on the same data source so that the join can be done at that data source without moving the tables out, then usually that results in better performance than if the two tables reside at two different sources.

In a join between tables that are not co-located, data from both tables must be moved to the federated server, which will then do the join. Note that the federated server never moves data between remote data sources; only between them and itself. DB2 Information Integrator has some efficient techniques for performing this data movement. One is a nested loop join in which the results of SQL sent to one data source are supplied as values for host variables sent in SQL to the second data source. The other is to use hash-joins to obtain a join result from two data sources.

### 6.1.2 The pushdown concept

Pushdown is an important aspect of federated query processing. If the PUSHDOWN server option is set to Y, the optimizer will consider generating a plan that “pushes down” certain parts of the query execution to the remote source. A component within the optimizer at the federated server called “pushdown analysis” decides which parts of a query can be pushed down to the data sources and processed remotely at the data sources. The decision on which parts and operators of a query to pushdown depends on several factors:

- The availability of required functionality at the remote source

If the remote source is simply a file system with a flat file, then it is probably not possible to pushdown any filtering predicates.

- The options specified in the server definition of a remote source

For instance, if the collating sequence at the federated server is different from the one at the remote server, then operations on string data like sorting, and some predicates involved in the query have to happen at the federated server, and cannot be pushed down.

- Few other involved issues that influence the pushdown ability

You can specify that a remote column always contains numeric strings so that differences in the collating sequence do not matter. Sometimes remote source functionality is dependent on data type issues; for example, a remote function may accept only certain argument types.

- Attributes within the Information Integrator wrappers that indicate which operations and functions are supported by the type and version of the data source

- Function mappings in the Information Integrator catalog. The function mappings in the catalog are created by the Information Integrator administrator, and are additions and overrides to the default function mappings that are in the wrappers.

Figure 6-1 gives a simple example of a query that cannot be pushed down as it is to the remote data source. In this case we assume that the data source is not relational and is unable to perform a COUNT(\*). Even for non-relational sources, the optimizer works on trying to make sure that no more than one column is retrieved to evaluate the COUNT function. It changes the SELECT \* FROM ... into a SELECT .. FROM ... , adjusting the select list to retrieve just enough data to evaluate the COUNT locally (SELECT 1 in our example).

An important thing is that the optimizer's decision to pushdown or not to pushdown an operation is cost-based. The decision is for example, influenced by information about the CPU and I/O speed of remote sources as well as by information about the communication bandwidth. So, if an operation can potentially be pushed down then that does not necessarily mean that it will be pushed down to the remote source. It depends on the costing done by the optimizer, and it is also heavily influenced by the estimated number of rows processed and returned from the remote sources.

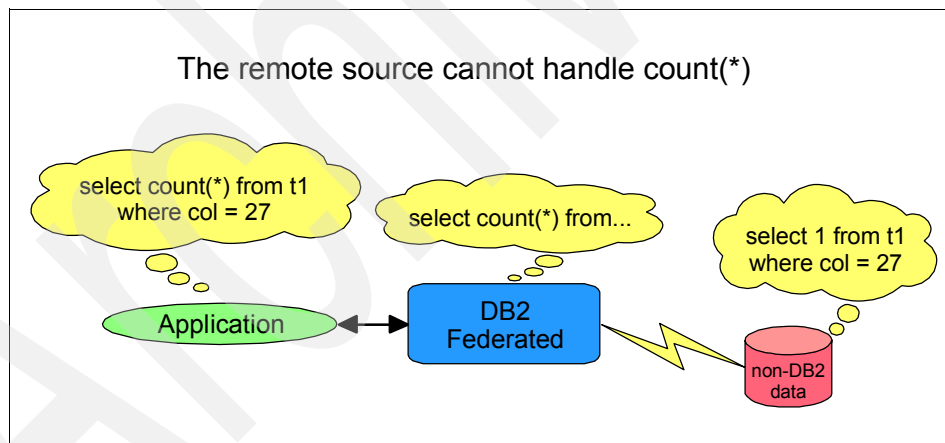


Figure 6-1 Pushdown example 1

Figure 6-2 shows a simple example. We have a single remote source which contains two tables, and we have defined two nicknames ORA.T1 and ORA.T2 for the remote tables. Table 1 has 25 rows and Table 2 has 10,000 rows.

We submitted a join to the federated server between the two nicknames, which returns nearly the full Cartesian product of the two tables.

That join can be pushed down to the remote source where it would produce almost  $25 * 10,000$  rows, so almost 250,000 rows then need to be shipped back to the federated server. But in this case it would actually be better not to pushdown the join, and to simply ship the rows from the two tables to the federated server, and then do the join locally. This way only 10,025 rows would be moved instead of 250,000!

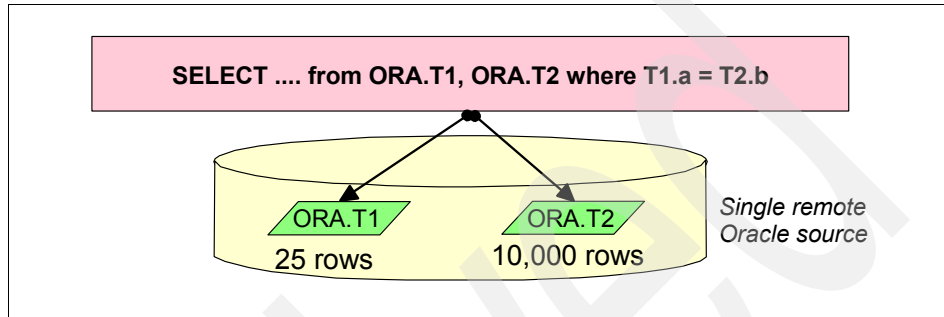


Figure 6-2 Pushdown example 2

Since data movement is usually the bottleneck, the optimizer tends to pushdown operations in such a way that the overall amount of data being moved is minimized.

In general, depending on the optimizer's decisions, the access plans might call for the query to be:

- ▶ Processed by the data sources
- ▶ Processed by the federated server
- ▶ Processed partly by the data sources, and partly by the federated server

The compiler develops alternative strategies, called *access plans*, for processing the query. DB2 evaluates the access plans primarily on the basis of information about the data source capabilities and the data. The wrapper and the global catalog contain this information. DB2 decomposes the query into segments that are called query *fragments*. Typically, it is more efficient to *pushdown* a query fragment to a data source, if the data source can process the fragment. However, the query optimizer takes into account other factors such as:

- ▶ The amount of data that needs to be processed
- ▶ The processing speed of the data source
- ▶ The amount of data the fragment will return
- ▶ The communication bandwidth

The query optimizer generates local and remote access plans for processing a query fragment, based on resource cost. DB2 then chooses the plan it believes will process the query with the least resource cost.

If any of the fragments are to be processed by data sources, DB2 submits these fragments to the data sources. After the data sources have processed the fragments, the results are retrieved and returned to DB2. If DB2 performed any part of the processing, it combines its results with the results retrieved from the data source. DB2 then returns all results to the client.

## Compensation

The DB2 Federated Server does not pushdown a query fragment if the data source cannot process it, or if the federated server can process it faster than the data source can process it. For example, suppose that the SQL dialect of a data source does not support an *outer join*. A query that contains an outer join and references a table in that data source is submitted to the federated server. DB2 does not pushdown the outer join to the data source, but processes it itself. The ability by DB2 to process SQL that is not supported by a data source is called *compensation*.

The federated server compensates for lack of functionality at the data source in two ways:

- ▶ It can ask the data source to use one or more operations that are equivalent to the DB2 function stated in the query. Suppose a data source does not support the cotangent (COT(x)) function, but supports the tangent (TAN(x)) function. DB2 can ask the data source to perform the calculation  $(1/\text{TAN}(x))$ , which is equivalent to the cotangent (COT(x)) function.
- ▶ It can return the set of data to the federated server, and performs the function locally.

With compensation, the federated server can support the full DB2 SQL dialect for queries against data sources. Even data sources with weak SQL support, or no SQL support, will benefit from compensation. You must use the DB2 SQL dialect with a federated system, except in a passthru session.

### 6.1.3 Optimizing a federated query

To obtain data from remote data sources, submit queries in DB2 SQL to the federated server. The DB2 query compiler then consults information in the DB2 federated database system catalog and the data source wrapper modules to process the query. This includes information about establishing the connection to the data source, server attributes, mappings, index information, and processing statistics.

Figure 6-3 shows the query compiler flow for federated queries.

First, the SQL text is parsed and checked semantically. View definitions are expanded into the main statement block, which might result in a more complex statement. This view expansion sometimes creates opportunity for better optimization.

Next, the query goes through the rewrite phase. Rewrite is an optimization strategy that transforms a valid query into a semantically equivalent form that can be more efficient to execute. The rewrite phase is particularly important for queries that are very complex, such as queries with many subqueries or many joins.

The two major categories of rewrites that are performed by the query compiler are operation merging and predicate manipulation. View merging and subquery to join transformation are examples of operation merging. Using views in a SELECT statement can restrict the join order of the tables, and limit the query optimizer choices when considering access plans. By merging the views, such restrictions can be lifted. Similarly, subqueries impose a certain order of join execution. This is alleviated by aggressively rewriting subqueries to joins. In more complex queries, especially in the presence of views, redundant joins can easily be introduced. Rewrite detects these and simplifies the statement. On the other hand, rewrite might introduce implied predicates through transitivity. As a result, the optimizer can consider additional joins when it is trying to choose the best access plan for the query.

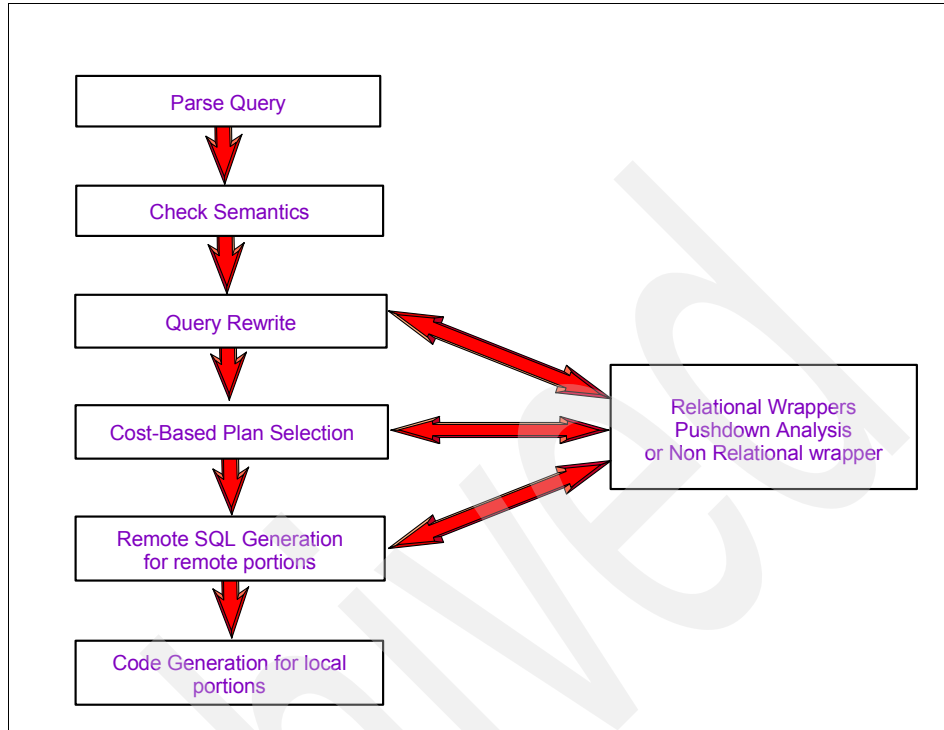


Figure 6-3 Query compiler flow

In a federated environment, the portion of a query that references relational nicknames goes through the pushdown analysis component. Pushdown analysis tells the query optimizer if an operation can be performed at a remote data source. An operation can be a relational operator, system, or user-defined function, or an SQL operator such as GROUP BY, ORDER BY, and so on. Functions and operations that cannot be pushed down can adversely impact query performance. Consider the effect of a selective predicate being evaluated at the federated server instead of the remote data source. This approach requires the federated system to retrieve the entire table from the remote data source, and then filters it locally against the predicate. Depending on the size of the table and network traffic, the query performance could suffer.

Pushdown analysis uses internal server attributes to decide if an operation can be performed remotely. Internal server options model a data source to the federated compiler. The federated server allows the use of a rich SQL dialect to submit queries. This dialect might offer more functionality than the SQL dialect supported by a remote data source. The federated server can compensate for the lack of function at the data server, but doing so might require the operation to take place at the federated server. Internal server attributes are key factors in

determining the differences in functionality that can be supported at the federated system versus remote data sources. They are categorized based on SQL language capabilities, SQL limits, and various other server specific restrictions.

Non relational data sources are modelled differently from relational data sources. The non relational data source characteristics are encapsulated within the wrapper. The pushdown decision is made by the wrapper during the planning phase while working with the query optimizer. Hence, the non relational portions of the query do not go through the pushdown analysis phase.

In general the optimizer has several options to pushdown or not to pushdown certain parts of the query, and based on those options, it will generate potential plans, it will do costing for these plans, and then chooses the one with the lowest cost.

After that, remote SQL is generated and shipped to the remote sources, and code is generated for executing the local fragment of the query.

Note that federated server issues queries to the remote sources just as any other client does. When the query arrives at the remote source, it is a regular dynamic statement that is optimized and executed right there as the remote source sees fit. This is even true if the remote source is DB2.

#### 6.1.4 Pushdown analysis and cost optimization

As previously stated, pushdown analysis (PDA) is invoked as part of query execution if a query includes nicknames. PDA interacts with several phases as indicated in Figure 6-3, but it is best to understand its place as being DB2 Query Rewrite and Cost Optimization. PDA tells the optimizer which of the SQL operations and functions are allowed to be pushed down. When the optimizer generates alternative access plans for which it will then estimate the total cost of each:

- ▶ If PDA indicated that an operation is allowed to be pushed down, the optimizer can generate alternative access plans that perform the operation/function at the data source (pushed down) and locally at the Information Integrator server (not pushed down).
- ▶ If PDA indicated that an operation is not allowed to be pushed down, then the optimizer can generate alternative access plans that perform the operation only locally at the Information Integrator server.

This is an important concept to understand when trying to find ways to improve the performance of federated queries. You may change inputs to the optimizer, such as nickname statistics and server options CPU\_RATIO and IO\_RATIO and

see different access plans in explain outputs, but see no access plan that pushes down a particular operation to the data source. All the access plans you see in the different explanations were alternative access plans generated by the optimizer; changing the nickname statistics and CPU\_RATIO/IO\_RATIO caused different plans to have the lowest estimated cost, and this is why the different access plans show up as the chosen plan in the different explains. The operation that is not pushed down in any of the access plans is one that pushdown analysis restricted from being pushed down, and so it was not pushed down in any of the access plans created and evaluated by the optimizer.

The server option DB2\_MAXIMAL\_PUSHDOWN can play a useful role in determining if it is cost optimization or pushdown analysis that is causing an operation not to be pushed down. Explain a query with the default setting for DB2\_MAXIMAL\_PUSHDOWN and the optimizer will pick the access plan with the lowest cost based on the current inputs such as nickname statistics, index information, and CPU\_RATIO/IO\_RATIO. Then add the server option DB2\_MAXIMAL\_PUSHDOWN with the setting Y, which will cause the optimizer to generate the same access plans, but to select the one that pushes down the most operations to the data source.

If the access plan in the second explanation output does not show an operation being pushed down, then it was PDA that was restricting the operation from being pushed down. If it was PDA restricting the operation from being pushed down, then explore the inputs to PDA (such as server version, server/column options COLLATING\_SEQUENCE/NUMERIC\_STRING, and the operations in the user's SQL) to get more push down if this is possible. If PDA is not restricting the operation from being pushed down, then explore the inputs to the optimizer, such as nickname statistics and index information, and the server options CPU\_RATIO, IO\_RATIO and COMM\_RATE.

### 6.1.5 Importance of pushdown

The amount of SQL pushed down to the data sources is important for two reasons:

- ▶ If less SQL is pushed down, this usually means that a larger result set must transit from the data source to the Information Integrator server, so that Information Integrator can apply the remaining SQL to further reduce the size to getting the final result for the end user. If more of the SQL is pushed down, that usually means that a smaller result set has to transit from the data source to the Information Integrator server. A smaller set means that the amount of time taken for results to transit the network is less, and the total execution time for the query should be less as well.
- ▶ If less SQL is pushed down, and Information Integrator has to receive results from data sources into temporary tables to complete the processing of the



query, then Information Integrator uses the memory buffers allocated for SORTHEAP, and the buffer pool for the temporary table space (TEMPSPACE1) to hold the first rows of these temporary tables. As the result set from the data source grows, memory allocated and available for SORTHEAP and TEMPSPACE1 may not be enough to hold the growing temporary table, and Information Integrator has to perform disk I/O to write pages of this temporary table out to the file containers of TEMPSPACE1, so that it can receive the rest of the result from the data source, and complete the temporary table. And, after all the rows have been inserted into the temporary table, the temporary table must be scanned, meaning more disk I/O to process the SQL that was not pushed down. So you can see, if more SQL is pushed down, there either are no temporary tables created, or the temporary tables are small enough to fit in memory at the Information Integrator server, so that no disk I/O is needed to process.

### 6.1.6 Interpreting federated query execution plans

The query optimizer in the federated system uses information stored in the DB2 global, together with the system configuration information and the query requirements to generate an optimal plan based on the cost estimate. The system configuration information can include the size of the buffer pool and the sort heap. Query requirements can include the current class of optimization techniques enabled for the query, and whether the query should be optimized for the best shortest time to return the first row instead of the entire result set. The output of the query optimizer on a federated system is the federated query execution plan. The plan shows how the query is being executed. There are a number of ways of viewing a plan. These include: Visual Explain, Explain and db2expln, etc. See Chapter 7, “Our performance tools” on page 283 for details.

The query optimizer adds two new operators for the federated plan:

- ▶ A SHIP operator, which is always present for a SELECT statement and sometimes present in the INSERT, UPDATE or DELETE queries involving a relational nickname.
- ▶ A remote pushdown (RPD) operator for non relational nicknames, used to encapsulate the access logic in the federated plan.

All the operators in the federated plan have an extra property called the server property. This property denotes the name of the server or the remote data source where that operation is performed.

For relational data, the data transfer from a remote data source to the federated server or vice versa occurs in a SHIP operator. The federated server communicates a remote request for data to the remote relational data source through an SQL statement in the native SQL dialect of the data source. For

SELECT queries, the remote SQL statement can be found inside the SHIP operator. For INSERT, UPDATE, DELETE queries involving nicknames all the work might be done by the remote data source, and a SHIP operator will not be found in the plan. In this case, the RETURN operator contains the remote SQL statement that is passed to the remote data source.

If a query references a non relational nickname, the portion of the query involving this nickname does not go through the pushdown analysis phase like the portion involving relational nicknames. There is communication between the query optimizer and the non relational wrapper during plan generation for the query. The access to the non relational nickname is modeled using the remote pushdown operator. The query optimizer, in consultation with the wrapper, determines how the data is to be accessed. The main difference between remote pushdown and SHIP is that the remote pushdown operator does not contain an SQL statement. The remote access plan contained in the remote pushdown operator is constructed by the wrapper and contains the necessary specification to invoke the remote request from the non relational data source.

Operations found in the access plan above SHIP and RPD operations are performed at the Information Integrator server. Operations performed at the relational data sources (or by the non relational wrappers) are found in the details for the SHIP and RPD operations.

For instance, if you see two SHIP operators to the same data source and a nickname under each, and a NLJOIN, MSJOIN, or HSHOIN operator above the two SHIP operators, that means the join of these two nicknames is being done at the Information Integrator server. If there is one SHIP operator with two nicknames beneath it, that means the join is being pushed down to the data source and can be found in the RMQTX of the SHIP operator.

Likewise, if the user's SQL includes an ORDER BY clause and you see a SORT above the SHIP operations of the access plan, and examine the RMQTX of the SHIP operators and do not see the ORDER BY clause there, that means the ORDER BY is being executed at the Information Integrator server.

When reviewing the access plan, it is usually more useful to understand the number of rows that will be involved in an operation than to focus on the incremental cost associated with the operation. The number of rows that the optimizer estimates will be involved in an operation appears above the operator in the access plan graph in **db2exfmt** outputs; in access plans created in DB2 Command Center (that is Visual Explain) the "cardinality" for an operation is in the details for the operation; the number displayed in the cell in the access plan is the cumulative cost estimate. The number of rows involved in an operation indicates things like:

- ▶ For SHIP operations, the number of rows that will cross the network from the data source to the Information Integrator server
- ▶ For SORT and TEMP operations, which signify temporary tables, the number of rows that will fill this temporary table. The estimated number of rows can be multiplied by the row-width (which will be in the details for the operation) to calculate estimated size and number of 4 KB buffers required. It can be estimated if the temporary table will fit in the available memory buffers for temporary tables at the Information Integrator server, or whether it will require disk I/O to the file containers of the temporary table space.

## 6.2 Tuning a query on a single data source

In this section we provide some examples on how to analyze federated queries against a single remote data source, and how to help the DB2 optimizer to choose the best performing access path.

### 6.2.1 What is a good query?

How can we tell if a query against a single remote data source has good performance or not?

One approach is that we execute the query using DB2 Federated Server to access the data source, and compare that execution against executing the same query using native access to the remote source.

For example, if the remote source is Oracle, the native interface might be a Net8 Oracle Client.

Compare the runtime of the query submitted through the Net8 client directly against the Oracle tables with the runtime of the same query submitted to the federated server. The federated query of course refers to local nicknames to the Oracle tables.

If the two execution times are close, then the federated query is performing well.

This comparison was done on a set of complex queries against a single remote data source represented by an industry standard data warehouse implemented on a non IBM relational database system. The queries were submitted to the DB2 Federated Server with nicknames to the tables residing in the remote database. Then the same queries were submitted directly to the local database using its native interface.

The comparison of the total elapsed times of these two alternatives is reported in Figure 6-4.

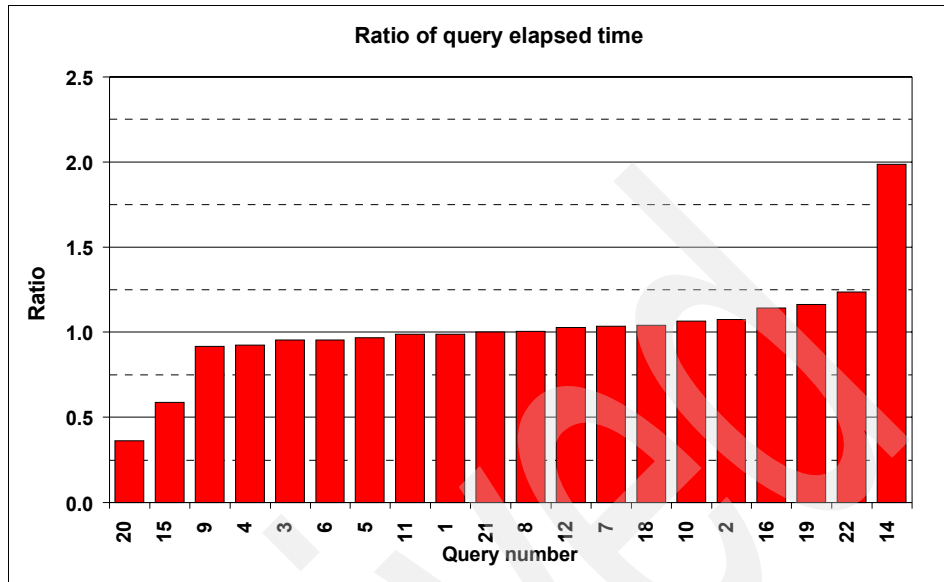


Figure 6-4 Federated vs. native - Single query

The chart shows the ratio of the elapsed times (which runs from 10 seconds and 3 minutes) using DB2 Information Integrator divided by the elapsed times using the native SQL access directly to the other database.

For most queries, we see that the ratio between federated access and native access is very close to one. This means that in many cases, accessing a remote data source through DB2 Information Integrator delivers about the same performance as using native access to the data source. For these queries, DB2 Information Integrator introduces a proportionally small overhead.

However, you can also see that there are exceptions. For example, query 14 took twice as long using DB2 federated as through the native interface. While query 20 performed much better through DB2 Federated Server than through the native access of the remote source. In order to explain that and learn in the process, we need to look at a few of these cases in more detail. In 6.2.2, “Evaluating the execution plan with Explain” on page 254, we look first at query 4, which is a very typical case where performance is about the same in both cases, and then we will look at the two other extreme situations: query 14 and query 20.

## 6.2.2 Evaluating the execution plan with Explain

The first step in investigating the performance of a federated query is to obtain the execution plan. A good way to get execution plans is to use the tool *DB2*

*explain format (DB2EXFMT)*. This tool requires to have created the Explain tables. You need to create the Explain tables only once for your database instance by going to the /sqlib/misc directory and finding the EXPLAIN.DDL file. See 7.1.2, “DB2 Explain facilities” on page 285 for activating and using Explain.

After that, you can type:

```
explain plan for <query>
```

where <query> is the SQL statement to be explained. This will populate the Explain tables with information about the query, and then you can use:

```
db2exfmt -d <dbname> -1 -o <output file>
```

to format and output this information from the Explain tables. You specify the database name, you specify -1 to indicate that you want to format the plan for the latest query explained, and you can specify an output file.

The formatted execution plans for federated queries looks just like plans for regular DB2 queries, with the exception of new SHIP and RPD operators, which indicates interaction with the remote sources. Note that in DB2 federated V7, the SHIP operator was replaced by the RQUERY operator.

#### **Query 4**

As an example, let us look at the execution plan for query 4, a well behaved query, reported in Figure 6-5.

This plan has four nodes: the ship operators, two place-holders for the two remote tables which are being accessed, and the RETURN node which also serves as a legend for the information which is displayed at each operator.

You see the name of the operator, above the name is the estimated number of rows produced by that operator, and below the name are the number of the operator as well as two values representing cost and I/O.

## Query 4 explain plan

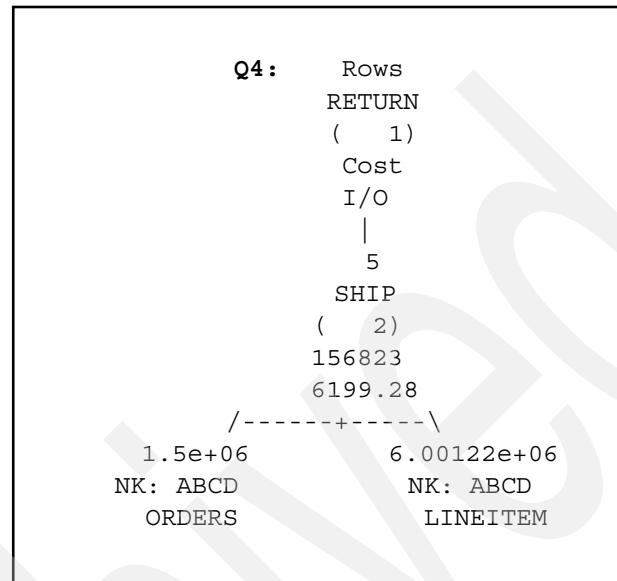


Figure 6-5 Query 4 Explain plan

Most of the time the estimated number of rows is the key thing to look at. For example, the ship operator produces 5 rows which are returned to the user or application. You see that above the ship operator there is no further processing of these 5 rows, so it is likely that no processing going on at the federated server and the entire query has been pushed down to the remote source. However, it is possible that there are functions on the select list that DB2 still chooses to process locally, so the best way to verify whether the entire query has been pushed down is to check the remote SQL statement to see if it indeed includes the original query.

The optimizer estimates that 5 rows will be returned from the SHIP. The optimizer makes this calculation using information in the Information Integrator catalog; the most important inputs to this calculation are the number of rows (SYSSTAT.TABLES.CARD) for the nicknames involved, any unique index records for these nicknames, and (if available) column value distribution statistics (SYSSTAT.COLUMNS.COLCARD). The cardinality of the nicknames is shown in the access plan; it is 1.5e+06 (1,500,000) for ABCD.ORDERS and 6.00122e+06 (6,001,2200) for ABCD.LINEITEM. These are fundamental building blocks to the optimizer's estimate that the result of the RMQTX of the SHIP operator will be only 5 rows.

Everything underneath the ship operator is pushed down to the remote source, and we cannot see the execution plan of that part which is pushed down. We only see that it references two remote tables which have nicknames ABCD.ORDER and ABCD.LINEITEM, and ORDER is expected to have 1.5 M rows and LINEITEM is expected to have 6 M rows. However, in the details of the db2explain format output we can see the actual SQL which is shipped to the remote source, see Figure 6-6.

Here on the top you see the original query text of query 4, and on the bottom you see the SQL which is actually being shipped to the remote source. You do not really need to look at all the details of these two SQL statements, just note that they are different. For example the original query used nicknames, but in the SQL shipped to the remote source the nicknames are replaced by the actual remote table names. Also, the original expression of *1st of July 1993 plus 3 months* is resolved to *1st of October 1993*, etc.

<p><b><u>Original query</u></b></p> <pre> SELECT O_ORDERPRIORITY, COUNT(*)       AS ORDER_COUNT FROM   ABCD.ORDER WHERE  O_ORDERDATE &gt;= TIMESTAMP('1993-07-01-00.00.00')       AND O_ORDERDATE &lt;  TIMESTAMP('1993-07-01-00.00.00') + 3 MONTHS AND EXISTS (SELECT * FROM ABCD.LINEITEM             WHERE L_ORDERKEY = O_ORDERKEY             AND L_COMMITDATE&lt;L_RECEIPTDATE ) GROUP BY O_ORDERPRIORITY ORDER BY O_ORDERPRIORITY ; </pre> <p><b><u>What is shipped to the remote source</u></b></p> <pre> SELECT A0."O_ORDERPRIORITY", COUNT(*) FROM   "ABCH"."ORDER" A0 WHERE  (TO_TIMESTAMP('19930701 000000000000','YYYYMMDD HH24MISSFF') &lt;= A0."O_ORDERDATE")       AND (A0."O_ORDERDATE" &lt; TO_TIMESTAMP('19931001 000000000000','YYYYMMDD HH24MISSFF'))       AND (EXISTS (SELECT A1."L_RETURNFLAG" FROM "ABCH"."LINEITEM" A1                   WHERE (A1."L_ORDERKEY" = A0."O_ORDERKEY")                       AND (A1."L_COMMITDATE" &lt; A1."L_RECEIPTDATE"))) GROUP BY A0."O_ORDERPRIORITY" ORDER BY 1 ASC </pre>	
---	--

Figure 6-6 Query 4 rewrite

Also, though it may be obvious, we will point out that the original query joined two nicknames and we see in the access plan one SHIP with the two nicknames beneath it. This indicates that the join is pushed down and it appears in the RMQTX for the SHIP operator. If the join were not pushed down, we would see two SHIP's in the access plan, each over a different nickname, and there would

be a join operation (NLJOIN, HSJOIN, or MSJOIN) above the two SHIP operators.

So, even though the entire query is being pushed down, the SQL that is shipped to the remote source is usually different from the original query. This is because the DB2 optimizer rewrites the query as it sees fit, and that there are also differences in the SQL which is understood by DB2 vs. the SQL understood by the remote source.

### Query 14

Now let us look at Q14, which took twice as long on federated than through the native interface, as shown in Figure 6-4 on page 254. The absolute elapsed times are relatively short, around 5 and 10 seconds, but this is an interesting example nonetheless.

For the moment, let us say that this query is a two-table join with a somewhat interesting CASE expression that calculates a single aggregate value.

The key to understanding what happened is in the Explain plan, see Figure 6-7.

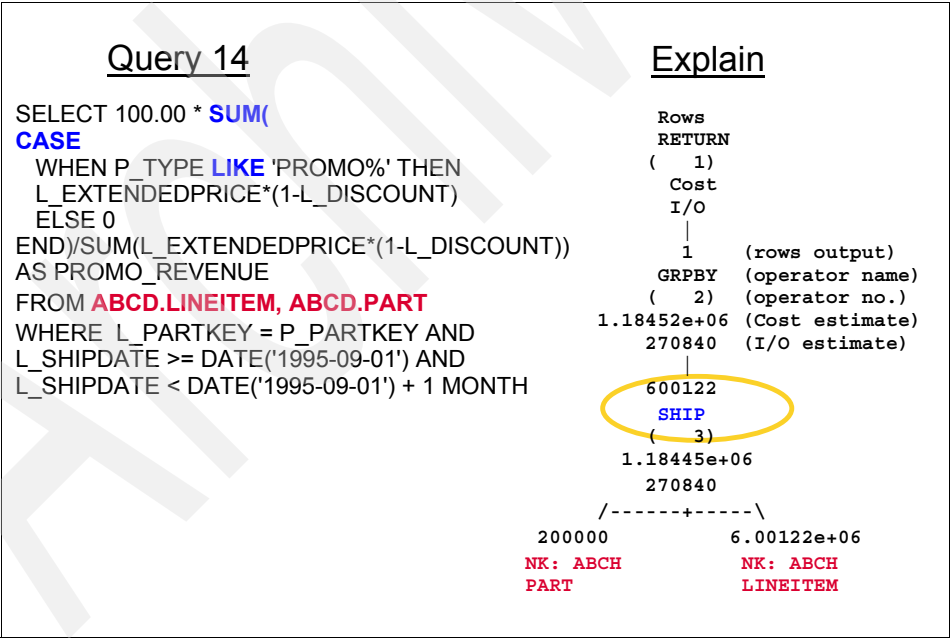


Figure 6-7 Query 14 rewrite and Explain



Remember how the legend for operators works: for each one, there's a name, number, costs, and above the name, the number of rows returned by that operator.

The join in this query is pushed down to the remote source, because both tables involved are underneath the SHIP node. But the SHIP node has returned back about 600k rows to the federated server. They are passed to the GRPBY operator above the SHIP node, which returns a single row. The aggregation (SUM/CASE/GROUPBY) is done locally.

Since the wrapper used to access this remote data source is not aware that this version of the data source can now support this type of CASE expression, it chooses not to push down the expression and thus the aggregation happens at the server. As we see, shipping so many rows from the remote source back to the federated server has a significant performance impact. In this case it would be possible to tune the server option to allow this pushdown to happen.

## Query 20

Let us look at the other extreme, query 20, which is twice as fast with DB2 federated than using the native interface.

The simplified Explain reported in Figure 6-8 shows that this query, a fairly complex join of five tables, was only partially pushed down. One join between two of the tables (operator 13), was performed at the remote source. Other joins were performed locally, for example, the nested-loop joins in operators 3 and 7.

How do you tell the difference between local and pushed-down joins? A SHIP node with multiple tables hanging underneath it implies a remote join. But any join that appears explicitly in an Explain (NLJOIN, HSJOIN, MGJOIN) is done locally.

Even in the local joins, though, the work of applying any local predicates (that is ones that involve only a single table), is usually pushed down. If you could look at the details on the SHIP operator (11), for example, you would see that a local predicate on the SUPPLIER table was evaluated at the remote source and only qualifying rows were shipped back to participate in the join at operator (7).

## Query 20 - A single-source 5-table join

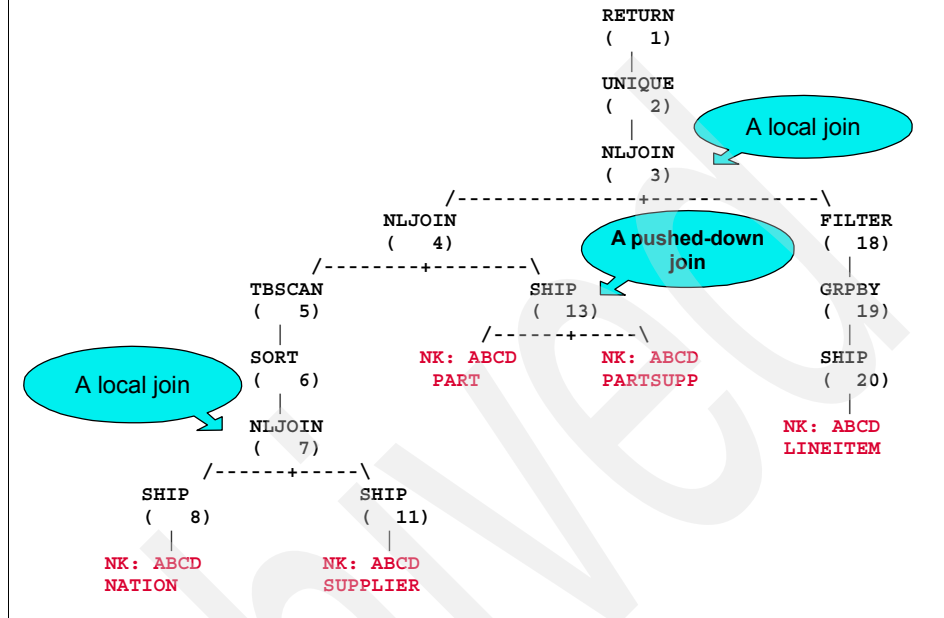


Figure 6-8 Query 20 Explain plan

This combination of remote and local processing by the federated server was apparently a good choice, as it executed the query faster than the remote source by itself. This is fairly rare, and indicates only that the remote source had chosen a less-than-optimal execution plan for the query. In this experiment, the remote source was only minimally tuned, so we could try to do other experiments such as setting `DB2_MAXIMAL_PUSHDOWN` to 'Y' to see the actual execution time if we do push down the entire query. We could also compare the remote plan to analyze why it is not performing as well as the decomposed version.

### 6.2.3 Federated server options for best performance

We have seen that the federated optimizer's decisions have a big impact on query performance. We look now at how we can make sure it does the best job possible.

Each remote data source has an entry in the federated server's catalog. Using special DDL, you can add more entries to the catalog that describe attributes of the server, called *server options*. It can really help to set these accurately.

We examine the most common server options:

► **COMM\_RATE, CPU\_RATIO, IO\_RATIO**

These describe the communication links to the remote source, and the relative speed of the remote system's CPU and I/O. By default, the federated server assumes that the remote machine is equal in power to the local machine, and that there is a 2 MB/sec. link to it. Setting these options to indicate a more powerful remote machine, or a faster link will tend to encourage query pushdown. These knobs are not perfect, but they are a way to indicate to the optimizer that a remote machine is fast or slow.

► **COLLATING\_SEQUENCE**

If you set this to Y, you are telling the pushdown that the remote source sorts characters the same way that DB2 does. This means that the federated server will consider pushing down operations involving sorting, grouping, or inequality comparisons on characters and varchar columns. For instance, `COLLATING_SEQUENCE Y` allows the optimizer to pushdown `ORDER BY` clauses that reference character and varchar columns. Pushdown of these operations on numeric, date, time, and date/time columns is not affected by this server option. Note that if the remote source's collating sequence does not match DB2's, you could get incorrect results!

► **VARCHAR\_NO\_TRAILING\_BLANKS**

This option is used for databases like Oracle that do not pad VARCHAR fields with trailing blanks. If you are sure that your VARCHAR columns do not contain trailing blanks to begin with, then setting this option at the server level will allow the use of the remote source's non-blank-padded comparison operations that return the same results as DB2. Again, note that an incorrect setting could mean wrong results.

Note also that for this option it could be better to use the nickname column option level. If it is at server level, the user would then need, for example, to ensure that all VARCHAR2 columns out of this Oracle data source indeed do not contain trailing blanks.

It is also very important to have accurate knowledge about statistical characteristics of a remote object, and available indexes to enable good optimizer decisions. The federated server relies on the remote source for its index and statistics information about each remote object. This information is retrieved when a nickname is created, but unfortunately, it is not automatically maintained if statistics on the remote object are refreshed or indexes are added or dropped, so you have to be careful to keep things up to date.

The federated server keeps information on a remote object's statistics and indexes in its catalog. The data describing remote indexes and statistics on nicknames is kept in exactly the same format as for local DB2 tables.

When a nickname is created, information about the remote object's statistics and indexes is retrieved and stored in the federated server's catalog. If the remote source does not maintain the kind of information that DB2 federated query engine uses, default values will get used instead. If you know the real values, you might think of inserting them into the federated server's catalog. Note that the federated server is completely dependent on the remote source for index and statistical information. In particular, it does not retrieve data from remote objects and attempts to gather statistics directly.

**Attention:** There is no RUNSTATS for nicknames. If statistics are updated or indexes change on the remote source, then the federated server's data will not be current.

When in doubt, drop and re-create the nickname to retrieve fresh information from the remote source. This can be somewhat disruptive, as it invalidates views and packages that reference the nickname, and it also clears any authorizations (GRANTS) on the nickname. Currently, a kind of **refresh** command for nicknames is not available.

For some data sources, the column statistics data maintained at the federated server may be minimal. Filling in the data by hand for important columns of frequently-used remote tables can improve query plans significantly. If you know basic properties about a column, such as minimum and maximum values, and the number of unique values it contains, this is not difficult to do. An example is shown in Figure 6-9.

SYSSTAT.COLUMNS is an updatable catalog view on the federated server. It stores statistical information on every column of every table. The really important attributes are the *column cardinality*, the number of distinct values that a column has, and *high2key* and *low2key*, which are the second-highest and second-lowest values in a column. This information is only important for nickname columns used in WHERE clauses, and column functions, as it helps the optimizer estimate the size of the result from the WHERE clause or function.

You can verify the values currently stored for a nickname with a query like Query 1 in Figure 6-9. You can then update the corresponding values by updating a row in SYSSTAT.COLUMNS as shown in Query 2 in Figure 6-9. If you do not know the second-highest and second-lowest values in a column, just use the MIN and MAX values if you happen to know the values. They will produce a much, much better plan than no values at all.

Manually updating these statistics will help the optimizer to build better execution plans.

## Improving nickname statistics

Query 1 - Checking column statistics for a table

```
select char(colname,20) as colname,  
colcard, char(high2key, 15) as high,  
char(low2key, 15) as low  
from sysstat.columns  
where tabschema = 'ABCD' and tabname = 'LINEITEM';
```

Query 2 - Setting column statistics for a nickname

```
update sysstat.columns  
set colcard=2526,  
    high2key  = '1998-11-30',  
    low2key   = '1992-01-03'  
where colname = 'L_SHIPDATE' and  
tabname = 'LINEITEM' and  
tabschema = 'ABCD';
```

Figure 6-9 Improving nickname's statistics

### 6.2.4 Analyzing performance of a single remote source query

We have seen how to look at query plans and how to help the optimizer find good ones. Now we go about analyzing the run time of a query to a single remote source.

First, get an Explain for your query Q from the federated server in the db2exfmt format. Check if the query is completely pushed down or not. If it is, look at the "RMQTXT" description in the details of the SHIP operator to find the exact text that is shipped to the remote source. Call this the remote query, Q'.

If you can, run the query Q' directly against the remote source using its native query interface and the same client server architecture. If Q' is about as long as Q, then you can safely conclude that the federated server is doing a reasonable job, and that any execution time issues need to be resolved on the remote source. In that case, you might want to Explain the remote query on the remote source.

If Q' is much faster than Q, then It could be that Q' looks very different from Q, and that they are getting very different remote access plans on the data source,

or the remote source may be returning a very large result set to the federated server.

If the query is not completely pushed down, but has only one SHIP operator, then you can go through the same steps to see how long the pushed-down portion of the query takes by itself. For a query with more than one SHIP operator, you might be able to analyze each of the remote queries on their own.

If the federated server seems to be adding a lot of overhead, you can use **db2batch** to measure CPU time at the federated server. It is also a nice tool generally for measuring elapsed time.

## 6.3 Tuning a query on multiple data sources

In this section we provide some examples on how to analyze federated queries against multiple remote data sources, and how to determine if they are performing at their best.

### 6.3.1 Multiple source queries

The ability to deal with data distributed over more than one physical data source is really the distinguishing feature of DB2 federated.

Usually, when we say multiple source or distributed query, we mean one that involves objects on two or more remote sources (nicknames). However, a query that involves both local data (on the federated server) and remote data (accessed through a nickname to a remote source) has many of the characteristics of a multi-source query.

The question is, how can we tell if a distributed query is performing well, and what does that even mean?

We will look at some qualitative and quantitative ways to evaluate distributed query performance. We will do it in the context of an example.

### 6.3.2 A simple distributed two-source query

Imagine a join of two tables called LINEITEM and PART. Suppose that these tables are physically located on two different data sources called ABCD2 and ABCD, and are represented by nicknames ABCD2.LINEITEM and ABCD.PART, respectively. The query is shown in Figure 6-10.

## A simple distributed two-source query

```
SELECT P_PARTKEY,
SUM (100.00 * L_EXTENDEDPRICE*(1-L_DISCOUNT)) AS
PROMO_REVENUE
FROM ABCD2.LINEITEM, ABCD.PART
WHERE L_PARTKEY = P_PARTKEY
and P_TYPE LIKE 'PROMO%'
AND L_SHIPDATE >= DATE('1995-09-01')
AND L_SHIPDATE < DATE('1995-09-01') + 3 MONTH
GROUP BY P_PARTKEY
ORDER BY PROMO_REVENUE FETCH FIRST 20 ROWS ONLY;
```

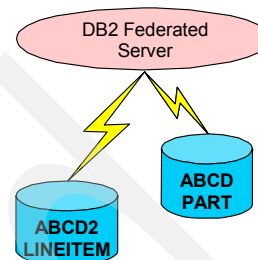


Figure 6-10 Distributed query

Each row in the LINEITEM table refers to one part purchased by a customer, and indicates when the item was shipped, along with price and discount information. The foreign key L\_PARTKEY references the primary key P\_PARTKEY of the PART table. Each PART has a P\_TYPE attribute, which may take on many values, including *PROMOTIONAL*.

We are checking the low performers and interested in knowing the revenue from the bottom 20 line items that were shipped between specific dates, and that reference promotional parts.

### 6.3.3 Execution plan for the simple distributed query

Let us look at how this query is executed by the federated server. There are four steps, and the three main steps are shown in Figure 6-11:

1. Issue remote query to source ABCD2 to filter LINEITEM rows on L\_SHIPDATE predicate and return 600 K out of 6 M rows to federated server (operator 8)

First, in operator 8, a remote query is shipped to the source where the LINEITEM table resides. The details in the SHIP node of the Explain show that it returns only those LINEITEM rows that satisfy the L\_SHIPDATE predicate. The optimizer estimates that 600,122 out of 6 million rows will be returned.

2. Issue remote query to source ABCD to retrieve filtered PART table rows - 7.5 K out of 200 K (operator 10)

Next, in operator 10, a query is shipped to the PART table that retrieves only the rows satisfying the *promotional* predicate. We think that this will retrieve 7,479 out of 200,000 rows.

3. Local hash join processed on the federated server (operator 7)

Operator 7 shows a local hash join between the results of the two remote queries. The rest of the processing includes sorting and grouping done at the federated server. It is estimated that the hash join will produce 23,442 rows and that the result of the SORT will be 25 rows.

- Final group by body 1. The related sort is not shown here.

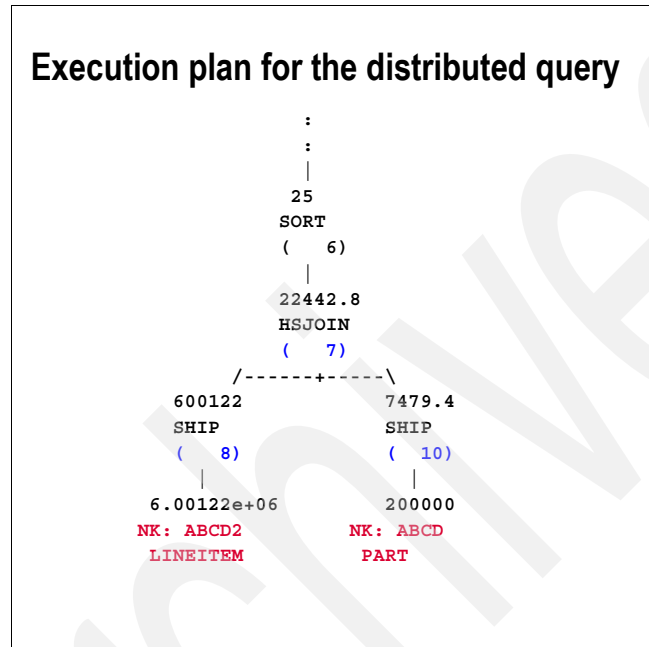


Figure 6-11 Distributed query - Execution plan

We will come back to this query. First, though, let us think about the factors that influence the performance of distributed queries like this one.

### 6.3.4 Performance of distributed queries

The first thing to note about queries against multiple sources is that there is no possibility of full pushdown in which all query processing is taken care of at a remote data source. The federated server always needs to do some local processing using the results of remote queries. Thus, the quality of the local execution plan becomes important.

Performance of such queries depends on the following factors, mostly the same factors as single-source queries:

- Pushdown of query fragments



Generally, pushing down as much processing as possible to remote sources improves performance, because it generally limits the amount of data that must be moved back to the federated server.

- Amount of data returned from remote sources to federated server

Data movement is again an important factor. And again, note that the federated server only moves data between the data sources and itself; it is not able to move data from one source to the other.

- How the federated server handles the data returned from remote data sources.

If a temporary table at the federated server is required, will it fit into available memory allocated for sorts, and for the buffer pool assigned to the temporary table space or will it overflow to the file containers of the temporary table spaces? This will require disk I/O to write the pages of the temporary table out to disk, and to scan them to process the remaining part of the query.

- Execution plan quality on the federated server

The quality depends on the accuracy of information.

- Data placement

Are the tables that need to be joined co-located on the same remote source? This is the new factor. For queries involving multiple tables, their physical locations really matter. A query that joins three tables will probably run faster if the two tables that need to be joined first physically reside on the same source, because that join may be able to be pushed down. If the two tables reside on different sources, the join between them will have to be done at the federated server after retrieving qualifying rows. This suggests that moving some of the data that is joined, so that more of it is collocated might improve performance. Also, moving some of the data to the Information Integrator database may also improve performance.

What constitutes good distributed query performance? How can we judge whether a distributed query is performing well, and what does that even mean? We can *eyeball* an execution plan and make a *qualitative judgement*. Good pushdown of operations seems like a good idea, so that data is not moved needlessly. And if we do have to move data, we should move as little as possible. For a given amount of data, it is also best to get it using the fewest possible interactions with the remote source.

Can we be more precise? Well, if we want to say that the federated system does a good job on a distributed query, we need to say that it does a good job compared to something else that we understand. This is a *quantitative comparison*. What are the alternatives that we could imagine to deal with distributed data if we did not use DB2 federated? Firstly, we could maintain a replica of some data from one source on another source, enabling us to do local

queries. That might be a reasonable solution if the replicated data is not too large, does not change very quickly, and we do not mind if it is a little out-of-date.

What if out-of-date data is not acceptable? Then we could do one of two things: We could write an application that connects to multiple sources using their native interfaces, and performs the query inside its own application logic. If we were prepared to accept slightly stale data, we might as an alternative decide to temporarily copy data from one source to another in order to do a local, co-located join on that source.

Any of these options makes an interesting performance comparison with DB2 federated.

Back to the sample query. Does it perform well? See Figure 6-12.

Qualitatively, it looks pretty good. The local predicates on Part and LINEITEM are pushed down to their respective sources, so that data can be filtered there. This means that only rows that actually qualify for the join are returned to the federated server. Also, the choice of a hash join, with the small number of filtered Part rows on the right side, looks good. DB2 federated can build a hash table in memory once with the qualifying Part rows, avoiding repeated interactions with source ABCD2.

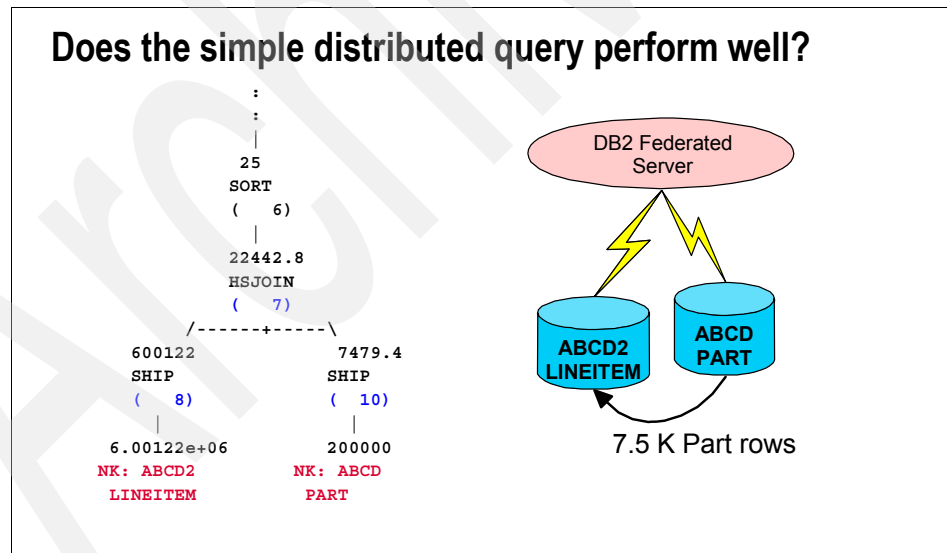


Figure 6-12 Back to the query

How about quantitatively? Instead of writing an application to access both sources and do the join, we elect to try the *move and join* method, in which data from both sources is moved to one place to do the join. Since the join is

somewhat lopsided (600 K rows on one side, 7.5 K rows on the other), it makes sense to move the little side to the big side.

We started the clock. Then we moved the qualifying Part rows from the ABCD source on the right to a temporary table on the ABCD2 source on the left by exporting to a flat file, using FTP, and loading the temp table. Finally, we executed the query completely on source ABCD2, since all the needed data is now available there. We then stopped the clock, and compared the total time to move the data and do a local join with the execution time using DB2 federated.

There is a trade off here. Notice that federated must move many rows to do the join:  $600\text{ K} + 7.5\text{ K} = 607.5\text{ K}$ . Our alternative strategy moves only 7.5 K rows, namely the qualifying PARTS rows. On the other hand, we need to export, move, and load. If we made a mistake and chose to move the qualifying LINEITEM rows to the source with Part, we would probably not do well at all:

- ▶ On source ABCD, we execute:  

```
select p_partkey from abcd.part where P_TYPE LIKE 'PROMO%'
```
- ▶ Export the result to a file.
- ▶ On source ABCD2, create a table called PARTTEMP that will hold the rows just collected from source ABCD.
- ▶ Move the file to source ABCD2, and use it to load the temporary PARTTEMP table there.
- ▶ Perform a local join between LINEITEM and PARTTEMP on source ABCD2.

Note that we can omit the `P_TYPE LIKE 'PROMO%'` predicate. Also, note that we did not execute `RUNSTATS` on the PARTTEMP table, nor did we create any indexes on it. Had we performed either of these additional two steps, we would have had to include them in the total measured time.

Is the move plus local join faster than the federated join? See Figure 6-13.

So here are the results of this experiment comparing federated with the *move and join* strategy for the sample distributed query. Moving qualifying rows from PARTS to be co-located with LINEITEM runs faster than running the same query through federated.

## Results of the comparison experiment

### Elapsed Time in seconds

Federated	Move Part to Lineitem, Local Join	Move Lineitem to Part, Local Join
109	57	496

Figure 6-13 Comparison results

But, if we make a mistake and choose to move things the other way around, which is the LINEITEM to PARTS described below, then federated wins by a long shot:

- ▶ On source ABCD2, we executed:

```
select * from abcd2.lineitem
WHERE (DATE ('1995-09-01') <= l_shipdate)
      AND (l_shipdate < DATE ('1995-12-01'));
```

- ▶ Export the result to a file.
- ▶ Move the file to source ABCD, and use it to load a temporary LINEITEMTEMP table there.
- ▶ Perform a local join between LINEITEMTEMP and PART on source ABCD.

The problem in this case is that we must export and load a fairly large number of LINEITEM rows, which federated avoids.

What is the conclusion here?

Move and join can perform well compared to federated if the join is lopsided and you correctly identify the smaller amount of data to be co-located with the larger amount.

However, it still allows for the possibility of slightly out-of-date data, and it is more work in that it requires additional scripting and setup. It would be hard to implement in a general-case way. Significantly, though, it can perform far worse than federated if you make the wrong choices about which data to move.

In summary, the object of this comparison was not really to convince you that you should not implement move and join. The idea was to put federated performance for distributed queries in some sort of perspective. For queries that you

understand very well, and that all follow a particular pattern, it may prove worthwhile to write a custom application or a *move and join* script to execute them.

For less predictable situations, the federated query optimizer's ability to make good decisions involving multiple sources becomes a large advantage. In an environment with changing data and dynamic queries, federated is a good bet.

### 6.3.5 Federated queries with nicknames and partitioned tables

Nicknames can be created in the DB2 ESE databases that have partitioned tables. Nicknames and partitioned tables can be referenced in the same query, but there are some performance considerations:

- ▶ In some cases, for certain queries, you may see the partitioning exploited, but in most cases, if nicknames and partitioned tables are joined, the join has to be performed at the coordinator node and the partitioning is not exploited.
- ▶ The usual reason that the join has to be performed at the coordinator node is that data from nicknames cannot be broadcast to partitions. This restriction is being studied by development. The implication is that a distributed request (DTQ) has to be sent to the partitions with the result flowing into a temporary table at the coordinator node so that the data of the partitioned table can be joined with the data from the nicknames. If the result of the distributed request to the partitions is large, then the temporary table can overflow the buffers that are available to the temporary table space at the coordinator node, and requires disk I/O to the file containers of the temporary table space.
- ▶ A technique to bypass this limitation if it will still provide a satisfactory answer to the users queries is to create either partitioned or replicated tables in the partitioned database to cache the data from the nicknames; refresh the data in these copy tables as appropriate to meet the users' requirements for the age of the data in them, and use the copies in the users' queries instead of the nicknames. If the copies are partitioned, DB2 will be able to broadcast (BTQ) their data to the partitions to be joined with the larger partitioned tables; if the copies are replicated, then they can be joined with the larger partitioned tables without broadcasting from one partitioned to another. Also, if the copies are created as materialized query tables (MQTs) that select from the nicknames, then the optimizer can make the choice between using the copy or the nickname, whichever will provide better performance for a particular query.

### 6.3.6 Summary of distributed query performance

Now let us step back and summarize what we have learned about distributed queries, as well as adding a few other things you should know.

Distributed queries can perform well if there is:

- ▶ Good pushdown of query fragments
- ▶ A lot of filtering at the sources, and not too much data moved to the federated server.

Tuning tips:

- ▶ Multiple-source queries require significant resources at the federated server (SORTHEAP, TEMP space, buffer pool for TEMP space)
- ▶ Hash joins are good for distributed queries. If using V7, enable them at federated server through DB2SET.

Limitations:

- ▶ No SMP intra-query parallelism at the federated server (yet). Operations involving nicknames cannot be parallelized. They are also synchronous!
- ▶ In partitioned databases (EEE), queries that join partitioned tables and nicknames may not perform well.

First, what kind of distributed queries are likely to perform reasonably using federated? Distributed queries that allow for good pushdown of processing to the remote source(s), and that do not need to return large number of rows in the intermediate results to the federated server are the best candidates. Even a distributed query involving two huge tables on separate sources can do well if it allows for good filtering at the sources before data is shipped to the federated server.

What about tuning the federated server? If it will be handling distributed queries, then you know that it will need to do significant local processing. You will want to add more hardware memory and allocate it to SORTHEAP and to the buffer pool assigned to the temporary tablespace. The default bufferpool for TEMPSPACE1 is IBMDEFAULTBP. Also, since we saw that hash joins are a good thing for distributed queries, you will want to enable them if you are using V7 (V8 enables them by default).

DB2 Federated Server does have some limitations. The first one is very important. If you are using a partitioned database (EEE), you can enable federated functionality in order to access remote objects through nicknames. However, any queries involving both a local partitioned table and a nickname will run on the coordinator partition only, and performance will be poor compared to the same join using all local data. The problem is the loss of parallelism. Note

that this problem affects only EEE systems that are federated clients. An EEE system can be a federated source with no special considerations. Any statements sent to it can execute in parallel.

Even if you are not using EEE, any processing involving nicknames on an EE system is currently not able to take advantage of any SMP parallelism. That is, even if you have enabled intra-partition parallelism, queries involving a nicknames will run serially. The implication is that if the federated queries are only used in batch mode to refresh copies of data either in the federated database or in other federated data source (insert into nickname/select from nickname), then additional CPUs will not help performance. Also, if federated queries are performed in online mode by multiple users, additional CPUs will probably not help performance unless there are multiple simultaneous requests from the users.

## 6.4 Performance and availability with MQTs

In this section we describe how to define and exploit materialized query tables over nicknames. In DB2 federated V8, you can now define MQT over queries involving relational nicknames, which can help improve performance of distributed queries involving local and remote data. In Figure 6-14 we depict a possible four-way join across two local tables and two nicknames at a remote data source.

## Materialized views on nicknames

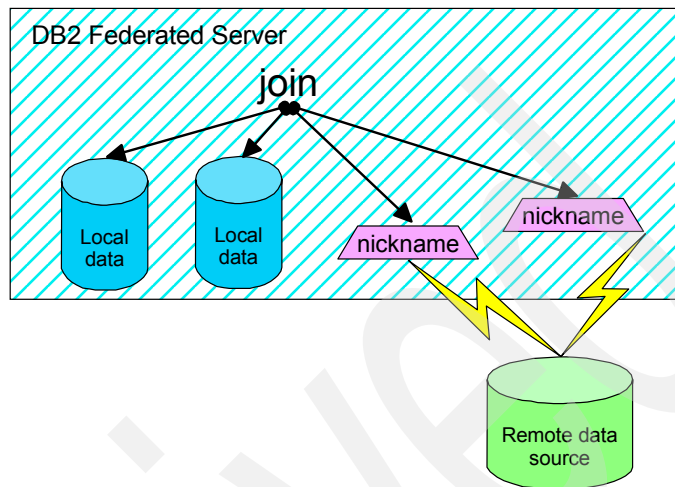


Figure 6-14 Materialized views

Suppose that you are joining local data with remote data accessed through nicknames. You frequently need to access this remote data; it is not too large, it does not change very often, and you can tolerate it being slightly out-of-date. Wouldn't it be nice to have a local copy of the data behind the nickname?

Materialized query tables are a way to define a local copy of data accessible through a nickname (see Figure 6-15). The local copy is not automatically updated; and must be refreshed.



## MQTs on nicknames

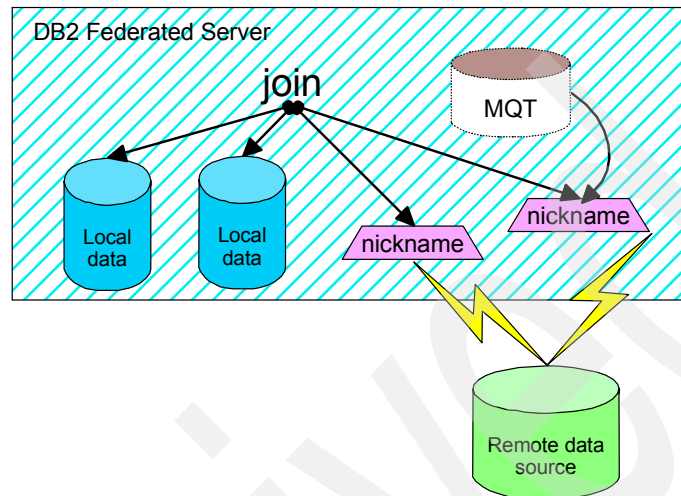


Figure 6-15 Materialized views on nicknames

So what is so “cool” about MQTs, and why would anybody do this instead of using replication? The MQT is smarter than a replica! If a query references a nickname, the optimizer may actually choose to use the nickname (and not the local MQT) if the query joins with other remote data on the same source. But if the nickname is joined with local data, then the optimizer will choose the MQT instead of the nickname. This stresses the importance of MQTs having good indexes and statistical information for the optimizer.

## 6.5 Federated query performance checklist

In this section we provide a checklist for tuning performance of federated queries. This checklist is based on the contents of the white paper *Using the federated database technology of IBM DB2 Information Integrator*, by Anjali Grover, Eileen Lin and Ioana Ursu, which is available from the Web site:

<http://www.ibm.com/software/data/pubs/papers/#iipapers>

The checklist helps you in identifying and analyzing bottlenecks in a federated environment.

Different techniques on tuning federated queries are mentioned, such as nicknames statistics, additional index specifications, materialized query tables (MQT), creating and analyzing access plans on your federated queries, setting server options for your federated objects, and tuning the query directly on your data source. We investigate most of these techniques in detail in Part 4, “Exploiting performance options” on page 291.

We focus on the steps for tuning query performance for federated systems. It is assumed that the federated system has been tuned for local processing in the areas of application heap, statement heap, buffer pool, sort heap, and temp table space. The DB2 snapshot monitor is a very helpful tool that provides a global view of the system. Information such as the buffer pool hit ratio, amount of sortheap overflows, and the system workload can be derived from the snapshot report. Ensure that your federated system is configured to perform efficient local join and sort operations before you start to look at tuning the queries.

We provide the list of issues that you should be aware of, rather than the exact order in which they need to be adjusted, as indicated in the figures below.

We have divided the checklist into three phases.

### 6.5.1 Phase 1

Figure 6-16 shows the first phase, Phase 1, of questions to ask when you start your evaluation of the federated performance:

► Step 1.1

db2batch and snapshot monitor are the tools to use to measure the queries. They are described in the *IBM DB2 Universal Database Administration Guide: Planning Version 8*, SC09-4822. You can determine if the performance is acceptable by finding out if you can tolerate the time taken to compile and execute the query, or if the time taken by the federated system is comparable to the time taken by the query when it is executed directly against the data source involved. You can also compare the federated system performance against performance of some existing user-written application program.

► Step 1.2

Currently, the RUNSTATS command is not supported for nicknames. The federated system collects statistics for relational nicknames and some non relational nicknames during nickname creation. It is a good idea to run a runstats-equivalent tool to gather statistics on the remote data source, and then drop and create the corresponding nicknames on the federated system. The query optimizer relies heavily on these statistics. The more accurate the statistics, the better is the plan. If for some reason the user does not want to drop the nicknames, the user may update statistics on the remote data source, and create a new set of nicknames on those remote tables. Now these new nicknames have current stats. The user may manually update the stats of the old nicknames based on the new set of nicknames. DB2 Information Integrator also provides a sample script get\_stats, which runs some SQL statements against the data source to gather stats, and updates the nickname stats on the federated system accordingly.

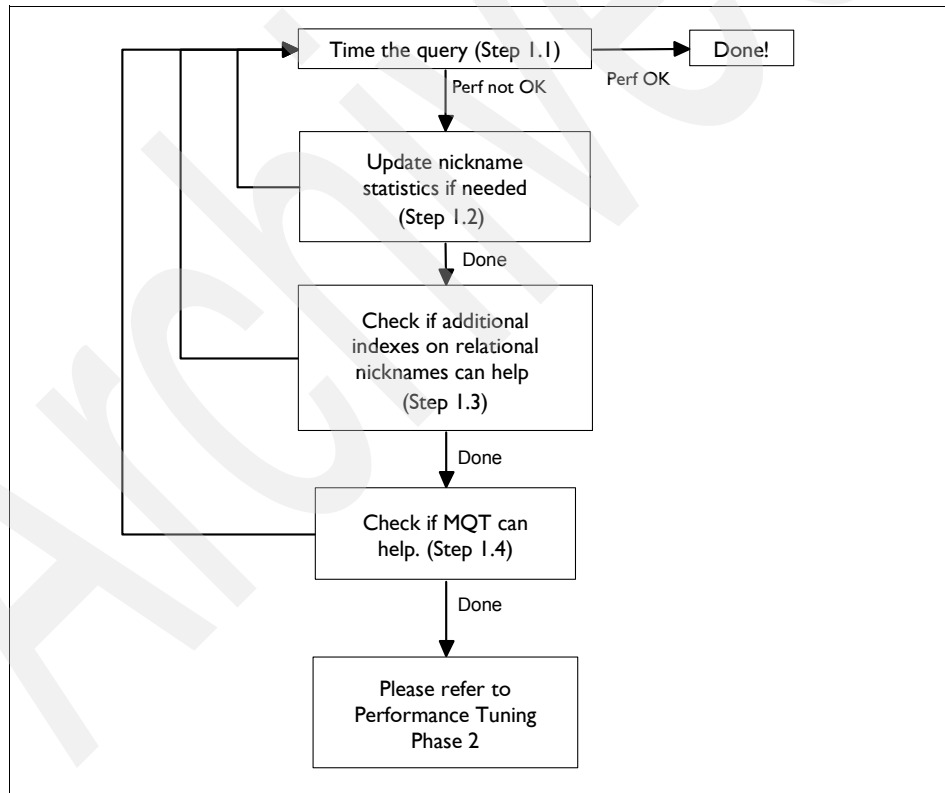


Figure 6-16 Performance tuning - Phase 1

► Step 1.3:

Defining indexes on the remote table on the data store may sometimes improve query performance. If you create a new index on an existing nickname, you might want drop and recreate the nickname to reflect the index information in the global catalog. Sometimes index specifications may improve performance. For examples, see Chapter 14, “Using index specifications” on page 355.

► Step 1.4:

Materialized Query Tables (MQT), also previously called Automatic Summary Tables (AST), can be defined to locally cache the results of queries that reference relational nicknames. This is especially helpful in cases where the network communication to such data sources can be slow, and users might be able to tolerate mildly stale data. Users do not need to modify the queries, the federated system detects which portion of the query can be answered with a specific MQT. Another side benefit is that if a query is chosen to be answered with MQTs only (no remote requests), the availability of the data sources will not be an issue for this query to be successfully answered. Currently, the *incremental update* for such MQTs is not available. Incremental update refers to the mechanism that updates the MQT when the underlying object is modified. For examples, see Chapter 15, “Using materialized query tables” on page 365.

## 6.5.2 Phase 2

See Figure 6-17 for a checklist on checking federated performance: Phase 2.

► Step 2.1:

The query execution plan may be obtained by using Visual Explain or **db2exfmt** output. Please refer to 7.1, “DB2 Explain” on page 284 to understand how to use these Explain facilities.

► Step 2.2:

Examine the **db2exfmt** output or Visual Explain output for the location and the number of remote plan operators in a statement (SHIP or RPD operator for SELECT and RETURN operator for Insert/Update/Delete). Check to see if each remote operator contains the maximal portion of the query that can theoretically be evaluated by a given data source. Inspect the optimized SQL in the **db2exfmt** or Visual Explain output, as this is what the relational data source pushdown analysis, and the optimizer bases the planning decision on. Experiment with rewriting the query to see how it affects the optimized SQL if necessary.

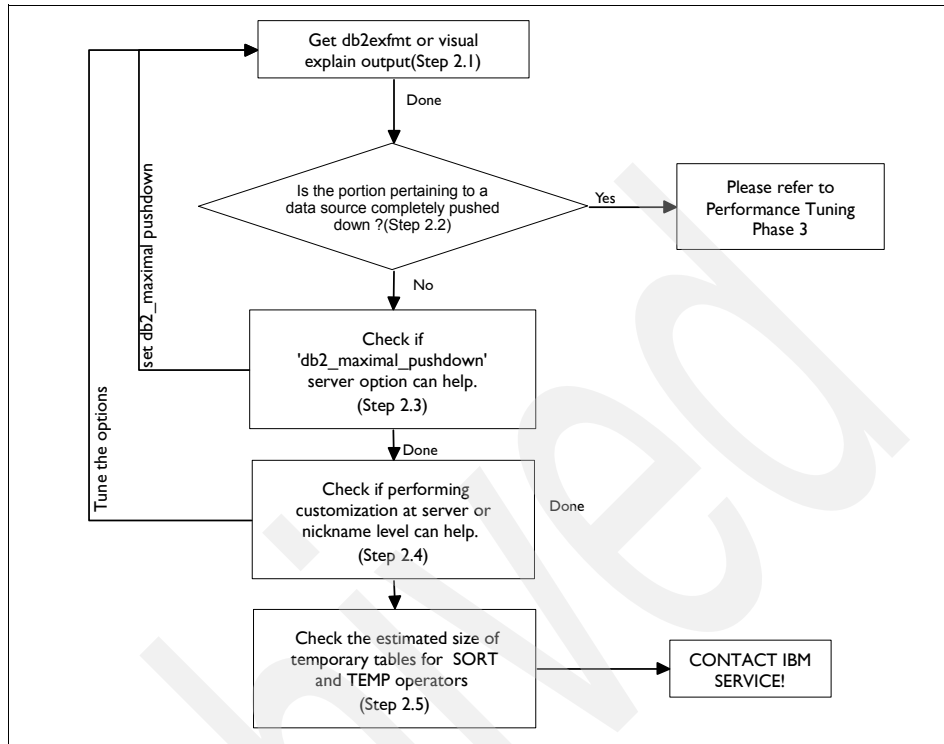


Figure 6-17 Performance tuning - Phase 2

► Step 2.3:

DB2\_MAXIMAL\_PUSHDOWN is a server option that you might consider setting to push as much work as possible to the remote data source. The query optimizer will try to reduce the number of trips made to the data source. If nicknames from more than one source participate in the query, you may choose to set DB2\_MAXIMAL\_PUSHDOWN for some servers and to reset it for others, depending on the capability of the remote query optimizer, the machine being used, etc. The query optimizer will give priority to a plan involving an MQT over a plan that involves access to remote tables. It will choose a query execution plan that pushes most work down to the data source within all the restrictions that are imposed on it (including, but not limited to data source restrictions, DB2 UDB restrictions, join enumeration method, etc.) when this option is set. For examples, see 10.1, “Server option db2\_maximal\_pushdown” on page 318.

The server option PUSHDOWN indicates whether the federated server should consider pushing down any SQL operation for a relational source. That is, when it is set to N for a data source, the federated server only sends remote requests similar to select column\_list from table, which definitely

affects the performance. The default setting is Y for all relational data sources except ODBC sources. For queries involving an ODBC source, consider setting this option if the data source can handle basic operations like predicates.

► Step 2.4:

To reduce the amount of data fetched from the data source, you can check if a predicate chosen to be evaluated locally on the federated system, should have been evaluated on the remote data source instead. Similarly you can determine if there is any GROUP BY or DISTINCT operator being performed locally that could have been evaluated on the remote data source. You might need to tune options at server, nickname and nickname column levels.

The federated system also provides three server options to adjust remote costing for relational data sources: CPU\_RATIO, IO\_RATIO and COMM\_RATE. CPU\_RATIO and IO\_RATIO indicate how faster or slower CPU and I/O on remote data source are as compared to the federated system. COMM\_RATE denotes the network bandwidth. If you want to push more work down to the remote data source (without setting DB2\_MAXIMAL\_PUSHDOWN), you might want to change these ratios to encourage the federated optimizer to utilize the faster CPU and IO on the remote data source. The query optimizer also tries to reduce the number of trips to the data source if the network is slow. Most federated system users do not need to tune these three options, as our experience has shown that default values for these options do relatively well.

► Step 2.5:

Check the estimate size of temporary tables in the access plan. These are SORT and TEMP operators in the plan. How many rows will there be? What is the row width? How many buffers will be required? Will the temporary table fit into the memory buffers available for the temporary tablespace (TEMPSPACE1); or will pages of the temporary table have to be written out to the file containers of the temporary tablespace. The Database Configuration parameter SORTHEAP indicates how much memory can be allocated for individual SORTs and IBMDEFAULTBP is the default bufferpool for the temporary tablespace. During execution of a query, the file system directory for the containers of TEMPSPACE1 can be monitored to see if files are created for temporary tables and how large the files become.

### 6.5.3 Phase 3

See Figure 6-18 for a checklist on checking federated performance, Phase 3.

► Step 3.1:

Block fetches are good for reducing the number of messages between the federated system and the remote data sources. It is enabled by default. The

database manager configuration parameter *RQRIOLBK* is used as the size of the csize of the communication buffer between the federated system and the data sources. Consider adjusting this setting to help the performance. However, some queries might present themselves as potential update targets (such as a query referencing only one nickname on the outermost SELECT), This forces the federated system to avoid using the block fetch feature as the remote cursor location will need to be maintained. For this reason, it is advised to append the FOR FETCH ONLY clause to those queries that will not be update targets or use BLOCKING ALL as bind option for your application

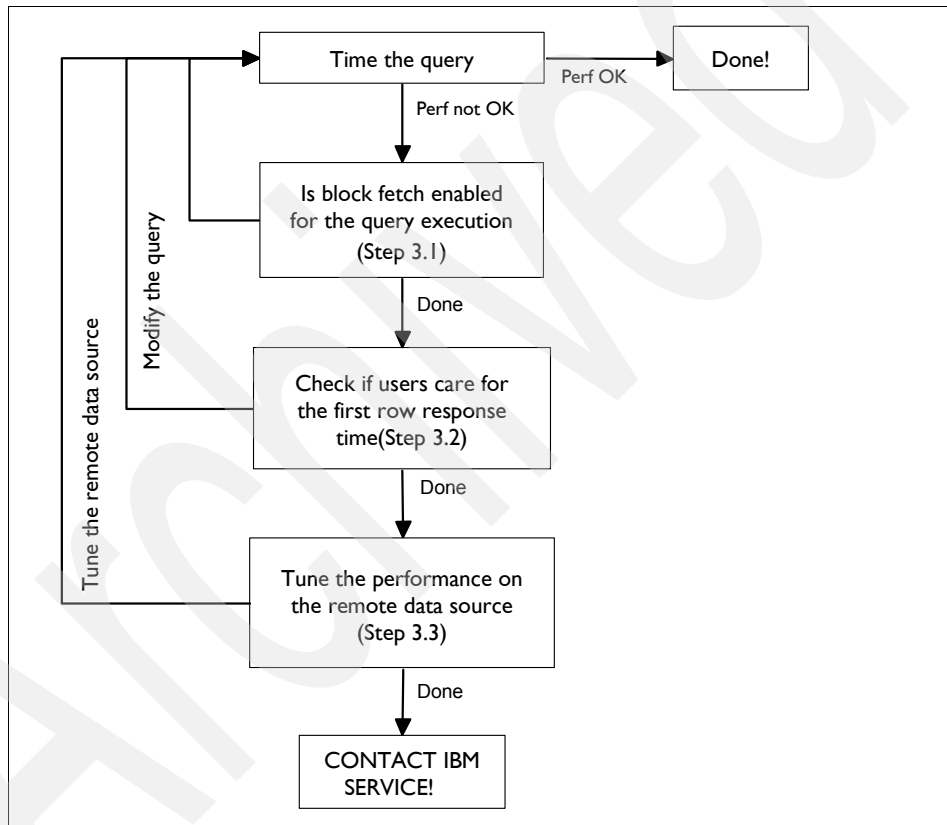


Figure 6-18 Performance tuning - Phase 3

Several parameters at different levels of the hardware/software “stack” can be examined and tuned to reduce the time taken to transfer large result sets from the data sources to the federated server:

- Network packet size. If the network packet size has been optimized only for terminal traffic, the maximum network packet size is probably small and

lots of packets will be required to transmit large result sets. If the maximum network packet size has been optimized for large data transfers, such as for FTP and downloads, then it will probably provide better performance for transferring large result sets from data sources to the federated server.

- Client software maximum packet or block size. The data source may have parameters that control the maximum packet/block/buffer size for transferring SQL statements, and results between the client and the server. Information Integrator uses the data source client software to access the data source. Increasing the maximum packet/block/buffer size that can be used between the client software and the data source server will permit the at source client software to receive large result sets in fewer transmissions from the data source.
- Information Integrator maximum block size. If results from data sources are greater than 32 KB, increase the DB2 UDB maximum block size. This is specified in the Database Manager Configuration of the DB2 Information Integrator instance; it is the RQRIOBLK parameter.

► Step 3.2:

In general, the query optimizer selects a plan that will return the entire result set in the shortest time. If a user only wants to see the first few rows of the query result without waiting for the entire result to be returned, the OPTIMIZE FOR N ROWS clause offers some performance tuning opportunity. The query optimizer may come up with a different query execution plan tailored to give good performance for fast retrieval of the first 'N' rows rather than all the result rows.

► Step 3.3:

Remote plans should be examined in response to the remote statements sent to each data source by using the equivalent of the EXPLAIN facility, if available. It is possible that the federated optimizer may have generated a remote statement that hits the remote server's limitation on performance. In this case, contact the IBM service team. If the remote plan appears optimal, the system administrator of the remote data source should be contacted to investigate the performance issue.



## Our performance tools

In this chapter we introduce DB2 performance tools that we used extensively during the project:

- ▶ DB2 Explain
- ▶ db2batch
- ▶ Get Statistics utility: `get_stats`

## 7.1 DB2 Explain

SQL is a high level language, which is parsed and transformed into access to data by DB2. DB2 builds an access path and the efficiency of this access plan depends on several factors. DB2 provides the most comprehensive *Explain* facility in the industry with detailed optimizer information on the access plan chosen for an explained SQL statement.

DB2 records all data related to the access plan in Explain tables. These Explain tables are generally defined by the user and then filled in with the access path information by DB2. The data in these tables can be accessed and presented in meaningful ways by utilizing several facilities.

For general information about DB2 Explain facilities, refer to the performance section in the *IBM DB2 Universal Database Administration Guide: Planning Version 8*, SC09-4822, under the Performance Appendix “SQL Explain Tools.”

DB2 Information Integrator V8.1 currently does not support the ability to see the Explain results on a remote source. The user needs to get the query fragment and Explain it at the data source.

### 7.1.1 Overview

DB2 Explain tools can be used to see the execution plan that DB2 is using to execute a SQL statement that includes nicknames. When there are performance problems, or to predict performance issues, we recommend the usage of these tools.

There are different features, where DB2 Explain tools supports you; basically it shows how federated server is trying to execute a query:

- ▶ Which operations are pushed down (SHIPped) to the data source including the SQL statement sent to the data source. The SQL statement pushed down to the data source can be entered into the vendor's performance tool to examine performance there.
- ▶ Does the optimizer have good information? Output displays the nickname statistics in the catalog on which the optimizer bases its decision on query execution strategy.
- ▶ On the DB2 Information Integrator server, will temp tables be used, because of lack of buffers in memory? Compare the size of the temp table as indicated in the access plan with the available and allocated memory for sorts and for the bufferpool assigned to the temporary tablespace. If the temp table are larger than the available memory, then some of its pages will require disk I/O to the file containers of the temporary tablespace.

- How many rows are estimated to be returned, and how many buffers will these records utilize.

## 7.1.2 DB2 Explain facilities

There are different ways of accessing and presenting the access path information:

### ► db2exfmt

This command based tool delivers a great graph of the execution plan and lots of details for each operation in the graph. It requires the DB2 Explain tables to be created by the user of the utility. The output is text based:

```
db2exfmt -d feddb -o explainedstmts.out
```

FEDDB identifies the database containing the Explain tables, that have records to be explained. The Explain tables are in the same schema as the user ID that db2exfmt is running. Explained stmts.out will be the output file:

```
db2exfmt -d feddb -n % -s % -g TIC -o feddb1.exfmt -w -1 -#0
-n      ... Name of source of explain (SOURCE_NAME)
-s      ... Schema of source of explain (SOURCE_SCHEMA)
-g TIC  ... Graph plan; T ... Include total cost in graph
              I ... Include I/O cost in graph
              C ... Include cardinality in graph
-o      ... output file
-w -1   ... Explain timestamp; -1 for the newest explain request
-#0     ... section number; 0 for all sections
```

### ► Visual Explain

This is a friendly tool available in the Command Center to show the access plans in a graphical way. It can also be run statically from the command line with the command **db2vexp.exe**. This GUI tool has the same functionality as **db2exfmt**. On initial usage it creates the Explain tables automatically. You can examine queries, whether static or dynamic, with each operation color-coded node in a tree structure. When you clicking on a node, you can view arguments, statistics, and cost estimate of the node.

To use Visual Explain in the DB2 Command Center:

- Open the DB2 Command Center.
- On the **Interactive** tab, connect to the federated database.
- Key or paste the query into the Command field.
- On the top menu, select **Interactive -> Create Access Plan**

Command Center will explain the query writing records to the explain tables, and then will evaluate these records and produce an access plan graph on the Command Center's Access Plan tab. If the explain tables have not already been created for you, Command Center will create them.

The access plan graph displayed on the Command Center's Access Plan tab is very similar to the access plan graph that can be found in **db2exfmt** output. There will be an icon for each operation of the plan and at the bottom of the graph will be the icons for nicknames and tables. The number appearing in the operator icons is the estimated cumulative cost for the operation. The number of rows estimated to be involved in an operation is in the detailed display for the icon.

On the **Access Plan** tab, you can find the details for the Access Plan that is displayed:

- Original SQL statement: on top menu, select **Access Plan -> Statement -> SQL Text**
- Optimized (Query Rewrite) SQL statement: on top menu, select **Access Plan -> Statement -> Optimized SQL Text**
- Instance/Database parameters: on top menu, select **Access Plan->Statement ->Optimization Parameters**
- Nickname statistics: Highlight icon for a **nickname ->Show Statistics**
- Operator details: Highlight icon for an **operation -> Show Detail**

RMQTX for a SHIP operation is in the **Input Arguments** of the detail display.

Rows (cardinality) for SORT/TEMP operations is in the **Cum Properties** of the detail display.

#### ► **db2expln**

This command-based tool delivers lightweight output. It shows the SQL statement sent to the data source, and a summary of the execution plan for SQL statements. It does *not* use any Explain tables.

#### ► **dynexpln**

The **dynexpln** tool can be used to describe the access plan selected for dynamic statements. It creates a static package for the statements and then uses the **db2expln** tool to describe them. However, because the dynamic SQL can also be examined by **db2expln**, this tool is retained only for backward compatibility.

Find general information about DB2 Explain facilities see the *DB2 UDB Administration Guide - Performance*

## 7.1.3 Explain tables

You need to use DDL to create the Explain tables. The DDL file named EXPLAIN.DDL is located in:

- UNIX: [db2 instance owner]~/sqllib/misc

- Windows: c:\Program Files\IBM\sqllib\misc

Characteristics of the script EXPLAIN.DDL is that the table names in the script are 1-part names, so they are created in the current schema of the user who runs this script. Additionally, the tables are created in the default table space for user tables, which is usually a Systems Managed Space (SMS) that is self-extending.

- Example for creating Explain tables:

```
db2 connect to feddb;
cd /home/db2fed32/sqllib/misc
db2 -vtf EXPLAIN.DDL
```

Explain tables are used whenever a query is explained with **db2exfmt** or Visual Explain. On each Explain process, DB2 will insert a group of records about the query into the Explain tables, which are in the schema of the user who is doing the Explain.

## 7.1.4 Explaining the queries

You have basically three different possibilities to activate explaining mode for a query:

- Using the Explain mode - for Explain format:

```
set current explain mode [no|yes|explain]
SELECT...
set current explain mode no
```

- Using the Explain snapshot - for Visual Explain:

```
set current explain snapshot [no|yes|explain]
SELECT...
set current explain snapshot no
```

- Using the Explain plan for statement:

```
explain plan for
SELECT...
```

We used the script listed in Example 7-1 to create access plans for our queries.

*Example 7-1 Our Explain script*

---

```
#!/bin/ksh
# Run query with explain mode and then format the explain tables
# Usage exfmt.ksh <SQL Query File> [query optimization level set a value]
# Usage: DIRIN is location of sql file
# DIROUT is location of output, name is <SQL Query File>.exfmt<optimization level>
if [[ $1 = '' ]]; then
echo "Missing Args: <SQL Query File> [Query Optimization Level : default is 5]"
```

```

exit
fi
QF=${1:-'sql.filename'}
# Default Query Optimization Level is 5
QO=${2:-'5'}
db2 terminate
#
# Change isolation level
db2 change isolation to cs
# Connect to DB2 Federated Database
db2 connect to feddb
# Change the query optimization level
db2 set current query optimization = $QO
# Change current mode to EXPLAIN MODE (no execute)
db2 set current explain mode = EXPLAIN
# Explain the query
db2 -tvf $QF
# Explain Format (Extract informations from the EXPLAIN tables (see file
$HOME/sql1lib/misc/EXPLAIN.ddl))
db2exfmt -d feddb -e ${USER} -n % -s % -g TIC -o $QF.exfmt$QO -w -1 -# 0

```

---

This script collects the actual access plan for the query and stores it into an output file. with the following steps:

1. Connects to our federated database
2. Sets the Explain mode to EXPLAIN
3. Performs the query in Explain mode
4. Finally performs the **db2exfmt** also

## 7.2 db2batch

This tool is used to perform one or multiple statements against a DB2 database and gather the related statistics. The statements are divided by a termination character, by default with a semicolon ; or by a delimiter specified within the **db2batch** command. We used the tool in our examples described in Part 4, “Exploiting performance options” on page 291 for measuring the actual execution time. Example 7-2 illustrates the **db2batch** syntax.

*Example 7-2 db2batch syntax and example*

---

```
db2fed32:/home/db2fed32>db2batch -h
```

```
db2batch -- version 2.0
```

```

db2batch -d dbname -f file_name [-a userid/passwd] [-t delcol] [-r
outfile,[outfile2]]
        [-c on/off] [-i short/long/complete] [-o options] [-v on/off]

```

```
[ -s on/off] [ -q on/off/del] [ -l x] [ -cli [cache size]] [ -h]
...
db2fed32:/home/db2fed32>db2batch -d feddb -f q.sql > q.out
```

---

## 7.3 Get Statistics utility: get\_stats

The Get Statistics utility is similar to RUNSTATS in the sense that it collects statistics on tables, and provides similar, basic statistics. After the scan of the nickname or index is complete, the data collected is inserted in the appropriate system tables as shown in Table 7-1.

*Table 7-1 Nickname catalog information updated by get\_stats utility*

Column	Description
sysstat.tables.card	Number of rows in a table
sysstat.columns.colcard	Number of unique values in a column
sysstat.columns.high2key	Second highest value in a column
sysstat.columns.low2key	Second lowest value in a column
sysstat.indexes.fullkeycard	Number of unique values in all the columns of an index
sysstat.indexes.firstkeycard	Number of unique values in the first column of an index

get\_stats is available for download from the DB2 Information Integrator technical support Web page:

<http://www.ibm.com/software/data/integration/db2ii/support.html>

and the DB2 Developer Domain Web page:

<http://www.ibm.com/developerworks/db2>

A get\_stats\_nr version of the utility, also available from the same Web sites, provides more suitable support for non relational data sources.





# Exploiting performance options

In this part we show several examples of implementing accesses to the data sources, and provide recommendations on how to improve performance. Each chapter is dedicated to a specific situation as follows:

- ▶ National language support  
Some examples of code page specifications
- ▶ Nickname statistics  
The availability of good and recent statistics is very important for the performance of a query.
- ▶ Major server options with relational data sources

There are server options that can influence the performance of a query accessing relational data sources. We examine two of them.

- ▶ Using data type mappings

It is important to have a correspondence between data types on remote sources and DB2.

- ▶ Using function mappings

You can create a function mapping if there is no default mapping available.

- ▶ Major server options with non relational data sources

The server options that can influence the performance of a query accessing *non* relational data sources

- ▶ Using index specifications

Index specification on a nickname allows you to use a new index on the remote data source.

- ▶ Using materialized query tables

Examples of how queries can take advantage of MQTs

## National language support

In this chapter we introduce the topic of code page specification by showing some examples of how to set the appropriate parameters in your DB2 Information Integrator environment in order to fully support a non-default national language.

This chapter is structured as:

- ▶ Introduction to code page settings
- ▶ Supporting a non-default language
- ▶ Adding a new language to an existing system

Although code page specification is more a usability topic, non explicitly related to performance, it is dealt with in this part because we do not have any better place, and we show some examples of how to set the code page correctly and minimize the costs associated with character set conversions.

## 8.1 Introduction to code page settings

When you display any document, you see a collection of characters or symbols. A group of characters or symbols taken together and treated as a single entity is called a character set. A character set may contain hundreds or even thousands of characters. In a single-byte character set (SBCS), a single octet (8-bit byte) is used to represent a single character. This means there are only 256 possible bit-patterns or code points available to represent a character. The collection of all 256 code points and the corresponding individual character assignments are called a code page. However, there are also mixed byte and double byte character sets to represent wide range and complexity of languages like Chinese.

There can be more than one code page for a character set. For example, the set of symbols used in the United States is different than those used in France. There are common characters (Latin alphabet, numerals, and some *specials*), but there are a lot of differences too. Sometimes the same symbol is used but is assigned a different code point, perhaps because different languages sort the character differently in their alphabet.

An encoding scheme is the collection of code pages for various languages on a specific platform. The EBCDIC scheme is used for zSeries and iSeries, the ASCII scheme for Intel and UNIX systems. They differ in how the characters are stored (code points) and produce results in different collating sequence. At least for DB2 for z/OS, code pages of the various encoding schemes are associated to, and identified by a Coded Character Set Identifier (CCSID), normally explicitly specified at install time.

Any server needs to support many different encodings, and whenever data is passed between different encodings or platforms, that data is converted, and there is always a risk of potential loss of characters that cannot be mapped.

Unicode was invented to address this situation by providing a unique mapping for every character, independently of the platform, or the language (see Appendix B, “Unicode tutorial” on page 395 for a brief introduction to Unicode). Incorporating Unicode into client-server or multi-tiered applications and Web sites enables the data to be transported through many different systems without conversion. Unicode is already being used by technologies such as Java, XML, and LDAP.

Unicode is not explicitly supported by DB2 Information Integrator, but Unicode support is intrinsic in DB2 UDB Version 8.1.

When you start your DB2 Information Integrator project, you need to find out which different code pages your system includes. You can start by analyzing the national language support (NLS) settings on your remote data sources. All the

different NLS settings on your remote data sources have to map to your federated server. This means, there must be equivalency in the code page conversion mechanism from every single remote data source to your federated server.

Figure 8-1 shows the three different layers of the DB2 Information Integrator architecture where you need to check national language settings:

- ▶ Layer one is the remote relational data source.
- ▶ Layer two is DB2 Information Integrator.
- ▶ Layer three is the DB2 Client.

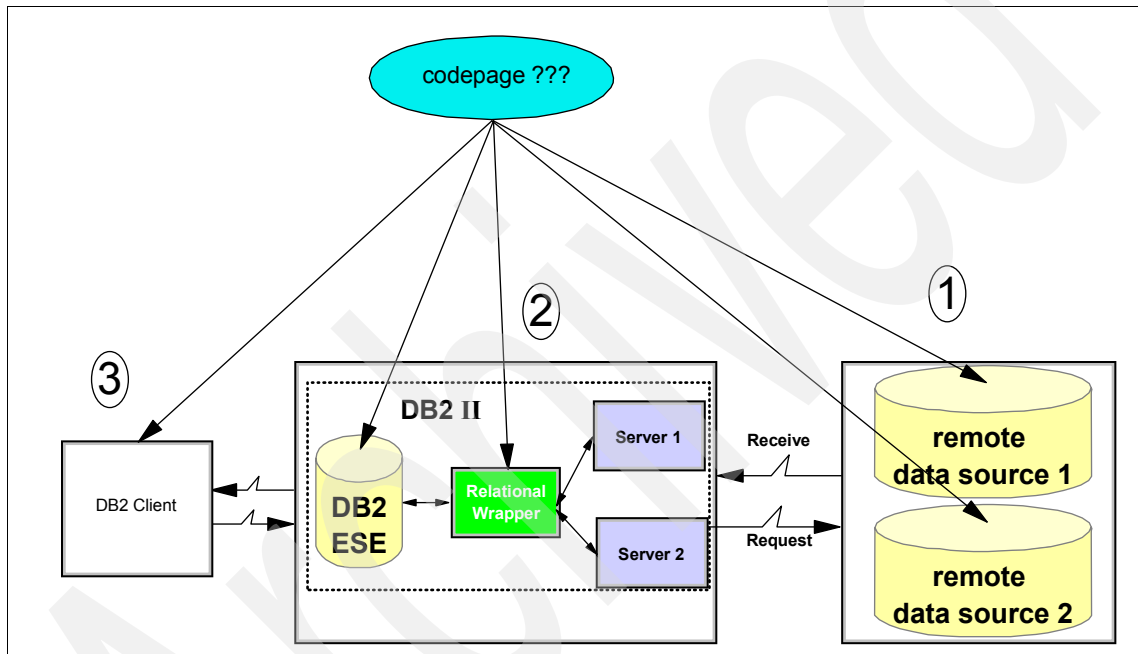


Figure 8-1 The three layers of code page setting

Another fourth possible level is the application; you also need to make sure that the language is handled correctly by your software package or application.

### 8.1.1 Remote relational data source

One characteristic of every database is the code page that has been assigned to it. The code page of a database is the entire set of characters the database is able to store. Some database systems support multiple code pages.

As a first step, you need to check the code page for each remote data source. You need this information in order for the DB2 Federated Server to create and set

automatically the correct client environment variables (that is NLS\_LANG for Oracle, CLIENT\_LOCALE for Informix) on the DB2 Federated Server.

See Appendix B.2, “Locale” on page 401 for the *locale* definition.

## 8.1.2 DB2 Information Integrator

Now we look to the national language definition from the DB2 Information Integrator perspective.

The wrappers for relational data sources have the information about which code pages they (and the data sources) are able to support. The wrapper then checks the code page of the DB2 Federated Server and configures the NLS environment automatically.

You can set the code page at two levels inside DB2 Information Integrator:

- ▶ When you create the federated database, you are able to specify the code page and collating sequence you want to use in your **create database** statement. All the NLS environment settings for each wrapper are derived from your DB2 Federated Server code page and territory settings.
- ▶ When you have specific knowledge of the content of the data you wish to store and if you want greater control, you can edit the db2dj.ini to override the wrapper selection for your environment. For instance, if the federated server uses an unusual code page, or there is an error in the wrapper’s lookup table, the wrapper might not set the client locale correctly. You can then override the client locale setting.

Examples of client environment settings are listed in Table 8-1.

Table 8-1 Client settings for relational wrapper

Remote data source	Client environment	Example
Informix	CLIENT_LOCALE	en_us.8859-2
Oracle	NLS_LANG	American_America. EE8ISO8859P2
Sybase	SYBASE_CHARSET	iso_1 sjis
Teradata	TERADATA_CHARSET	kanjiSJIS_OS

You can verify the DB2 Information Integrator configuration settings from the DB2 Command Line Processor (CLP) as shown in Example 8-1, and check the code set and territory values.

### Example 8-1 Checking your code page

---

```
GET DATABASE CONFIGURATION FOR <dbname>
```

Database Configuration for Database feddb

Database configuration release level	= 0x0a00
Database release level	= 0x0a00
Database territory	= US
Database code page	= 819
<b>Database code set</b>	<b>= ISO8859-1</b>
Database country/region code	= 1
...	

---

## 8.1.3 DB2 Client

The DB2 Client code page can be set by the DB2CODEPAGE registry variable. You might want to do that to avoid any character conversion or to convert all characters to the Unicode code page from the DB2 Federated Server.

DB2 uses UTF8 when transmitting data over the network. So, if for instance the client and the database use ISO8859-1, conversion will take place at the client and at the server.

After you have configured the DB2 Client settings, you need to check your application. In our case study we used two Microsoft Windows *applications* to perform the test:

- ▶ DB2 Command Center V8.1, which is included with the DB2 UDB Administrative Client, and also with all the DB2 Connect, DB2 UDB, and DB2 Information Integrator products
- ▶ DB2 Query Management Facility (QMF™) V8.1: A separate product available from IBM

Both of these applications support the Unicode code page.

**Note:** The DB2 Command Line Processor (CLP) environment behaves differently from Command Center and QMF. This is because under Windows, the CLP in DOS Windows uses its own code page.

If you use the CLP with Windows 2000, make sure to configure the Regional Options in the Windows control panel to check that the list in Language Settings for the system includes the language for the locale you need.

If this language is not included, you need to install it. To change the System Locale, click the **Set default...** button, and select the new setting from the list box, as shown in Figure 8-2.

After clicking **OK**, Windows will prompt you to either use the files already installed or to install new files. You can use the files already installed.

**Attention:** This step will require a restart of Windows.

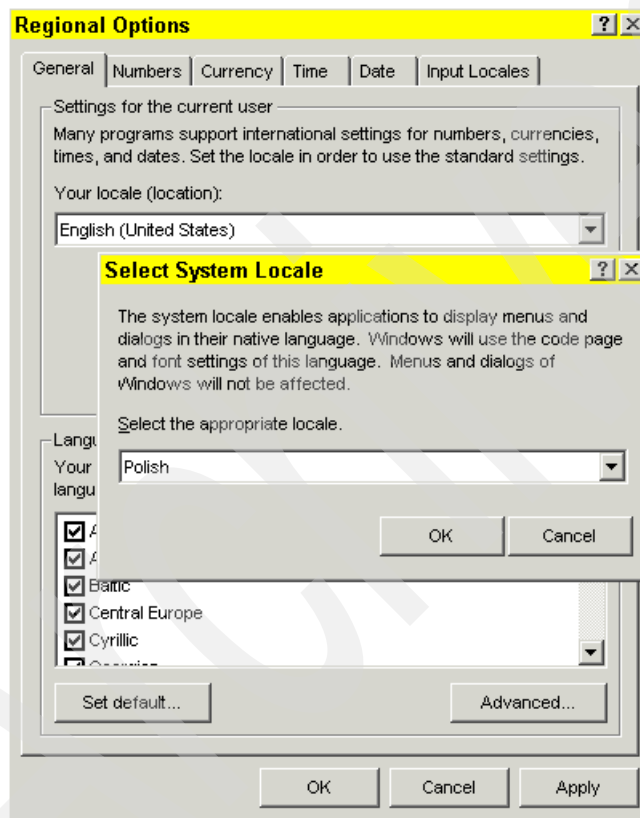


Figure 8-2 Windows 2000 - Regional options

All culturally sensitive parameters, such as date or time format, decimal separator, and others, are based on the current territory of the client.



## 8.2 Supporting a non-default language

In this example, we want to show how to set up DB2 Information Integrator properly when you need to access data that is not stored in a very commonly used code page like American (ISO8859-1) or West European (ISO8859-1). To perform this scenario, we use a new remote Oracle database with the East European code page definition (ISO8859-2 or EE8ISO8859P2 for Oracle setting).

### 8.2.1 DB2 Information Integrator setting

In this case, we are aware of the Oracle code page and we decided to create a new DB2 Federated database with the Central European code page (ISO8859-2) to coincide with the Oracle database as shown in Example 8-2:

*Example 8-2 Setting the DB2 code page*

---

```
CREATE DATABASE CEDB USING CODESET ISO8859-2 TERRITORY PL COLLATE USING  
IDENTITY
```

---

The Oracle NLS setting is automatically performed by the Oracle wrapper.

To emphasize the point that users can set the NLS\_LANG variable, Example 8-3 shows what has been set by the wrapper by default by using the \$HOME/sql/lib/cfg/db2dj.ini file:

*Example 8-3 Setting the Oracle code page*

---

```
ORACLE_HOME=/oracle9ir2  
NLS_LANG=POLISH_POLAND.EE8ISO8859P2
```

---

With these settings, effectively the character set at both sides is the same, so no conversion is needed between Oracle and DB2 Information Integrator.

### 8.2.2 DB2 Client setting

In this scenario, we now set DB2CODEPAGE (see Example 8-4) to the Central European code page for Windows 2000 from a DB2 Command Window (db2cmd). DB2 Client and DB2 Information Integrator will not perform any character conversion. You can get the list of DB2CODEPAGE values from Appendix B. National language support (NLS) of the *IBM DB2 Universal Database Administration Guide: Planning Version 8*, SC09-4822, under the supported territory codes and codes pages.

#### Example 8-4 DB2 registry setting

On Windows:  
db2set DB2CODEPAGE=1250

**Note:** If you do not set DB2CODEPAGE properly, you cannot connect to the DB2 Information Integrator Server and you will get this error message:

SQL0332N There is no available conversion for the source code page "1252" to the target code page "UNKNOWN". Reason Code "1".  
SQLSTATE=57017

### 8.2.3 Conclusion

With the setting shown in Figure 8-3, we are sure there is no data conversion across DB2 Information Integrator and Oracle, and DB2 Information Integration and the DB2 Client.

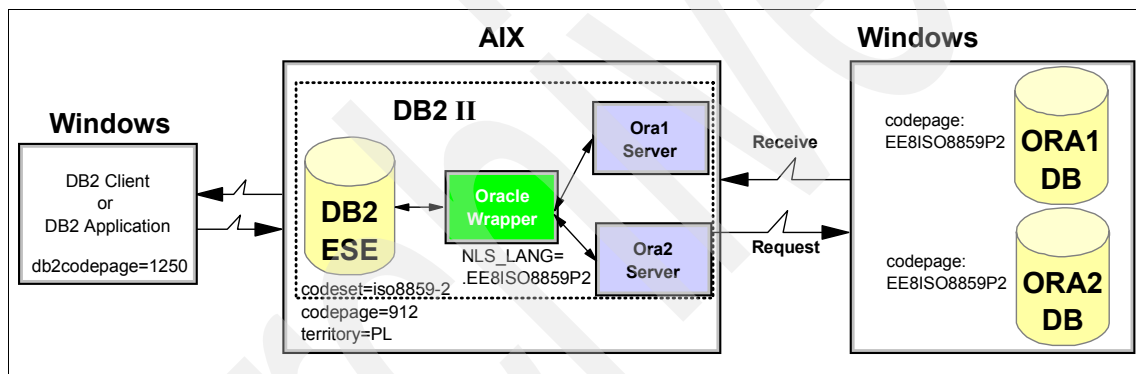


Figure 8-3 Setting the code page for no conversion

## 8.3 Adding a new language to an existing system

In this scenario, we assume that we need to add support for a new language in a remote Oracle database.

### 8.3.1 Remote data source code page definition

As shown in Figure 8-4, we have an existing federated system with DB2 Information Integrator set to the default US code page, and an Oracle remote source supporting the West European code page, and now we need to add a *new* Oracle database with a *new* code page for the East European code page.

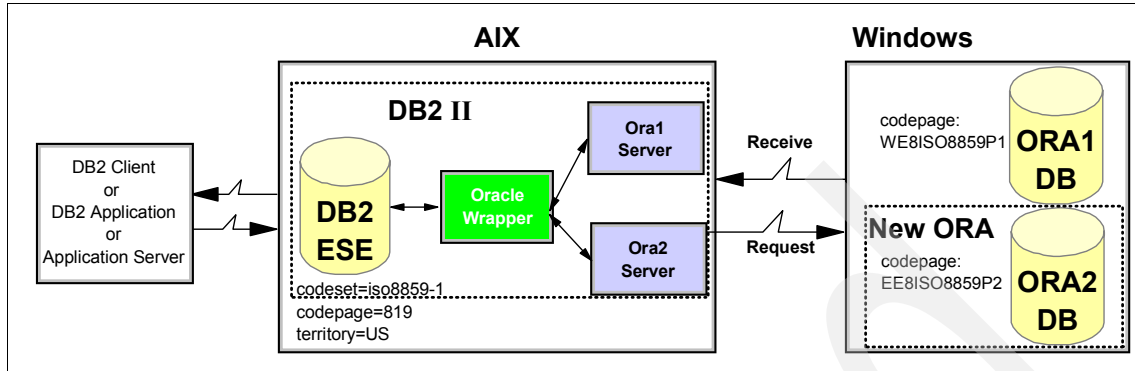


Figure 8-4 The default code page settings

Our intent is to have DB2 Information Integrator server retrieve data stored on Oracle data sources with East and West European code pages.

### 8.3.2 DB2 Information Integrator setting

For this DB2 Information Integrator server (see Figure 8-4), the internal setting for the Oracle wrapper is `NLS_LANG=AMERICAN_AMERICA.WE8ISO8859P1` and cannot be changed dynamically to `AMERICAN_AMERICA.EE8ISO8859P2`.

It is therefore not possible to implement the solution as described in Figure 8-4 because the DB2 Information Integrator database was not created with a Unicode code page. The reason is that the 8859-1 and 8859-2 codesets are disjoint, one is not a proper superset of the other, so there is no possible lossless mapping between them. A solution then is to create the DB2 Information Integrator database with a Unicode code page because the Unicode codeset is a full superset of 8859-1.

We created a new DB2 Information Integrator server with a Unicode specification like Figure 8-5.

DB2 UDB V8.1 supports Universal multiple-octet coded Character Set (UCS-2 (two bytes)). This means that it supports Unicode without surrogates and UCS Transformation Format (UTF-8), or in other words, it supports an algorithmic transformation from fixed length Unicode chars into variable length byte (or octet) strings.

When a Unicode database is created, CHAR, VARCHAR, LONG VARCHAR, and CLOB data are stored in UTF-8, and GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, and DBCLOB data are stored in UCS-2.

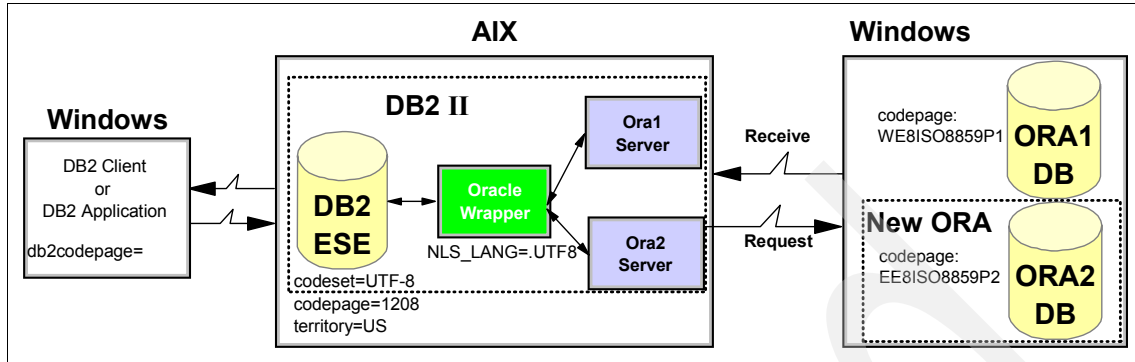


Figure 8-5 The Unicode code page settings

In Example 8-5 we show how we created a new DB2 federated database with the Unicode code page (UTF-8), and we therefore asked the Oracle Client to convert all characters to the UTF-8 code page:

*Example 8-5 Setting the Unicode code page for DB2*

---

```
CREATE DATABASE UFT8DB USING CODESET UTF-8 TERRITORY US COLLATE USING IDENTITY
```

---

A Unicode database allows connection from every code page supported by DB2 UDB. Code page character conversions between the client's code page and UTF-8 are automatically performed by the database manager.

The Oracle NLS setting is automatically performed by the Oracle wrapper.

To emphasize the point that users can set the NLS\_LANG variable, Example 8-6 shows what has been set by the wrapper by default by using the \$HOME/sql/lib/cfg/db2dj.ini file:

*Example 8-6 Setting the Unicode code page for Oracle*

---

```
ORACLE_HOME=/oracle9ir2
NLS_LANG=AMERICAN_AMERICA.AL32UTF8
```

---

### 8.3.3 DB2 Client setting

In this scenario, we do not need to set DB2CODEPAGE (see Example 8-7). DB2 Client and DB2 Information Integrator automatically perform the character conversion.

#### Example 8-7 DB2 registry setting

---

```
On Windows:  
db2set DB2CODEPAGE=
```

---

Going back to our sample applications at the client:

- ▶ The DB2 Command Center V8.1 is capable of printing all characters without a particular setting.
- ▶ For DB2 QMF for Windows, we need to change the default font from the *Microsoft Sans Serif* to *Lucida Sans Unicode* in order to be able to print all characters on the report window, otherwise we might lose some information.

### 8.3.4 Conclusion

With the settings shown in Figure 8-5, we are sure there is no data conversion from DB2 Information Integrator and Oracle, since all ASCII code page conversions are performed by the Oracle Client to UTF-8 and DB2 Information Integration, and DB2 Client conversions are made by the DB2 database manager.

**Note:** With the Unicode support, relational wrappers have some limitations with DB2 Information Integrator V8.1. You can check these limitations on the DB2 Information Integrator V8.1 documentation under **Known Problems** and **Workarounds**, in the most current *IBM DB2 Information Integrator Installation Release Notes Version 8*, available from the Web site:

<http://www.ibm.com/software/data/integration/db2ii/support.html>



## Nickname statistics

In this chapter we describe how the availability of good statistics for nicknames can influence the performance of a query.

The chapter is structured into:

- ▶ Overview
- ▶ Nicknames with missing statistics
- ▶ Nicknames with statistics
- ▶ Conclusion

## 9.1 Overview

DB2 stores statistical information on objects stored in the database including tables, table columns, and indexes. These statistics help the optimizer to work out the best access plan for queries. In order to help the optimizer do its job, it is necessary to keep the statistics for each object in the database up to date. DB2 stores statistical information for nicknames as well. As nicknames are really just local aliases for remote tables, they look much like local tables to the optimizer. In fact, statistics for both local tables and nicknames are stored in the same way, and are accessible through DB2 system catalog views in the schema SYSSTAT.

DB2 stores the following types of nickname statistics:

- ▶ Table cardinality (row count) and page counts (SYSSTAT.TABLES)
- ▶ Column cardinality (number of unique values) and column minima and maxima (SYSSTAT.COLUMNS)
- ▶ Information on remote indexes for nicknames (SYSSTAT.INDEXES)

The amount of statistical information stored for nicknames varies depending on the type of remote source involved. For example, while table cardinality is available for nicknames on most sources, column minima and maxima are available for only some sources.

Nickname statistics and index information are retrieved from available information on the remote source at the time that the nickname is created. Therefore, nickname statistics can only be as good as available remote statistics at nickname creation time. In particular, if no statistics have been collected on a remote object before a nickname is created, the nickname itself will not have any statistics. Similarly, if statistics are updated for an object on a remote data source, the new information is not automatically propagated to the corresponding DB2 nickname. The same principle applies to indexes; that is, DB2 is only aware of remote indexes for an object that are in existence at the time of the nickname creation. To make sure that DB2 nicknames have the best possible statistics and index data:

- ▶ Update statistics for objects on remote sources and create remote indexes *before* defining DB2 nicknames to them, so that DB2 can retrieve and store the statistics information for the nickname.
- ▶ If updated statistics are collected for a remote object, or a new remote index is created, the DB2 statistics and index information for the corresponding nickname will be out of date. You can drop and re-create the DB2 nickname for the object to re-start the remote statistics and index discovery process, and retrieve the updated information. Dropping and re-creating the nickname may have side effects such as view and package invalidation, and loss of GRANTs for the nickname



- Some remote data sources store little or no statistical data for their objects, or the information is not available to an external client. In this case, DB2 Information Integrator is not able to retrieve statistical data at nickname creation time. An alternative in this situation is to use a special tool called `get_stats` that populates DB2 nickname statistics by issuing queries directly against the nickname, or by updating the statistics and creating the index information manually using SQL.

In this chapter, we show a query example involving nicknames for three tables located on two different data sources. First, we access these tables through nicknames without having the right statistical information in place. We analyze the access plan and perform the query to check the elapsed time. After this step, we update the statistical information for all three objects on both remote data sources. We then analyze the access plan again.

The query we use throughout this chapter retrieves the total revenue, order date, and shipping priority for each order placed before March 15th, 1995 for the market segment BUILDING.

---

*Example 9-1 Revenue for orders*

---

```
SELECT
    L_ORDERKEY,
    SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT)) AS REVENUE,
    O_ORDERDATE,
    O_SHIPPRIORITY
FROM
    DB2ZOS.CUSTOMER,
    DB2ZOS.ORDERS,
    ORA.LINEITEM
WHERE
    C_MKTSEGMENT='BUILDING'
AND    C_CUSTKEY = O_CUSTKEY
AND    L_ORDERKEY = O_ORDERKEY
AND    O_ORDERDATE < DATE('1995-03-15')
AND    L_SHIPDATE > TIMESTAMP('1995-03-15-00.00.00')
GROUP BY
    L_ORDERKEY,
    O_ORDERDATE,
    O_SHIPPRIORITY
ORDER BY
    REVENUE DESC,
    O_ORDERDATE;
```

---

The following steps are performed:

1. Check the statistics stored for nicknames used in the query, and show that they are empty.

2. Create the access plan for the query having no nickname statistics using EXPLAIN and **db2exfmt**.
3. Run the query having no nickname statistics and measure the elapsed time with **db2batch**.
4. Update the statistics for all three remote objects on both data sources.
5. Drop and recreate the nicknames in the query to retrieve the statistics from data sources.
6. Check the statistics stored for nicknames to see that they are set.
7. Create the access plan for the query again with nickname statistics using EXPLAIN and **db2exfmt**.
8. Run the query with the nickname statistics and measure the elapsed time with **db2batch**.
9. Compare both access plans.
10. Compare both elapsed times.

## 9.2 Nicknames with missing statistics

This section describes the first three of the steps listed above, in the cases when having no statistical information on the nicknames. A nickname may be missing statistical information either because there is no statistical information stored on the data source itself, or because the nickname was defined before loading data and updating remote statistics at the data source.

### 9.2.1 Check statistics for nicknames

The statements in Example 9-2 retrieve all statistical information stored for the nicknames involved, which include table LINEITEM in schema ORA, and tables CUSTOMER and ORDERS in schema DB2ZOS.

---

*Example 9-2 Check statistical information*

---

```
select * from sysstat.tables
where (tabschema = 'ORA' and tabname = 'LINEITEM')
OR (tabschema = 'DB2ZOS' and tabname in ('CUSTOMER', 'ORDERS'));

select * from sysstat.columns
where (tabschema = 'ORA' and tabname = 'LINEITEM')
OR (tabschema = 'DB2ZOS' and tabname in ('CUSTOMER', 'ORDERS'));

select * from sysstat.indexes
where (tabschema = 'ORA' and tabname = 'LINEITEM')
```

```
OR (tabschema = 'DB2ZOS' and tabname in ('CUSTOMER', 'ORDERS'));
```

---

Table 9-1 shows a sample output for the first query listed in Example 9-2. We can see all statistical information stored for the three nicknames. In fact, there is no information available at this time. The unavailability of statistics is expressed with the value of -1. Based on this result, we suspect that statistics have not yet been collected for the underlying tables on the remote data sources.

*Table 9-1 Output from sysstat.tables*

TABSCHEMA	TABNAME	CARD	NPAGES	FPAGES	OVERFLOW	ACTIVE_BLOCKS
DB2ZOS	ORDERS	-1	-1	-1	-1	-1
DB2ZOS	CUSTOMER	-1	-1	-1	-1	-1
ORA	LINEITEM	-1	-1	-1	-1	-1

## 9.2.2 Create the access plan

Now we are ready to perform the creation of the access plan. Since our nicknames are missing statistics, the optimizer has to choose an access plan with very little knowledge about the underlying tables. In the absence of nickname statistics, the optimizer makes some default assumptions; for example, it assumes that all tables have 1000 rows. By executing the script listed in Example 9-3, we create the access plan by performing three steps:

1. Set the Explain mode to EXPLAIN.
2. Call the optimizer with the DB2 statement.
3. Print execution path with the **db2exfmt** utility.

*Example 9-3 Script to create the access plan for a query*

---

```
#!/bin/ksh
# Run query with Explain mode and then format the Explain tables
# Usage exfmt.ksh <query file>
if [[ $1 = '' ]]; then
    echo "Missing Args: <SQL query file>"
    exit
fi
QF=${1:-'sql.filename'}
Q0=${2:-'5'}

db2 terminate
#
#   Change to whatever you want
#
db2 change isolation to cs
```

```
db2 connect to feddb

db2 set current query optimization = $Q0
#db2 set current refresh age = any
db2 set current explain mode = EXPLAIN

db2 -tvf $QF

db2exfmt -d feddb -e ${USER} -n % -s % -g TIC -o $QF.exfmt$Q0 -w -1 -# 0
```

---

We discuss the result of the actual access plan in 9.4, “Conclusion” on page 313.

### 9.2.3 Perform the query

Now we want to execute the query shown in Example 9-4. We use the tool **db2batch**, which gives us the execution elapsed time. **db2batch** can optionally provide additional performance data, such as CPU consumption as well. Remember, we have no statistics stored on our nicknames at this time.

*Example 9-4 Perform the query without statistics*

```
db2batch -d feddb -f sc2query.sql -i complete
```

---

The query returns 2666 rows in about 176 seconds, as shown in Example 9-5.

*Example 9-5 db2batch output without statistics*

L_ORDERKEY	REVENUE	O_ORDERDATE	O_SHIPPRIORITY
-----			
606274.	4.19377576500000e+05	03/14/1995	0
181414.	3.93083442600000e+05	03/08/1995	0
...			
Number of rows retrieved is: 2666			
Number of rows sent to output is: 2666			
...			
Elapsed Time is:		176.479	seconds
-----			

---

We keep this number in mind as we proceed to the next section.

## 9.3 Nicknames with statistics

This section repeats the steps above after ensuring that the nicknames have updated statistical information. Because the remote sources involved in this example collect statistics on their objects and make them available externally, it is sufficient to update the statistics for all three tables, and then drop/recreate the corresponding DB2 nicknames. DB2 will retrieve the remote statistical information when the nicknames are created.

In cases where the remote source does not keep statistics on its objects, or keeps them in a form inaccessible to external clients, you can use the `get_stats` tool to populate DB2 statistical information for a nickname. `Get_stats` is available from DB2 Developer Domain, and is described in 9.3.2, “Option 2: Run `get_stats` utility” on page 312.

Another option would be to manually update the `sysstat` system catalog views for these nicknames on the federated server with the current stats. We examine this case in 9.4.2, “Summary” on page 315.

### 9.3.1 Option 1: Update stats and recreate nickname

This is a two step update:

#### Update statistics on data source

At the beginning of our example we assumed that we have no statistical information about the tables involved in the query on the data sources at all. This can happen when you create tables and indexes on a remote source, load data into your tables, and do not collect statistics before creating DB2 nicknames.

To update remote statistics data, we perform `ANALYZE TABLE` for the `LINEITEM` table on our Oracle data source, and `RUNSTATS` for the Orders and Customer tables on our DB2 for z/OS data source.

#### Recreate nicknames

Having current statistical information in place at the data sources, we are now ready to recreate the nicknames involved in our query. By recreating these nicknames, some statistical information stored in the system catalog of the data source for each table are transferred to the DB2 Federated Server, and stored in the system catalog tables.

*Example 9-6 Recreating nicknames*

---

```
DROP NICKNAME ORA.LINEITEM;  
DROP NICKNAME DB2ZOS.CUSTOMER;  
DROP NICKNAME DB2ZOS.ORDERS;
```

```
CREATE NICKNAME ORA.LINEITEM FOR ORACLE9.TPCH.LINEITEM;  
CREATE NICKNAME DB2ZOS.CUSTOMER FOR DB2ZSERIES.PAOLOR4.CUSTOMER;  
CREATE NICKNAME DB2ZOS.ORDERS FOR DB2ZSERIES.PAOLOR4.ORDER;
```

---

### 9.3.2 Option 2: Run get\_stats utility

Running the get\_stats utility is an alternative to running update statistics on the data source, and dropping and recreating nicknames. You may want to use get\_stats to avoid the side effects of dropping and recreating nicknames (which include package invalidation and loss of GRANTs for the nicknames). More commonly, get\_stats may be the only way to populate DB2 statistics information for nicknames on some remote data sources that either do not collect statistics, or do not make them available externally. Get\_stats directly collects the actual statistical information from the data source by running queries against the nicknames themselves. For example, to determine the number of distinct values in a column of a nickname, get\_stats will execute:

```
SELECT COUNT(DISTINCT <column>) FROM <nickname>
```

For tables with millions of rows, collecting statistics in this way can take a considerable amount of time.

*Example 9-7 Get\_stats utility on Lineitem, Orders, Customer*

```
db2fed32:/get_stats> ./get_stats_aix db2fed32 db2fed32 feddb ORA.LINEITEM  
db2fed32:/get_stats> ./get_stats_aix db2fed32 db2fed32 feddb DB2ZOS.ORDERS  
db2fed32:/get_stats> ./get_stats_aix db2fed32 db2fed32 feddb DB2ZOS.CUSTOMER
```

---

### 9.3.3 Check statistics for nicknames again

Now we want to check again what changed after recreating the nicknames or using the get\_stats utility. Table 9-2 shows the new output for the first query listed in Example 9-2. We can see all statistical information initialized to values stored in the remote data source statistics collection.

*Table 9-2 Output from sysstat.tables after update*

TABSCHEMA	TABNAME	CARD	NPAGES	FPAGES	OVERFLOW	ACTIVE_B LOCKS
DB2ZOS	ORDERS	336519	5624	5921	-1	-1
DB2ZOS	CUSTOMER	150000	5952	6266	-1	-1
ORA	LINEITEM	6001215	109159	109568	0	-1

### 9.3.4 Create the access plan

Now that we have the correct statistical information in place for the nicknames, we need to rerun the Explain. The optimizer has different information, and it will create a different approach for accessing data.

Please refer to 9.2.2, “Create the access plan” on page 309

### 9.3.5 Perform the query

Now we want to execute the query again. The optimizer now has correct information about the cardinality of the tables involved.

The answer set we get is the same, but with a much better elapsed time of 67 seconds, as shown in Example 9-8.

*Example 9-8 Perform the query with statistics*

L_ORDERKEY	REVENUE	O_ORDERDATE	O_SHIPPRIORITY
606274.	4.19377576500000e+05	03/14/1995	0
181414.	3.93083442600000e+05	03/08/1995	0
...			
Number of rows retrieved is: 2666			
Number of rows sent to output is: 2666			
...			
Elapsed Time is: 67.280 seconds			
-----			

## 9.4 Conclusion

In this section we discuss the result of this example. We compare both the access plan and elapsed time for the query without and with nickname statistics.

### 9.4.1 Comparing the access plans

Figure 9-1 shows two access plans. The left plan results from having no statistics on our nicknames DB2ZOS.ORDER, DB2ZOS.CUSTOMER, and ORA.LINEITEM. On the right side the access plan with the correct nickname statistics is displayed.

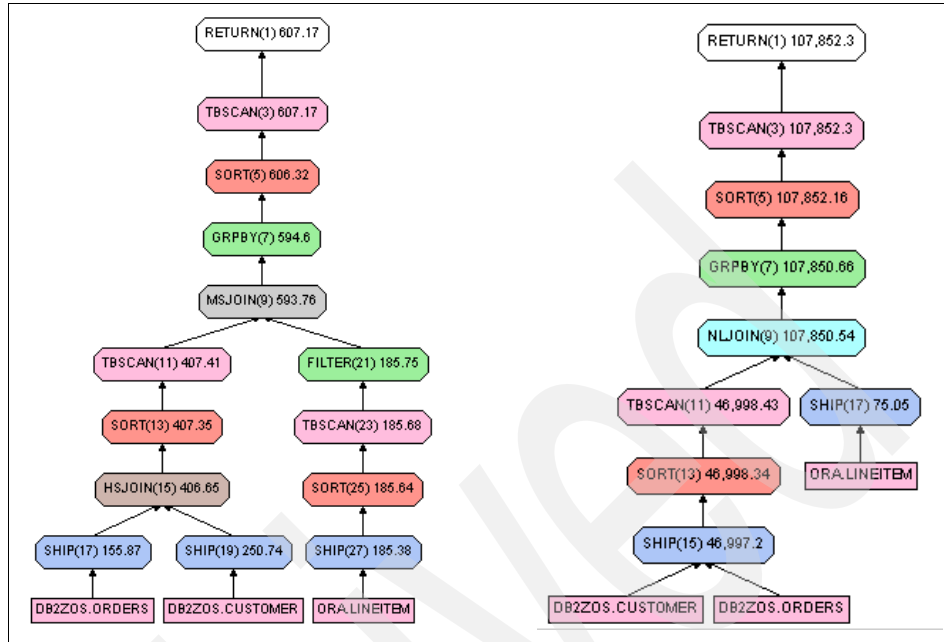


Figure 9-1 Access plan without and with statistics on the nickname

## Without statistics

The optimizer sends three requests (indicated by the SHIP operators) to the data sources; a separate query to each of the three tables is shown in Example 9-9. The optimizer's default assumption in the case of missing statistics is that each table contains 1000 rows, so it appears to be cheaper to fetch all rows and do the join, sorting, and grouping operations on the federated server. In particular, we notice in the plan on the left (created when there were no nickname statistics) that Information Integrator is using separate SHIP operations to DB2 for z/OS for each of the DB2 for z/OS tables, and joining the DB2 for z/OS data at the federated server using a hash join (HSJOIN).

### Example 9-9 Shipped queries - No statistics on nicknames

#### Ship Operator 17:

```
SELECT A0."O_CUSTKEY", A0."O_ORDERKEY", A0."O_ORDERDATE", A0."O_SHIPPRIORITY"
FROM "PAOLOR4"."ORDER" A0
WHERE (A0."O_ORDERDATE" < '1995-03-15')
FOR READ ONLY
```

#### Ship Operator 19:

```
SELECT A0."C_CUSTKEY"
FROM "PAOLOR4"."CUSTOMER" A0
```



```
WHERE (A0."C_MKTSEGMENT" = 'BUILDING ')
FOR READ ONLY
```

**Ship Operator 27:**

```
SELECT A0."L_ORDERKEY", A0."L_EXTENDEDPRICE", A0."L_DISCOUNT"
FROM "TPCH"."LINEITEM" A0
WHERE (TO_DATE('19950315 000000','YYYYMMDD HH24MISS') < A0."L_SHIPDATE")
```

---

## With statistics

With the right nickname statistical information in place, the access plan looks quite different. The optimizer now decides to pushdown the join for ORDERS and CUSTOMER to our DB2 for z/OS data source. The number of rows resulting from the join is by far smaller (about 33000) than the number of rows for both tables (about 490000 rows). This minimizes the amount of data transferred to the federated server.

The select for the Oracle table now gets an additional predicate, the column `l_orderkey`. Because there is an index on this column, we face the ideal situation to perform a nested loop join with the data coming from DB2 for z/OS.

*Example 9-10 Shipped queries - With statistics on nicknames*

---

**Ship Operator 15:**

```
SELECT A0."O_ORDERKEY", A0."O_ORDERDATE", A0."O_SHIPPRIORITY" \
FROM "PAOLOR4"."ORDER" A0, "PAOLOR4"."CUSTOMER" A1 \
WHERE (A0."O_ORDERDATE" < '1995-03-15') \
AND (A1."C_MKTSEGMENT" = 'BUILDING ') \
AND (A1."C_CUSTKEY" = A0."O_CUSTKEY") \
FOR READ ONLY
```

**Ship Operator 17:**

```
SELECT A0."L_ORDERKEY", A0."L_EXTENDEDPRICE", A0."L_DISCOUNT"
FROM "TPCH"."LINEITEM" A0
WHERE (A0."L_ORDERKEY" = :H0 )
AND (TO_DATE('19950315 000000','YYYYMMDD HH24MISS') < A0."L_SHIPDATE")
ORDER BY 1 ASC
```

---

## 9.4.2 Summary

Consider the following recommendations regarding statistics:

- ▶ Make sure to update statistics on your data sources, at a minimum when there are major changes in the cardinality of the tables.
- ▶ Based on available statistics on the data sources, make sure this statistical information is reflected in the DB2 nicknames to those tables.

- ▶ You can update statistics and index information for nicknames by any one of the following methods:
  - Dropping and recreating the nicknames to retrieve updated statistics from the remote source
  - Executing the get\_stats utility for nicknames
  - Directly (and carefully) manually updating the system catalog views SYSSTAT.TABLES, SYSSTAT.COLUMNS, or SYSSTAT.INDEXES for nicknames.
- ▶ Create the access plan for your queries to check whether nickname statistics are being used. If no statistics are available for a table or nickname, DB2 estimates a table cardinality of 1000 rows.

## Major server options with relational data sources

In this chapter we describe how some server options can influence the performance of a query accessing relational data sources.

The chapter describes three server options:

- ▶ `db2_maximal_pushdown`
- ▶ Server option `collating_sequence`
- ▶ Oracle server option `varchar_no_trailing_blanks`

## 10.1 Server option db2\_maximal\_pushdown

The external server option for relational nicknames db2\_maximal\_pushdown tells the optimizer whether to determine the execution plan based on cost (default behavior), or to favor pushing down the maximum number of operations. By setting this option to Y you direct the query optimizer to choose access plans that allow the remote relational data sources to evaluate as much of the query as possible, regardless of cost.

If it is enabled for a particular source, the federated optimizer will choose a plan that minimizes the number of trips (SHIP operators) made to that data source. If nicknames from more than one source participate in the query, you may choose to set db2\_maximal\_pushdown for some servers, and to reset it (i.e. disable it) for others depending on the capabilities of each one.

This setting makes sense if the remote source is as intelligent as the federated server, and in all cases when you want to compare the performance of your queries. For more details on its functionalities see 6.1.4 “Pushdown analysis and cost optimization” on page 249.

### 10.1.1 Overview

In this section we want to show a sample experiment to demonstrate the effect of the db2\_maximal\_pushdown option.

The example involves two remote data sources, one is DB2 UDB, the other is Oracle, and covers the following steps:

1. Server option DB2\_MAXIMAL\_PUSHDOWN is set to N, which is the default.
2. Perform the query shown in Example 10-1.

*Example 10-1 Query for db2\_maximal\_pushdown*

---

```
explain plan for
select count(*) from
ora.part, ora.partsupp, ora.supplier, db2dat.nation, db2dat.region where
r_name = 'ASIA' and r_regionkey = n_nationkey and
n_nationkey = s_nationkey and
s_suppkey = ps_suppkey and
ps_partkey = p_partkey and
p_type not like 'PROMO%';
!db2exfmt-dfeddb-1-t|more;
```

---

The first three nicknames in the FROM list reference tables on the Oracle source; the last two nicknames reference tables on the DB2 UDB source. Collect the query access plan:

1. Analyze the access plan against two sources, Oracle and DB2.
2. Set `db2_maximal_pushdown` to Y for the Oracle server. This enables the pushdown of the query to the Oracle remote source.

**Important:** Enabling this option offers an alternative comparison, not a guarantee for better performance.

3. Perform the query again.
4. Collect the query access plan.
5. Analyze the access plan, explain it again, and show how this server option is used to minimize the number of SHIP operators in a query, and push down as much processing as possible to the Oracle server.

The intent is to show what this option provides, and that it does not necessarily improve performance (i.e. the optimizer might have already made the right decision). Its main use is to test the performance impact of maximal pushdown to a server for a given set of queries.

### 10.1.2 DB2\_MAXIMAL\_PUSHDOWN set to N (default)

The query execution plan obtained for the sample query *without* the server option enabled is shown in Figure 10-1.

The response time is 11.8 seconds.

Note that the join between the two nicknames (NATION and REGION) on the DB2 UDB source is already pushed down underneath a single SHIP operator. However, there are two SHIP operators involving Oracle nicknames. The first retrieves the result of a pushed-down join to the Oracle PART and PARTSUPP tables. The second retrieves rows from the Oracle SUPPLIER table as the right side of a nested join processed by the DB2 Information Integrator server.

Also, note the Total cost: 69726.6 for this access plan.

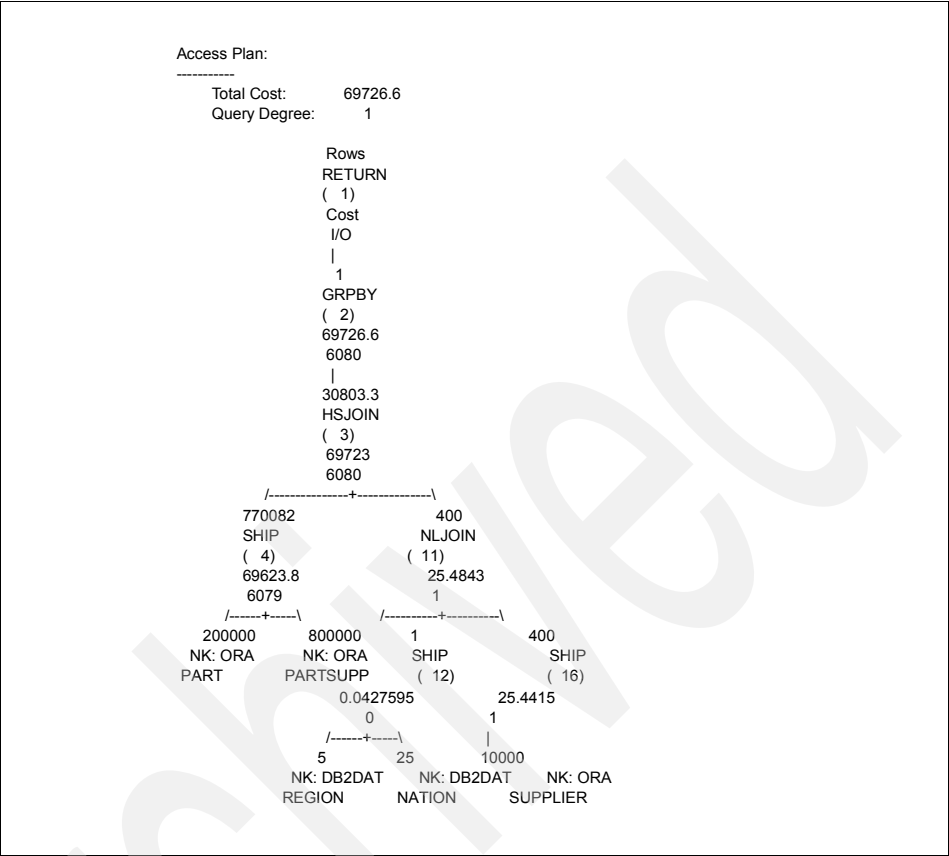


Figure 10-1 Query plan for db2\_maximal\_pushdown to 'N'

### 10.1.3 DB2\_MAXIMAL\_PUSHDOWN set to Y

We now enabled db2\_maximal\_pushdown for the Oracle server to reduce the number of SHIP operators to Oracle.

The attributes and function mappings in the Oracle wrapper tell pushdown analysis (PDA) what SQL operations and functions are supported by Oracle Version 9. PDA sets the restrictions that will determine the minimum number of SHIPs required to process the query. With db2\_maximal\_pushdown, we are telling the optimization phase of the query preparation the access plan with the minimum number of SHIPs instead of the lowest estimated cost.

Enabling db2\_maximal\_pushdown on the Oracle source will minimize the number of SHIP operators to Oracle. For this query, that minimum is one, as the join between all three Oracle nicknames can be completely pushed down. In

general, the minimum number of SHIP operators to a given remote server may be more than one, depending on the functionality that can be pushed down.

It does not make sense to enable `db2_maximal_pushdown` for the DB2 UDB server, because as much work as possible has already been pushed down to it in our sample query. Enabling the server option would thus have no effect on the portion of the execution plan involving the remote DB2 server.

The query execution plan obtained for the sample query *with* the Oracle server option enabled is shown in Figure 10-2.

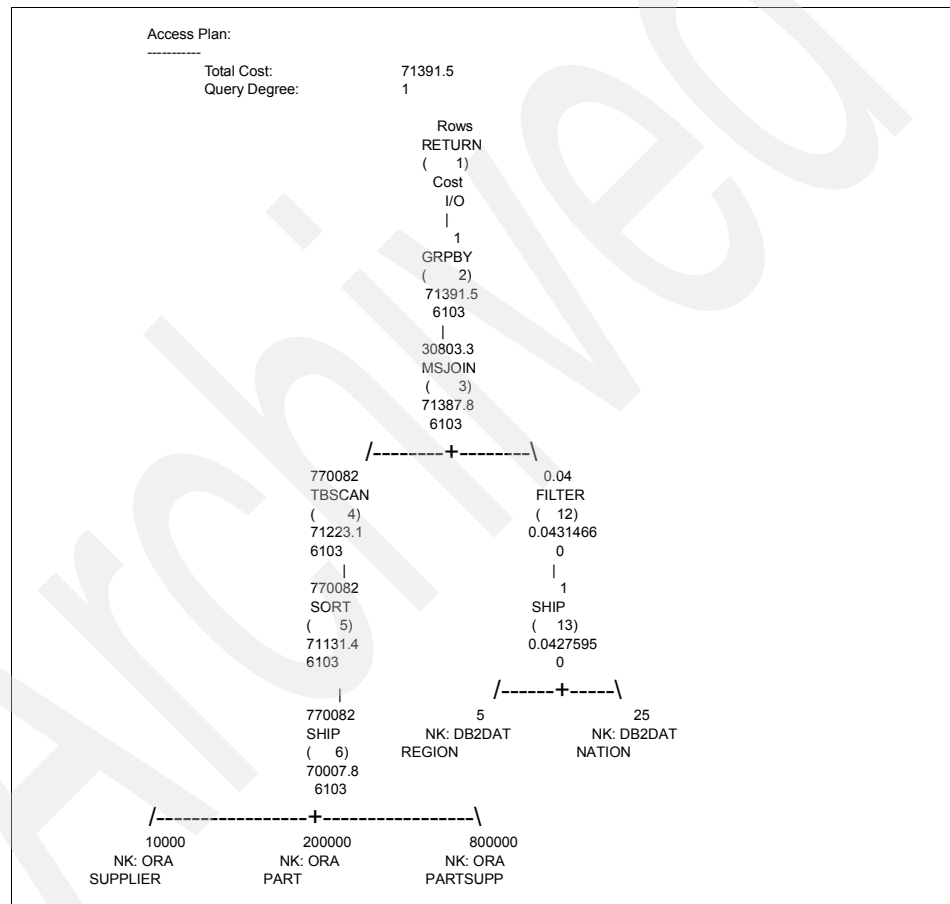


Figure 10-2 Query plan for `db2_maximal_pushdown` to 'Y'

The response time is 14.3 seconds.

Note that all processing involving Oracle objects is pushed down to Oracle, in contrast to the first case.

Also, note the Total cost: 71391.5 for this access plan. The basic inputs to this cost estimate are the same as for the plan where the options was not set, that is:

- ▶ The statistics for the nicknames
- ▶ Index information for the nicknames
- ▶ Server options CPU\_RATIO, IO\_RATIO, and COMM\_RATE.
- ▶ The operations in the plan

In this case, the first three inputs are the same, but the fourth one (the operations in the plan) is different, and so the cost estimate is different from the plan where DB2\_MAXIMAL\_PUSHDOWN was not set.

It is a higher cost estimate than the cost estimate (69726.6) for the access plan in Figure 10-1 on page 320 that we got with DB2\_MAXIMAL\_PUSHDOWN setting 'N' (the default setting). The higher cost for the plan with DB2\_MAXIMAL\_PUSHDOWN setting Y explains why this plan (Figure 10-2 on page 321) was not selected by the optimizer when the option was not set.

If after setting DB2\_MAXIMAL\_PUSHDOWN to Y and explaining the query, we had again gotten the same access plan that we got with DB2\_MAXIMAL\_PUSHDOWN set to N (Figure 10-1 on page 320); this would have told us that PDA is restricting the three-table join from being pushed down to Oracle. An access plan with only one SHIP to Oracle would not have even been available for the optimizer to evaluate.

#### 10.1.4 Conclusion on DB2\_maximal\_pushdown

In our sample query, execution time rose from 11.8 seconds to 14.3 seconds when db2\_maximal\_pushdown was enabled for the Oracle server. While pushing down processing to the remote source(s) in a query is generally beneficial, our example shows that “full” pushdown does not always offer the best performance. In this case, forcing the SUPPLIER table to be joined with the PART and PARTSUPP tables early in the plan means that an attractive index on the SUPPLIER table is no longer available to be exploited by the DB2 II server. We see that for this query, the best performance is obtained when processing is appropriately shared between the DB2 Information Integrator server, and the remote Oracle and DB2 UDB servers. In this case, the optimizer did the right thing without the use of db2\_maximal\_pushdown.

Do not enable this option indiscriminately; use the db2\_maximal\_pushdown option if you have good reason to believe that forcing pushdown of processing will have performance benefits, and verify that you are obtaining them.



### 10.1.5 DB2\_MAXIMAL\_PUSHDOWN vs. PUSHDOWN

Enabling pushdown simply allows the optimizer to consider execution plans that push down simple processing to the remote source. In contrast to `db2_maximal_pushdown`, it does not force pushdown. The use of pushdown includes pushed-down plans for the optimizer's consideration, but pushdown is still chosen (or not) on the basis of costing. `db2_maximal_pushdown` forces as much pushdown of processing as possible without regard for optimizer costing.

## 10.2 Server option `collating_sequence`

This server option tells the optimizer if the remote source sorts characters the same way that DB2 does. This means that the federated server will consider pushing down operations involving sorting, grouping, or inequality comparisons. Note that if the remote source's collating sequence does not match DB2's, you can get incorrect results. In this chapter's example, we demonstrate `collating_sequence` characteristics with an inequality comparison.

### 10.2.1 Overview

The example involves our Oracle data source and covers the following steps:

1. Server option `COLLATING_SEQUENCE` is set to `N`, which is the default.
2. Do `DESCRIBE TABLE` for the nickname `ORA.PART` (or use `Alter Nickname` in the Control Center) to determine the data type of the column `P_NAME` that will be used in our query. We note that the type is `VARCHAR`.
3. Perform the query:

```
select p_partkey, p_name
from ora.part
where p_name < 'an';
```
4. Collect the query access plan.
5. Analyze the access plan.
6. Set server option `COLLATING_SEQUENCE` to `Y`, this enables pushdown of sorting, grouping, and inequality comparisons.

**Important:** You need to be careful in enabling this option, otherwise, the result of inequality predicates might be wrong.

Please check the national language settings of your data source and the federated server to make sure they have the same collating sequence.

7. Perform the query again.
8. Collect the query access plan.
9. Analyze the access plan.

### 10.2.2 COLLATING\_SEQUENCE set to N (default)

By default, the server option COLLATING\_SEQUENCE is set to the value N. This value tells the optimizer *not to pushdown* any functionality to data sources that involve sorting, grouping, or inequality comparisons.

#### Step 1: Check the server option with Control Center

1. Expand your instance, your database and Federated Database Objects.
2. Expand the **ORACLE** wrapper and click **Servers**.
3. Right-click your Oracle server - **ORACLE9** in our case study and click **Alter**.
4. Switch to **Settings**.
5. The value needs to be either unchecked, or the value set to N.

If the value is set to Y or I, uncheck the check box at the beginning of the line, or set the value to N.

#### Step 2: Perform the query

We performed the query on the Oracle table PART.

#### Step 3: Collect the query access plan

We used **db2exfmt** to format the access plan for the query.

#### Step 4: Analyze the access plan

Please see Example 10-2 and the left picture in Figure 10-3 for the access plan for the query with COLLATING\_SEQUENCE set to N. The SHIP operator shows that the query submitted to the remote Oracle source retrieves data from all the rows of the PART table. A filter routine running subsequently on the federated server qualifies the rows coming from the data source. This operation is very expensive, especially for huge tables, as large amounts of data must be transferred to the federated server.

We note that:

- The RMQTX operator does not include a WHERE clause, so a row will be sent from the data source to the federated server for every row of the table TPCH.PART. The row width of the transmitted rows is not large; it has only two columns, P\_NAME and P\_PARTKEY.

- The filter P\_NAME<'an' is applied at the federated server. This means while a row is transmitted from the data source to the federated server for every row of the table, the rows not satisfying the criteria of the filter will be discarded by the federated server. Only the rows that satisfy the filter's criteria will be passed to the operation that receives the result from the filter. The implication is that if the result of the filter goes into a temporary table, the temporary table will not have to hold a row for every row of the table; only the rows that meet the criteria of the filter.

*Example 10-2 Query Explain with collating\_sequence set to N*

---

3) FILTER: (Filter)

Predicates:

-----

2) Residual Predicate

Relational Operator: Less Than (<)

Subquery Input Required: No

Filter Factor: 0.55429

Predicate Text:

-----

(Q1.P\_NAME < 'an')

5) SHIP : (Ship)

Arguments:

-----

RMTQTX : (Remote statement)

SELECT A0."P\_NAME", A0."P\_PARTKEY" FROM "TPCH"."PART" A0

---

### 10.2.3 COLLATING\_SEQUENCE set to Y

Now, we want to tell the optimizer that our target data source Oracle sorts characters the same way that DB2 does.

#### Step 5: Set COLLATING\_SEQUENCE to Y

Set the server option to Y by either performing the command in Example 10-3, or using the Control Center as described in “Step 1: Check the server option with Control Center” on page 324. If the server option is not yet defined, you need to use the ADD qualifier in the ALTER SERVER statement. If it is already defined, change its value with SET.

### Example 10-3 Add or Modify server option collating\_sequence

---

```
-- Add server server options
ALTER SERVER "ORACLE9" OPTIONS (ADD COLLATING_SEQUENCE 'Y' )

-- Change server options
ALTER SERVER "ORACLE9" OPTIONS (SET COLLATING_SEQUENCE 'Y' )
```

---

### Step 6: Perform the query again

Having set the server option to Y, we performed the query again.

### Step 7: Collect query access plan

Again, we created the Access plan for the query.

### Step 8: Analyze access plan

DB2 pushes down the query to the data source, including the predicate on the column p\_name, as shown in Example 10-4. The RPAD is an Oracle SQL function, which in this case returns the value of P\_NAME *right-padded* with blanks to the length of 55 bytes.

### Example 10-4 Query Explain with collating\_sequence set to Y

---

```
3) SHIP : (Ship)

Arguments:
-----
RMTQTX : (Remote statement)
        SELECT AO."P_PARTKEY", AO."P_NAME" FROM "TPCH"."PART" AO WHERE
        (RPAD(AO."P_NAME",55,' ') < RPAD('an',55,' '))
```

---

Figure 10-3 gives a visual description of both queries performed during the example, using the **Create Access Plan** function in Command Center.

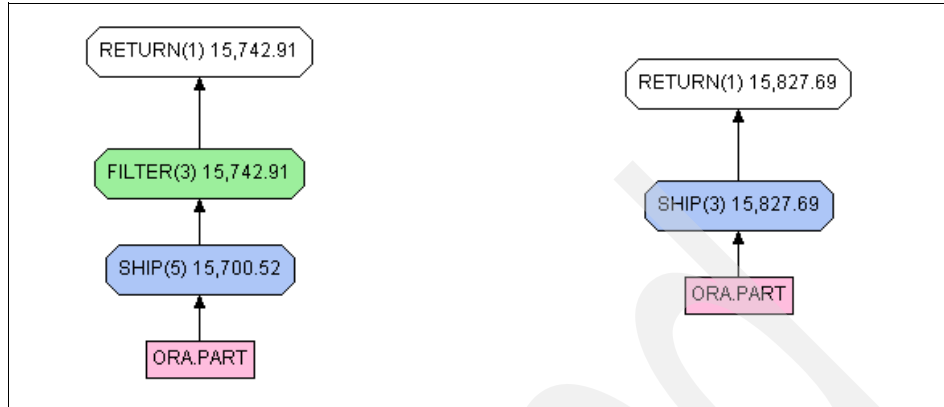


Figure 10-3 Visual Explain on collating\_sequence example

### 10.2.4 Conclusion on collating\_sequence

Using data sources that sort characters and numbers in a different way from DB2 can result in expensive operations. If you perform comparisons, sorts, or group functions on very large tables, data from potentially all rows of the table has to be transferred to the federated server. Check this option for your servers, and perform access plans for your queries to tune your environment.

## 10.3 Oracle server option varchar\_no\_trailing\_blanks

This chapter deals with the topic of trailing blanks in tables with VARCHAR columns. Trailing blanks are handled in two basically different ways by the database systems considered here. The first method, which is used by DB2 as well as Informix and SQL Server, ignores trailing blanks in VARCHAR columns in predicates, but includes these invisible characters at the end in counting functions. Oracle has some unique behavior in dealing with trailing blanks. The varchar\_no\_trailing\_blanks server option controls this behavior.

Throughout the example we used the select statement shown in Example 10-5.

#### Example 10-5 Query statement

---

```
select p_partkey, p_name, length (p_name) AS LENGTH
from ora.part
where p_name = 'metallic lime cyan forest burlywood';
```

---

### 10.3.1 Trailing blanks with DB2, Informix, and SQL Server

First, we discuss the behavior of DB2, Informix Dynamic Server, and Microsoft SQL Server with trailing blanks in VARCHAR columns.

Example 10-6 uses a SELECT statement that includes a WHERE condition comparing a VARCHAR column with a literal string. Initially, the actual qualifying string in the table has no trailing blanks. Next, we updated the table to insert a trailing blank into the qualifying string, and executed the statement again. We compared the results of both executions.

*Example 10-6 Trailing blanks with DB2*

```
db2 \  
"select p_partkey, p_name, length (p_name) AS LENGTH \  
from db2dat.part \  
where p_name = 'metallic lime cyan forest burlywood'"
```

P_PARTKEY	P_NAME	LENGTH
135	metallic lime cyan forest burlywood	35

```
db2 \  
> "update db2dat.part \  
> set p_name = p_name || ' ' \  
> where p_partkey = 135"
```

```
db2 \  
"select p_partkey, p_name, length (p_name) AS LENGTH \  
from db2dat.part \  
where p_name = 'metallic lime cyan forest burlywood'"
```

P_PARTKEY	P_NAME	LENGTH
135	metallic lime cyan forest burlywood	36

When we compare a VARCHAR column with a literal, DB2, Informix and SQL Server do not include trailing blanks in their comparison. For this reason, the second execution of the select statement returns the same value as the first time, even though the length of this string has increased by the blank we added.

## 10.3.2 Trailing blanks with Oracle

In comparison to the previous section, Oracle deals with trailing blanks in a different way. See Example 10-7.

*Example 10-7 Trailing blanks with Oracle*

```
SQL> r
  1 select p_partkey, p_name, length (p_name) length from part
  2* where p_name = 'metallic lime cyan forest burlywood'
```

P_PARTKEY	P_NAME	LENGTH
135	metallic lime cyan forest burlywood	35

```
SQL> update part set p_name = p_name || ' '
  2 where p_partkey=135;
```

1 row updated.

```
SQL> select p_partkey, p_name, length (p_name) length from part
  2 where p_name = 'metallic lime cyan forest burlywood'
  3 ;
```

**no rows selected**

```
SQL> select p_partkey, p_name, length (p_name) length from part
  2 where p_partkey=135;
```

P_PARTKEY	P_NAME	LENGTH
135	metallic lime cyan forest burlywood	36

We performed the same query as in 10.3.1 “Trailing blanks with DB2, Informix, and SQL Server” on page 328, added a trailing blank to our VARCHAR column, and repeated the query again. No row is returned, because the literal string in the query no longer exactly matches the data in the VARCHAR column. Oracle includes the trailing blank in the comparison operation. Oracle does not pad blanks when comparing varying length character strings, like DB2 does.

## 10.3.3 VARCHAR\_NO\_TRAILING\_BLANKS set to N

By default this server option is set to N. We performed our example query statement again, but this time, we referenced the nickname to the Oracle Part table. The execution plan for the query shows that the statement listed in Example 10-8 is shipped to the Oracle data source.

*Example 10-8 Query shipped to Oracle*

---

```
SELECT A0."P_NAME", A0."P_PARTKEY"  
FROM "TPCH"."PART" A0  
WHERE (RTRIM(A0."P_NAME") = 'metallic lime cyan forest burlywood')
```

---

The query shipped to the Oracle data source contains a modified WHERE condition. The federated server adds the function *RTRIM* for varchar columns in comparison operations, so that character comparisons will disregard trailing blanks.

**Important:** When VARCHAR\_NO\_TRAILING\_BLANKS is set to N, if you create an index on the varchar column (p\_name in our example), Oracle will not be able to use the index in this case.

### 10.3.4 VARCHAR\_NO\_TRAILING\_BLANKS set to Y

We set the server option for our Oracle server with the statement shown in Example 10-9. If this option is set for the first time, use ADD instead of SET, or set the option using the Control Center.

*Example 10-9 Set server option*

---

```
ALTER SERVER "ORACLE9" OPTIONS (SET VARCHAR_NO_TRAILING_BLANKS 'Y' );
```

---

We performed our sample query statement again and created the access plan for it. The statement listed in Example 10-10 is shipped to the Oracle data source.

*Example 10-10 Query shipped to Oracle*

---

```
SELECT A0."P_NAME", A0."P_PARTKEY"  
FROM "TPCH"."PART" A0  
WHERE (A0."P_NAME" = 'metallic lime cyan forest burlywood')
```

---

The query shipped to the Oracle data source now contains the original WHERE clause. The federated server leaves predicates involving VARCHAR columns unchanged. This means that only VARCHAR column values that exactly match the literal string in the predicate will qualify and be returned. VARCHAR column values that include trailing blanks will not be returned unless the literal string also contains trailing blanks.



**Attention:** Once the server option `VARCHAR_NO_TRAILING_BLANKS` is set to `Y`, be aware that this affects all `VARCHAR` columns of all nicknames defined on that server. If any `VARCHAR` column values on the underlying Oracle tables contain trailing blanks, they will not match literal string comparisons that do not include trailing blanks, and you may get wrong results for your queries. If you do not want to set this option to `Y` for all nicknames defined at the entire server level, you can set it individually on a per-column basis. It is much safer to set `VARCHAR_NO_TRAILING_BLANKS` for only those nickname column of type `varchar` that you know do not have trailing blanks, with the statement:

```
ALTER NICKNAME GK ALTER COLUMN GK_NAME OPTIONS (ADD  
VARCHAR_NO_TRAILING_BLANKS 'Y')
```



## Using data type mappings

In this chapter we describe how to define data type mappings so that remote data sources correspond correctly to DB2 data types. We show that correct data type mappings can have a significant effect on query performance.

The chapter is structured as follows:

- ▶ Overview
- ▶ Step 1: Explicit cast on nickname column
- ▶ Step 2: Accommodate the default mapping
- ▶ Step 3: Altering local column data type

## 11.1 Overview

Data types from remote data sources must correspond to DB2 data types. An appropriate mapping enables the federated server to retrieve data from the data source. DB2 Information Integrator supplies a set of default data type mappings, for example:

- ▶ Oracle type FLOAT maps by default to the DB2 type DOUBLE.
- ▶ Oracle type DATE maps by default to the DB2 type TIMESTAMP.
- ▶ DB2 for z/OS type DATE maps by default to the DB2 type DATE.

If you want to map data types in a different way than the default mapping, you need to create alternative data type mappings. In order to use an alternative data type mapping when you create a nickname, you must create this mapping in advance of creating the nickname. If you create your nickname first, you may set the appropriate mapping later by:

- ▶ Altering the nickname
- ▶ Changing default mapping types and recreating the nickname

For non relational data sources, you cannot override existing data type mappings or create new mappings.

We show different ways to deal with data type mapping in the following sections. We use the same query to show three different ways to map an Oracle date column to the DB2 date and timestamp.

Example 11-1 shows the query we will consider. This query runs against local tables on the DB2 database. In the following sections, we want to access the same table on our Oracle data source through DB2 Information Integrator and want to look specifically at the data type conversion issues in the WHERE clause.

### *Example 11-1 Original query*

---

```
SELECT
    L_RETURNFLAG,
    L_LINESTATUS,
    SUM(L_QUANTITY) AS SUM_QTY,
    SUM(L_EXTENDEDPRI) AS SUM_BASE_PRICE,
    SUM(L_EXTENDEDPRI * (1-L_DISCOUNT)) AS SUM_DISC_PRICE,
    SUM(L_EXTENDEDPRI * (1-L_DISCOUNT) * (1+L_TAX)) AS SUM_CHARGE,
    AVG(L_QUANTITY) AS AVG_QTY,
    AVG(L_EXTENDEDPRI) AS AVG_PRICE,
    AVG(L_DISCOUNT) AS AVG_DISC,
    COUNT(*) AS COUNT_ORDER
FROM
    LINEITEM
WHERE
```

```

        L_SHIPDATE <= DATE ('1998-12-01') - 90 DAYS
GROUP BY
    L_RETURNFLAG,
    L_LINESTATUS
ORDER BY
    L_RETURNFLAG,
    L_LINESTATUS;

```

---

## 11.2 Step 1: Explicit cast on nickname column

We want to execute the same query, this time against nicknames to remote Oracle tables. One possibility is to execute the query listed in Example 11-2. Notice the WHERE condition in the query. In Oracle, the column L\_SHIPDATE is stored as type DATE. However, DB2 Information Integrator maps this data type to DB2 as a TIMESTAMP by default. This is done in order not to lose any data, since the Oracle DATE type contains a subset of DB2 TIMESTAMP information, but more than just the DB2 DATE information. So, we added the cast DATE (L\_SHIPDATE) to the comparison, to establish comparison of corresponding data types. If we did not use the DATE cast function in the WHERE clause, we would get a data type mismatch error. Without the DATE function, Information Integrator would produce TIMESTAMP values from the L\_SHIPDATE column; these values cannot be used in comparison with the DATE value produced from the expression: DATE ('1998-12-01') - 90 DAYS

*Example 11-2 Casting the remote column l\_shipdate to DATE*

---

```

SELECT
    L_RETURNFLAG,
    L_LINESTATUS,
    SUM(L_QUANTITY) AS SUM_QTY,
    SUM(L_EXTENDEDPRICE) AS SUM_BASE_PRICE,
    SUM(L_EXTENDEDPRICE * (1-L_DISCOUNT)) AS SUM_DISC_PRICE,
    SUM(L_EXTENDEDPRICE * (1-L_DISCOUNT) * (1+L_TAX)) AS SUM_CHARGE,
    AVG(L_QUANTITY) AS AVG_QTY,
    AVG(L_EXTENDEDPRICE) AS AVG_PRICE,
    AVG(L_DISCOUNT) AS AVG_DISC,
    COUNT(*) AS COUNT_ORDER
FROM
    ORA.LINEITEM
WHERE
    DATE (L_SHIPDATE) <= DATE('1998-12-01') - 90 DAYS
GROUP BY
    L_RETURNFLAG,
    L_LINESTATUS
ORDER BY
    L_RETURNFLAG,

```

L\_LINESTATUS;

---

Looking at the query pushed down to Oracle in Example 11-3, we figured out how DB2 Information Integrator transforms this CAST into Oracle syntax. Actually, this CAST is accomplished by a conversion to CHAR and then to DATE format before the subsequent comparison with a DATE constant.

*Example 11-3 Query pushed down to Oracle (db2exfmt output)*

---

```
SELECT AO."L_RETURNFLAG", AO."L_LINESTATUS",
SUM( AO."L_QUANTITY"),
SUM( AO."L_EXTENDEDPRICE"),
SUM( (AO."L_EXTENDEDPRICE" * (+1.000000000000000E+000 - (AO."L_DISCOUNT")))),
SUM( ((AO."L_EXTENDEDPRICE" * (+1.000000000000000E+000 - (AO."L_DISCOUNT")) *
(+1.000000000000000E+000 + AO."L_TAX")))),
AVG( AO."L_QUANTITY"),
AVG( AO."L_EXTENDEDPRICE"),
AVG( AO."L_DISCOUNT"),
COUNT(*)
FROM "TPCH"."LINEITEM" AO
WHERE (TO_DATE(TO_CHAR(AO."L_SHIPDATE", 'YYYY-MM-DD'), 'YYYY-MM-DD') <=
TO_DATE('19980902 000000', 'YYYYMMDD HH24MISS'))
GROUP BY AO."L_RETURNFLAG", AO."L_LINESTATUS" ORDER BY 1 ASC, 2 ASC
```

---

Let us look at the performance aspect. As a data type conversion is performed on the column L\_SHIPDATE in the WHERE condition of our query, the Oracle system cannot utilize an available index on this column. Furthermore, as the query accesses our largest table with 6 million rows, not being able to use any index is very costly. Example 11-4 shows the elapsed time figures for our Step 1 query.

*Example 11-4 Performance with the explicit cast*

---

Number of rows retrieved is:	4	
Number of rows sent to output is:	4	
Prepare Time is:	0.023	seconds
Execute Time is:	126.107	seconds
Fetch Time is:	0.000	seconds
Elapsed Time is:	126.131	seconds

---

## 11.3 Step 2: Accommodate the default mapping

For our next step we eliminated the casting operation used in the previous chapter, and compare the L\_SHIPDATE column with the data type used by default for mapping, the TIMESTAMP. Let us look to Example 11-5 and at the modified WHERE condition.

*Example 11-5 Using TIMESTAMP to compare column l\_shipdate*

---

```
SELECT
    L_RETURNFLAG,
    L_LINESTATUS,
    SUM(L_QUANTITY) AS SUM_QTY,
    SUM(L_EXTENDEDPRICE) AS SUM_BASE_PRICE,
    SUM(L_EXTENDEDPRICE * (1-L_DISCOUNT)) AS SUM_DISC_PRICE,
    SUM(L_EXTENDEDPRICE * (1-L_DISCOUNT) * (1+L_TAX)) AS SUM_CHARGE,
    AVG(L_QUANTITY) AS AVG_QTY,
    AVG(L_EXTENDEDPRICE) AS AVG_PRICE,
    AVG(L_DISCOUNT) AS AVG_DISC,
    COUNT(*) AS COUNT_ORDER
FROM
    ORA.LINEITEM
WHERE
    L_SHIPDATE <= TIMESTAMP('1998-12-01-00.00.00') - 90 DAYS
GROUP BY
    L_RETURNFLAG,
    L_LINESTATUS
ORDER BY
    L_RETURNFLAG,
    L_LINESTATUS;
```

---

The query pushed down to Oracle in Example 11-6 reflects this change in the WHERE condition. Instead of converting the column L\_SHIPDATE into CHAR format and back to DATE, this column is now compared with a TIMESTAMP constant.

*Example 11-6 Query pushed down to Oracle*

---

```
SELECT AO."L_RETURNFLAG", AO."L_LINESTATUS", SUM( AO."L_QUANTITY"),
SUM( AO."L_EXTENDEDPRICE"),
SUM( (AO."L_EXTENDEDPRICE" * (+1.000000000000000E+000 - (AO."L_DISCOUNT")))),
SUM( ((AO."L_EXTENDEDPRICE" * (+1.000000000000000E+000 - (AO."L_DISCOUNT")) *
(+1.000000000000000E+000 + AO."L_TAX")))),
AVG( AO."L_QUANTITY"),
AVG( AO."L_EXTENDEDPRICE"),
AVG( AO."L_DISCOUNT"),
COUNT(*)
FROM "TPCH"."LINEITEM" AO
```

---

```
WHERE (A0."L_SHIPDATE" <= TO_TIMESTAMP('19980902 000000000000','YYYYMMDD
HH24MISSFF'))
GROUP BY A0."L_RETURNFLAG", A0."L_LINESTATUS"
ORDER BY 1 ASC, 2 ASC
```

---

What about performance? Can we expect better execution time? In this example no function in the WHERE condition is applied on the L\_SHIPDATE column. The Oracle data source is able to utilize the index for this column. Immediately, we obtained the performance increase shown in Example 11-7.

*Example 11-7 Performance accommodating the default mapping*

---

```
Number of rows retrieved is:      4
Number of rows sent to output is: 4

Prepare Time is:                0.025      seconds
Execute Time is:                76.678      seconds
Fetch Time is:                  0.000      seconds
Elapsed Time is:                76.703      seconds
```

---

## 11.4 Step 3: Altering local column data type

In the previous step, we really improved query performance. For our last experiment we wanted, of course, to keep this improvement in performance (or make it even better) and wanted to come back to the original query syntax shown in Example 11-1. We can do this by altering the data type mapping for the L\_SHIPDATE column of the ORA.LINEITEM nickname from its default setting of TIMESTAMP to our custom setting DATE as shown in Example 11-8. Now we are able to run the query against the Oracle nickname using the original WHERE clause.

*Example 11-8 Using data type mapping and original query syntax*

---

```
ALTER NICKNAME ORA.LINEITEM ALTER COLUMN l_shipdate LOCAL TYPE date;

SELECT
  L_RETURNFLAG,
  L_LINESTATUS,
  SUM(L_QUANTITY) AS SUM_QTY,
  SUM(L_EXTENDEDPRICE) AS SUM_BASE_PRICE,
  SUM(L_EXTENDEDPRICE * (1-L_DISCOUNT)) AS SUM_DISC_PRICE,
  SUM(L_EXTENDEDPRICE * (1-L_DISCOUNT) * (1+L_TAX)) AS SUM_CHARGE,
  AVG(L_QUANTITY) AS AVG_QTY,
  AVG(L_EXTENDEDPRICE) AS AVG_PRICE,
```



```

        AVG(L_DISCOUNT) AS AVG_DISC,
        COUNT(*) AS COUNT_ORDER
FROM
    ORA.LINEITEM
WHERE
    L_SHIPDATE <= DATE('1998-12-01') - 90 DAYS
GROUP BY
    L_RETURNFLAG,
    L_LINESTATUS
ORDER BY
    L_RETURNFLAG,
    L_LINESTATUS;

```

---

The query pushed down to Oracle in Example 11-9 reflects the new data type mapping. Without any casting operation, DB2 Information Integrator uses the new data type mapping on column L\_SHIPDATE between Oracle and DB2.

*Example 11-9 Query pushed down to Oracle (db2exfmt output)*

```

SELECT AO."L_RETURNFLAG", AO."L_LINESTATUS",
SUM( AO."L_QUANTITY"),
SUM( AO."L_EXTENDEDPRICE"),
SUM( (AO."L_EXTENDEDPRICE" * (+1.000000000000000E+000 - (AO."L_DISCOUNT")))),
SUM( ((AO."L_EXTENDEDPRICE" * (+1.000000000000000E+000 - (AO."L_DISCOUNT")) *
(+1.000000000000000E+000 + AO."L_TAX")))),
AVG( AO."L_QUANTITY"),
AVG( AO."L_EXTENDEDPRICE"),
AVG( AO."L_DISCOUNT"),
COUNT(*)
FROM "TPCH"."LINEITEM" AO
WHERE (AO."L_SHIPDATE" <= TO_DATE('19980902 000000','YYYYMMDD HH24MISS'))
GROUP BY AO."L_RETURNFLAG", AO."L_LINESTATUS"
ORDER BY 1 ASC, 2 ASC

```

---

Finally, we look at performance again. We expected similar performance as in Step 2, as we see similar queries pushed down to Oracle. The difference is that in Step 2, Oracle compares the L\_SHIPDATE column with a TIMESTAMP literal, while Step 3 compares it with the actual type DATE.

We even get better performance for Step 3 than for Step 2, as shown in Example 11-10.

*Example 11-10 Performance with data type mapping*

```

Number of rows retrieved is:      4
Number of rows sent to output is: 4

```

```

Prepare Time is:      0.014      seconds

```

Execute Time is:	71.220	seconds
Fetch Time is:	0.000	seconds
Elapsed Time is:	71.234	seconds

-----

---

# Using function mappings

In this chapter we describe how user function mapping options can influence the performance of a query accessing relational data sources.

The chapter is structured into:

- ▶ Overview
- ▶ Mapping user defined functions

## 12.1 Overview

DB2 Information Integrator supplies default mappings between existing built-in relational data source functions, and their built-in DB2 counterpart functions. These default function mappings are implemented in the wrappers.

For non relational data sources, you cannot override the existing function mappings or create new mappings.

Basically, you can create a function mapping if there is no default mapping available. There are several reasons for creating function mappings:

- ▶ No DB2 function corresponding to a remote data source function is available.
- ▶ A corresponding DB2 function is available, but with a specification that is different from that of its remote counterpart.
- ▶ A new built-in function becomes available at the data source.
- ▶ A new user defined function becomes available at the data source.

The DB2 catalog view for function mappings is SYSCAT.FUNCMAPPINGS

Function mappings are one of several inputs to the pushdown analysis performed by the query optimizer. If your query includes a function or operation, the optimizer evaluates if this function can be sent to the data source for processing. If the data source has the corresponding function available, processing of this function can be pushed down, and it will help to improve performance.

A DB2 function template can be used to force the federated server to invoke a data source function. Function templates do not have executable code, but they can be the object of a function mapping. After creating a DB2 function template, you need to create the actual function mapping between the template and the corresponding data source function.

The CREATE FUNCTION MAPPING statement is very powerful, and gives you considerable control over the scope of the mapping. For example, you can:

- ▶ Create a function mapping for all data sources of a specific type, for instance all Informix data sources.
- ▶ Create a function mapping for all data sources of a specific type and version, for instance all Oracle 9 data sources.
- ▶ Create a function mapping for all data source objects located on a specific server.
- ▶ Disable a default function mapping. Default function mappings cannot be dropped.

For more information, please refer to the *IBM DB2 Information Integrator Data Source Configuration Guide Version 8*, available as softcopy from the Web site:

<http://www.ibm.com/software/data/integration/solution>

## 12.2 Mapping user defined functions

In this example, first we created a stored function on a remote Oracle server. Then we established a mapping between DB2 and Oracle for this function, which enabled us to use this functionality from DB2.

As previously explained, function mappings enable the DB2 user to use functionality that resides on different data sources.

In Example 12-1, we created a simple function directly on our Oracle server. We executed this CREATE FUNCTION statement from within SQL\*Plus.

### *Example 12-1 Oracle function*

---

```
create or replace function
avg_balance (v_nation IN Number) return Number IS
  v_acctbal Number;
BEGIN
  select avg(s_acctbal)
    into v_acctbal
    from supplier
    where s_nationkey = v_nation;
  return v_acctbal;
EXCEPTION when no_data_found then
  return 0;
when others then
  return 0;
END;
/
```

---

In the example, we were connected to Oracle as user tpch, so the function avg\_balance is created in schema tpch. This information is needed when the function mapping is created to this function in Information Integrator.

Within the DB2 command window, we opened a passthru session to our Oracle server, and tested the function directly on the data source as shown in Example 12-2.

### Example 12-2 Test Oracle function

---

```
set passthru oracle9;  
select n_nationkey, n_name, avg_balance (n_nationkey) from nation;  
set passthru reset;
```

#### Query result:

=====

```
select n_nationkey, n_name, avg_balance (n_nationkey) from nation
```

N_NATIONKEY	N_NAME	AVG_BALANCE(N_NATIONKEY)
0	ALGERIA	4316.85626
1	ARGENTINA	4393.06562
2	BRAZIL	4407.13975
3	CANADA	4955.39374
4	EGYPT	4349.71337
5	ETHIOPIA	4727.04834
6	FRANCE	4610.41087
7	GERMANY	4288.12379
8	INDIA	4478.81583
9	INDONESIA	4642.87953
10	IRAN	4633.44812
11	IRAQ	4618.4208
12	JAPAN	4788.34297
13	JORDAN	4304.1516
14	KENYA	4419.07229
15	MOROCCO	4710.59869
16	MOZAMBIQUE	4106.16401
17	PERU	4684.25235
18	CHINA	4206.45838
19	ROMANIA	4576.54819
20	SAUDI ARABIA	4350.06793
21	VIETNAM	4309.61015
22	RUSSIA	4624.56958
23	UNITED KINGDOM	4612.20215
24	UNITED STATES	4666.34239

---

In Example 12-3 we show the statements to create the mapping for our Oracle function created earlier. Two steps are necessary to perform the mapping:

1. Create the DB2 function template that defines the input parameters and the returned value(s).
2. Create the function mapping, which establishes the actual link between the Oracle data source function and the DB2 template function.

If you need to drop the function mapping definitions, you need to drop the function mapping first, and then drop the function template second.

*Example 12-3 Create function mapping*

---

```
drop function mapping ora9_avg_balance;
drop function avg_balance (decimal ());

-- Step 1
create function avg_balance (decimal (10))
returns decimal (10)
as template
deterministic
no external action;

-- Step 2
create function mapping ora9_avg_balance
for avg_balance (decimal (10))
server oracle9
options(remote_name 'tpch.avg_balance');
```

---

More information on the steps in Example 12-3:

1. The federated server recognizes a data source function when there is a mapping between the data source function and a counterpart function at the federated database. You can create a *function template* to act as the counterpart when no local DB2 counterpart exists.

**Important:** You should consider specifying the DETERMINISTIC and NO EXTERNAL ACTION clauses in the create function ... as template statement, if applicable. Otherwise, restrictions will be imposed on the SQL operations that are supported with this function template.

2. The CREATE FUNCTION MAPPING statement is used to create a mapping between the function template and a data source function. The mapping can associate the template with a function at a specified data source. In our example, we mapped the Oracle function avg\_balance to the DB2 template avg\_balance.

On our federated server we performed the query shown in Example 12-4. The function avg\_balance is now known to the DB2 server, even though no local implementation is available.

*Example 12-4 Function mapping test*

---

```
select n_nationkey, n_name, avg_balance (n_nationkey) from
ora.nation;
```

N_NATIONKEY	N_NAME	3
-----		
0.	ALGERIA	4316.
1.	ARGENTINA	4393.
2.	BRAZIL	4407.
3.	CANADA	4955.
4.	EGYPT	4349.
5.	ETHIOPIA	4727.
6.	FRANCE	4610.
7.	GERMANY	4288.
8.	INDIA	4478.
9.	INDONESIA	4642.
10.	IRAN	4633.
11.	IRAQ	4618.
12.	JAPAN	4788.
13.	JORDAN	4304.
14.	KENYA	4419.
15.	MOROCCO	4710.
16.	MOZAMBIQUE	4106.
17.	PERU	4684.
18.	CHINA	4206.
19.	ROMANIA	4576.
20.	SAUDI ARABIA	4350.
21.	VIETNAM	4309.
22.	RUSSIA	4624.
23.	UNITED KINGDOM	4612.
24.	UNITED STATES	4666.

---

Example 12-5 shows the query, which is pushed down to the Oracle data source. As the function is implemented on Oracle, it is very logical that the execution takes place at the Oracle data source.

*Example 12-5 Function mapping test - Query pushed down to Oracle*

---

```
SELECT A0."N_NATIONKEY", A0."N_NAME",
tpch.avg_balance(A0."N_NATIONKEY")
FROM "TPCH"."NATION" A0
```

---



## Major server options with non relational data sources

In this chapter we describe how server options can influence the performance of a query accessing non relational data sources.

The chapter is structured in:

- ▶ Overview
- ▶ Pushdown with Excel data sources
- ▶ Table-structured files parameter - Sorted

## 13.1 Overview

Non relational data sources such as Excel spreadsheets, XML tagged files, or table-structured files are accessed by their respective wrappers. This wrapper opens the data source file directly out of the filesystem. We discuss server and nickname options of interest when dealing with Excel spreadsheets and table-structured files.

We used the ODBC wrapper to connect to our Excel data source. There is an Excel wrapper available within DB2 Information Integrator, but this wrapper is available only in the Windows environment. The ODBC wrapper we used is able to connect to various different data sources, of which Excel is just one. Because the ODBC wrapper is able to access data sources with widely varying capabilities to manage data, the default options for a server based on the ODBC wrapper are set assuming lowest common denominator functionality. In particular, the `PUSHDOWN` option for a server based on the ODBC wrapper is set to `N`. When using the ODBC wrapper to access any relational source, such as Microsoft SQL Server, it makes sense to set `PUSHDOWN` to `Y`, indicating that the optimizer should consider pushing down simple operations to the remote data source.

**Tip:** Remember to set `PUSHDOWN` to `Y` for a server based on the ODBC wrapper before accessing a data source based on any DBMS environment.

Our first example shows that even a non relational source such as Excel, which is accessed through ODBC, can benefit from setting the server option `PUSHDOWN` to `Y`.

**Recommendation:** It is recommended that the ODBC wrapper only be used when there is not a wrapper for a specific type of data source. For instance, the Oracle wrapper should be used with Oracle; not the ODBC wrapper. And the SQL Server wrapper should be used with SQL Server, even though that wrapper uses ODBC, rather than using the ODBC wrapper. The wrappers created for specific types of data sources contain intelligence about the data source such as function mappings and attributes that tell PDA and the optimizer what SQL operations and functions the data source supports. These wrappers can be expected to deliver much better performance with their name type of data sources than could be achieved with the ODBC wrapper.

Servers defined using the table-structured files or XML wrappers do not have any server options. In our second example, we describe one of the table-structured files nickname options, the `SORTED` parameter, and show how its use to access sorted table-structured files results in a performance improvement.

## 13.2 Pushdown with Excel data sources

By default, the server option `PUSHDOWN` for our Excel data source is disabled. (set to `N`). The example uses the query shown in Example 13-1, which we executed once with the `PUSHDOWN` option disabled, and then again with `PUSHDOWN` enabled.

The query asks for the name of the nation with `nationkey` equal to 13, and for the name of the region to which this nation belongs.

*Example 13-1 Simple query*

---

```
SELECT n_name, r_name
FROM xls.nation$
     , xls.region$
WHERE n_regionkey = r_regionkey
      AND n_nationkey=13;
```

---

### 13.2.1 `PUSHDOWN` set to `N` (default)

The pushdown server option for ODBC wrappers is set to `N` by default. The access plan for this case, depicted in Figure 13-1, shows two `SHIP` operators. DB2 Information Integrator does not pushdown the join operation to the Excel data source; rather, it performs two independent queries. The local filtering predicate on `n_nationkey` is also not pushed down to the remote source.

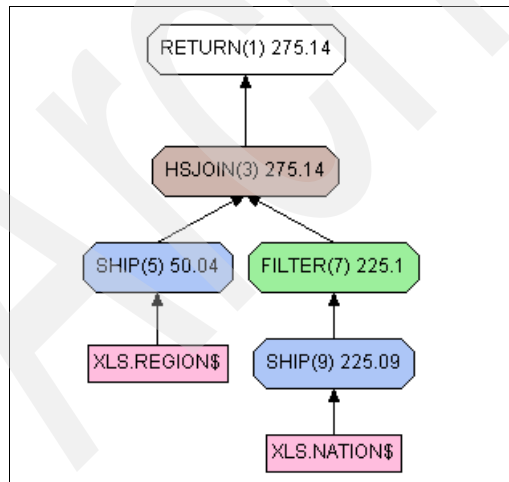


Figure 13-1 Access plan - Pushdown set to `N`

The queries that are pushed down to Excel are shown in Example 13-2. DB2 Information Integrator performs two queries that simply retrieve the needed columns from *all* rows of *both* tables.

*Example 13-2 No pushdown - Queries shipped to the server*

---

```
SELECT A0."R_REGIONKEY", A0."R_NAME"
FROM "Region$" A0

SELECT A0."N_REGIONKEY", A0."N_NATIONKEY", A0."N_NAME"
FROM "Nation$" A0
```

---

The result sets of both queries are transferred to the federated server. DB2 applies the filter condition on the returned NATION rows, and joins them with the returned REGION rows through hash join.

## 13.2.2 PUSHDOWN set to Y

We now enable the PUSHDOWN server option as shown in Example 13-3.

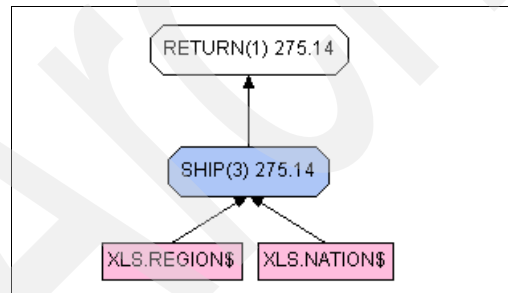
*Example 13-3 Set PUSHDOWN to Y*

---

```
ALTER SERVER "ALLSRV" OPTIONS (ADD PUSHDOWN 'Y' ) ;
```

---

The access plan looks much simpler now. The optimizer decides to pushdown the join including the predicate to the ODBC server. DB2 returns the result set of the statement sent to Excel directly to the user.



*Figure 13-2 Access plan - Pushdown set to Y*

The query pushed down to the data source is shown in Example 13-4.

*Example 13-4 Pushdown - Query shipped to the server*

---

```
SELECT A0."N_NAME", A1."R_NAME"
FROM "Nation$" A0, "Region$" A1
WHERE (A0."N_NATIONKEY" = +1.300000000000000E+001)
```

---

AND (A0."N\_REGIONKEY" = A1."R\_REGIONKEY")

---

## 13.3 Table-structured files parameter - Sorted

Using this parameter, you are able to tell the table-structured files wrapper whether your table-structured files data source is sorted in ascending order. By “sorted” in this context we mean that the values in the first field of the file:

- ▶ Are unique in every record
- ▶ Increase in value from one record to the next
- ▶ From the first record of the file to the last of the file

Not all values in the collating sequence need to be used, but, as stated above, the value in the first field of any record in the file needs to be greater than the value in the first field of the record before it in the file.

Throughout this section, we will consider the query described in Example 13-5.

*Example 13-5 Query used for sorted table-structured file*

---

```
SELECT c_name
FROM FLAT.customer
WHERE c_custkey < 10;
```

---

Our table-structured file contains 150,000 rows of customer information. The column C\_CUSTKEY is the first column, and the file is sorted by this column.

With this example, we want to show how you can dramatically improve query performance with the table-structured files wrapper by using sorted table-structured files.

We will see that other operators can also take advantage of having a sorted file, including the *greater than* (>) and *equal* (=) operators.

### 13.3.1 Nickname parameter SORTED set to N (default)

This is the default setting. You set the nickname option with a statement like the one listed in Example 13-6.

**Note:** If it is the first time you execute this command, you need to type ADD instead of SET.

*Example 13-6 Set parameter sorted to N*

---

```
ALTER NICKNAME "FLAT"."CUSTOMER" OPTIONS (SET SORTED 'N' );
```

---

Part of the **db2batch** output for execution of our query is shown in Example 13-7. Since the table-structured files wrapper cannot assume that the file is sorted, it has no option but to read the entire file in evaluating the predicate on C\_CUSTKEY.

*Example 13-7 Performance details*

---

Number of rows retrieved is:	9	
Number of rows sent to output is:	9	
Prepare Time is:	0.000	seconds
Execute Time is:	2.298	seconds
Fetch Time is:	0.000	seconds
Elapsed Time is:	2.298	seconds

---

-----

---

### 13.3.2 Nickname parameter SORTED set to Y

Now we will use the SORTED option for the nickname FLAT.CUSTOMER. To enable this option, we executed the command listed in Example 13-8.

**Note:** If it is the first time you execute this command, you need to type **ADD** instead of **SET**.

*Example 13-8 Set parameter sorted to Y*

---

```
ALTER NICKNAME "FLAT"."CUSTOMER" OPTIONS (SET SORTED 'Y' )
```

---

We executed the same query again, and found a huge difference in execution time for the query shown in Example 13-9. In fact, the wrapper just has to read 10 records in order to return 9 rows to the SQL statement. The same algorithms are used for the greater than (>), lower than (<), greater than or equals (>=), lower than or equals (<=), equals (=), and not equals (<>) operators.

*Example 13-9 Performance details*

---

Number of rows retrieved is:	9	
Number of rows sent to output is:	9	
Prepare Time is:	0.004	seconds
Execute Time is:	0.004	seconds
Fetch Time is:	0.000	seconds

---

### 13.3.3 Conclusion

Figure 13-3 shows the two access plans. The left access plan describes basically all queries with SORTED set to N. The entire table-structured file is read and the WHERE condition (FILTER) is applied afterwards by DB2.

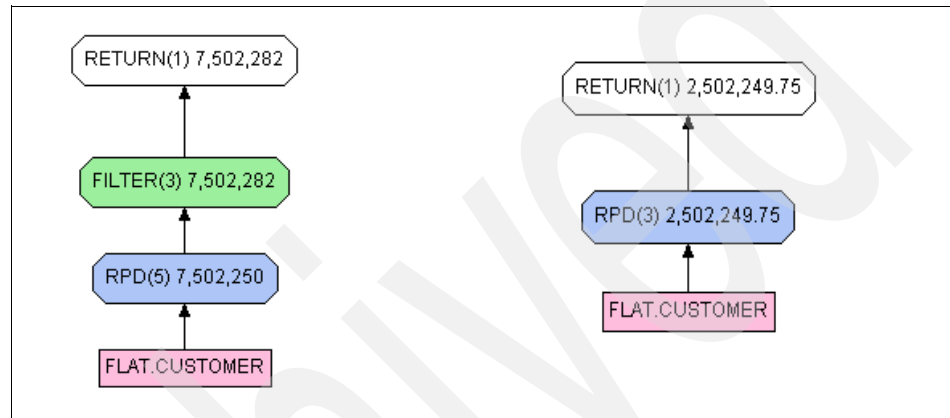


Figure 13-3 `SORTED = 'N'` versus `SORTED = 'Y'`

For a WHERE condition against the key\_col with a single operator such as:

- ▶ Lower than (<)
- ▶ Greater than (>)
- ▶ Equals (=)
- ▶ Greater than or equals (>=)
- ▶ Lower than or equals (<=)
- ▶ Not equals (<>)

The right access plan is used, if the `SORTED` nickname option is set to `Y`. Because the optimizer knows that the file is sorted, the wrapper is able to skip reading the entire file. It can avoid reading rows, which will surely not qualify for the filtering predicate.





## Using index specifications

In this chapter we describe two situations in which creating an index specification on a nickname can improve the performance of a query accessing relational data sources.

The chapter is structured as:

- ▶ Overview
- ▶ Table acquires new index after nickname creation
- ▶ Nicknames over remote views

## 14.1 Overview

When you create a nickname, DB2 Information Integrator retrieves information about the indexes defined on the table at the remote source. This information is stored in DB2's global catalog as an attribute of the nickname, and is used during query optimization.

Index information for a nickname will not be retrieved if:

- ▶ The nickname for a table has no indexes.
- ▶ The nickname is for a remote view, Informix synonym, table structured file, Excel spreadsheet, or XML tagged file. For the views and Informix synonyms, these objects do not have index information in the data source catalog, though the tables referred by the view or synonym may have indexes.
- ▶ The remote index has a column with more than 255 bytes, or a total index key length with more than 1024 bytes.
- ▶ The remote index is on LOB columns.

Another possible case in which index information for a nickname will be missing from the catalog is when you create a new index on a remote object after creating the nickname for the object. DB2 is not notified of the change, and has no way of knowing that it needs to update its index information to include the new index.

To notify DB2 Information Integrator of the existence of a missing index for a nickname, you can create a so-called *index specification*. The specification registers the existence of the new index on a remote table in DB2's global catalog. The information recorded includes the keys that comprise the index, but does not include any statistical information. Thus, creating an index specification does not record as much data about the remote index as would be obtained by dropping and recreating the nickname. If you want to ensure that index information for nicknames includes all available statistical data, you need to follow the steps described in Chapter 9, "Nickname statistics" on page 305.

Similarly, when a nickname is created for a remote view, the federated server is unaware of the underlying tables (and their indexes) from which the view was generated. An index specification can be used to tell the optimizer about indexes on the underlying tables of a remote view, which may help it choose better access paths for queries involving the nickname to the remote view.

In either case, you supply the necessary index information to the global catalog using the CREATE INDEX... SPECIFICATION ONLY statement. We examine two examples where index specifications are helpful.

## 14.2 Table acquires new index after nickname creation

We go through an example in which the PART table on our remote DB2 data source acquires a new index. This new index needs to be reflected at our federated server, where a nickname to the PART table has already been defined.

Assume that:

- ▶ There is a nickname DB2DAT.PART on the federated server that points to the remote data source table DATDB.PART.
- ▶ We created a new index on our data source table DATDB.PART, as shown in Example 14-1.

*Example 14-1 Create new index on table part on DB2 DATDB data source*

---

```
db2 "connect to datdb user db2dat64 using db2dat64"
db2 "create index ni_part on datdb.part (p_name asc)"
db2 "runstats on table datdb.part for index db2dat64.ni_part"
```

---

- ▶ The nickname needs an update about this new index the remote PART table. There are two ways to make this happen:
  - Drop and recreate the nickname DB2DAT.PART, so that all current information about indexes and statistics will be transferred again from the data source to the federated server catalog tables for the nickname. This is simple, but has the side effect that any views and packages depending on the nickname are invalidated, and all GRANTS on the nickname are lost.
  - Create an index specification on the nickname, as shown in Example 14-2, so that the new index on P\_NAME is known to the federated server.

*Example 14-2 Add information about new index on data source table to nickname*

---

```
db2 'create index feddb.ni_part on db2dat.part (p_name asc) specification only'
```

---

**Note:** Statistics on this index are not collected on the federated server. This is not important. What is most important to the federated server's optimizer is just to know which columns of the remote table or view are indexed, so that the optimizer can factor this into estimating the cost of pushing down a WHERE clause to the data source.

## 14.3 Nicknames over remote views

We have mentioned in the 14.1, "Overview" on page 356 that nicknames created for remote views will not contain any information about indexes on the underlying (remote) tables of the remote view. Similarly, a nickname over a remote view will

have no statistical information about the remote view; it will appear to the federated server as a nickname to a remote table with the default 1000 rows! If you think it will be helpful, you need to add information about existing indexes defined on the tables underneath the remote view.

This example shows how to create an index specification for a nickname over a remote view with indexes on the underlying tables of the view. Actually, we show a little more. The example also shows another pitfall of creating nicknames over remote views, which is lack of statistical information.

We created a view on a remote source as shown in Example 14-3. First we connected to our federated server database, FEDDB. We used a passthru session to connect to our DB2 database server named DB2UDB.

*Example 14-3 Create view soldbypart on DB2 database datdb*

```
connect to datdb user db2dat64 using db2dat64;
set passthru db2udb;
-- Number of parts sold, total, by part.
create view sc07.soldbypart as
(select l_partkey, sum(l_quantity) as total_sold from
datdb.lineitem group by l_partkey);
set passthru reset;
```

Back on our federated server, we created the nickname for our view, as shown in Example 14-4.

*Example 14-4 Create nickname for view on DB2 database feddb*

```
create nickname sc07.soldbypart for db2udb.sc07.soldbypart;
```

Let us learn a little more about this nickname with Example 14-5. We wanted to know the column names and data types by issuing a DESCRIBE TABLE statement. We received the cardinality of the nickname, by counting the number of rows.

*Example 14-5 Information about our new nickname*

```
describe table sc07.soldbypart;
```

Column name	Type schema	Type name	Length	Scale	Nulls
L_PARTKEY	SYSIBM	INTEGER	4	0	No
TOTAL_SOLD	SYSIBM	DOUBLE	8	0	Yes

```
select count(*) from sc07.soldbypart;
1
```

```
-----
200000

1 record(s) selected.
```

---

Now, this nickname-over-remote-view has *no* statistics. This is a good opportunity to tell the federated server how many rows it should expect, otherwise, it will assume only 1000 rows, and will generate non-optimal access plans. In Example 14-6 we show how to perform this. We simply set the table cardinality of the nickname to the row count just determined.

*Example 14-6 update cardinality for new nickname*

---

```
update sysstat.tables set card = 200000 where tabschema = 'SC07' and tabname =
'SOLDBYPART';
```

---

We are now ready to perform the query based on our new nickname. Having statistical information in place, we will execute a join between the nickname based on the remote DB2 view, and the PART table from the Oracle data source.

We want to get the total quantity sold for all parts with container JUMBO PKG, as shown in Example 14-7.

*Example 14-7 Get total quantity sold for parts with special container*

---

```
explain plan
for
select l_partkey, total_sold
from sc07.soldbypart, ora.part
where P_CONTAINER = 'JUMBO PKG'
and p_partkey = l_partkey;
```

---

As the nickname sc07.soldbypart is based on a view, the nickname does not have any information about indexes available. For the PART nickname, the optimizer has information about the remote indexes available. We show the query execution time in Example 14-8.

*Example 14-8 Result of query in Example 14-7*

---

```
Statement number: 1

select l_partkey, total_sold from sc07.soldbypart, ora.part where
P_CONTAINER = 'JUMBO PKG' and
p_partkey = l_partkey

L_PARTKEY      TOTAL_SOLD
-----
1 8.600000000000000e+02
```

```

28 5.980000000000000e+02
91 5.560000000000000e+02
...

```

```

Number of rows retrieved is:    5013
Number of rows sent to output is:  5013

```

```

Prepare Time is:      0.013      seconds
Execute Time is:      15.729     seconds
Fetch Time is:        0.160     seconds
Elapsed Time is:      15.902     seconds
-----

```

The execution takes nearly 16 seconds. Now, we see if we can do better. Let us look at the access plan reported in Figure 14-1.

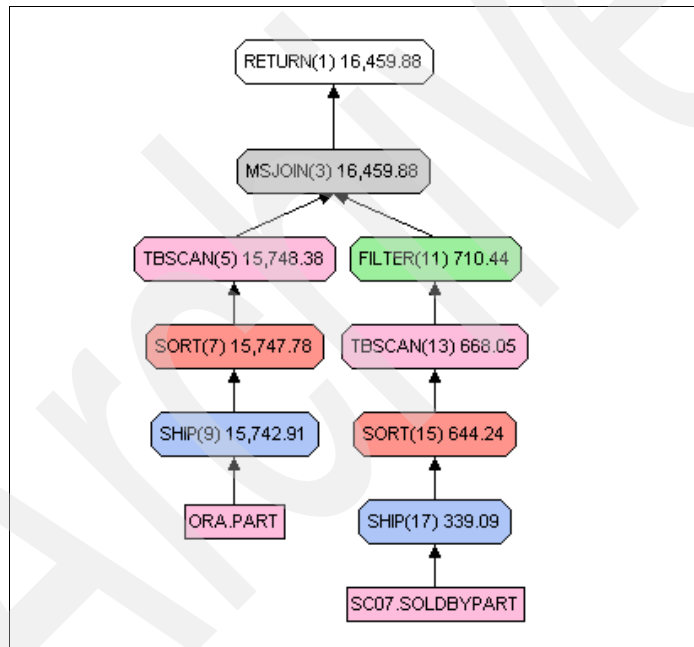


Figure 14-1 Access plan without index specification on nickname

The DB2 Federated Server sends the two select statements in Example 14-9 to the data sources. The first select statement accesses Oracle's PART table, the second select statement scans through the entire remote DB2 view.

#### Example 14-9 Shipped queries

---

```
SELECT A0."P_PARTKEY"  
FROM "TPCH"."PART" A0  
WHERE (A0."P_CONTAINER" = 'JUMBO PKG ')  
  
SELECT A0."L_PARTKEY", A0."TOTAL_SOLD"  
FROM "SC07"."SOLDBYPART" A0  
FOR READ ONLY
```

---

As we can see from the access plan in Figure 14-1 and Example 14-9, we fetched all qualifying parts for which the P\_CONTAINER column equals JUMBO PKG from the Oracle data source. From our DB2 data source, we scanned the entire view to retrieve the total quantity sold for *all* part keys. At the end, DB2 performs a merge join of the two query results.

Since the optimizer is not aware that any of the columns of the view SOLDBYPART are indexed at data source DB2UDB, it assumes that the data source will have to do a table scan to process any WHERE clauses sent to it. The optimizer compares the cost with processing the WHERE clause by doing a table scan at the data source with retrieving all the rows of the table, and applying the WHERE clause at the federated server. In many cases, the plan that fetches all the rows and applies the predicate at the federated server is estimated to have lower cost than the plan that pushes down the predicate to be processed at the data source using a table scan.

When index information is added to the nickname for SOLDBYPART indicating that the column P\_CONTAINER is indexed at the data source, the optimizer will probably assume that the data source will use this index if the WHERE clause is pushed down. Then the plan that pushes the predicate down will have a lower estimated cost than before when there was no index information for the nickname. With the index information for the nickname, the plan that pushes down the WHERE clause will probably have a lower cost than the plan that retrieves all rows and applies the predicate at the federated server.

### 14.3.1 What about utilizing an existing index?

We know that there is an index on L\_partkey on the LINEITEM table underlying the remote view. We would like to tempt the optimizer into choosing a nested join that retrieves qualifying part keys from Oracle, and probes into the remote view using the L\_partkey index. And, from our federated server point of view, we can even claim that this remote index is unique, because the view has only one row per distinct partkey. So, we tell our federated server about the index.

**Note:** DB2 Information Integrator does not verify the existence of a remote index defined in a local index specification. Even if there is no remote index actually defined, we could still create a fake specification, if we thought it would give a better plan!

We created the index specification for the nickname on the view as shown in Example 14-10. It informs us that there is an index existing on the underlying table LINEITEM.

*Example 14-10 Create index specification for nickname on view*

```
create unique index sc07.soldx1
on sc07.soldbypart (l_partkey, total_sold) specification only;
```

Our idea is to encourage the optimizer to perform a nested loop join to the view, because we know there is a good access path on the underlying table LINEITEM for column l\_partkey.

**Note:** A nested loop join to the view goes through the result set of the query for table PART, and for every PART record, DB2 accesses the view.

We collected our Explain plan for the same query shown in Example 14-7.

The result of performing the query again, with the index specification in place, is displayed in Example 14-11.

*Example 14-11 Result of query in Example 14-7*

```
Statement number: 1

select l_partkey, total_sold from sc07.soldbypart, ora.part where
P_CONTAINER = 'JUMBO PKG' and
p_partkey = l_partkey

  L_PARTKEY      TOTAL_SOLD
-----
1 8.6000000000000000e+02
28 5.9800000000000000e+02
91 5.5600000000000000e+02
...

Number of rows retrieved is:      5013
Number of rows sent to output is: 5013

Prepare Time is:      0.018      seconds
```



Execute Time is:	2.530	seconds
Fetch Time is:	0.416	seconds
Elapsed Time is:	2.965	seconds

-----

Nice! Less than 3 seconds.

Let us look at the access plan in Figure 14-2.

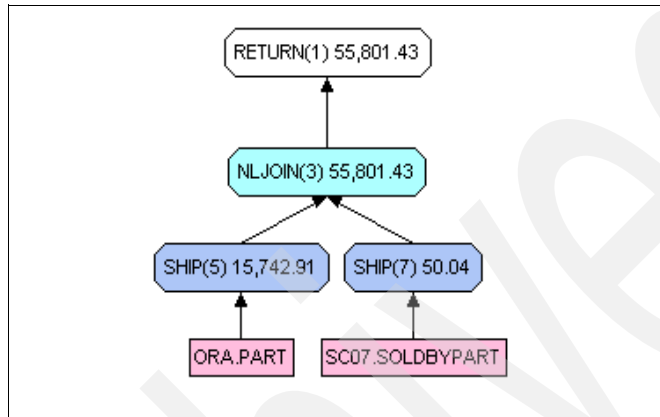


Figure 14-2 Access plan with index specification on nickname

The access plan looks much easier and simpler. You can look at Example 14-12 for details on the queries shipped to the data sources.

#### Example 14-12 Shipped queries

```

SELECT A0."P_PARTKEY"
FROM "TPCH"."PART" A0
WHERE (A0."P_CONTAINER" = 'JUMBO PKG ')

SELECT A0."L_PARTKEY", A0."TOTAL_SOLD"
FROM "SC07"."SOLDBYPART" A0
WHERE (CAST(:H0 AS DECIMAL(10,0)) = A0."L_PARTKEY")
ORDER BY 1 ASC, 2 ASC
FOR READ ONLY
  
```

We got rid of the merge join at the first try. Instead of the merge join, DB2 now performs a nested loop join, which seems to be ideal for this query. Remember, there is an index available on column `l_partkey` on table `LINEITEM`, which is the reason for the large performance difference.

In this case we got the plan to change without using any “good stats”; it just changed for the better with default values. In general, we might need to update

the statistics in order to achieve the plan change, either manually, or using `getstats` to do that.

### 14.3.2 Adding index statistics to nicknames for views

It is always recommended that you create nicknames for the tables that contain the data the federated users want to access, rather than create nicknames for views that refer to these tables. However, there are instances where this is not allowed. An example is where the users of the data at the data source themselves are not allowed to access the tables except through views, such as when the data in the tables is being updated constantly, and the query users use read-only views that keep the update programs and query users from deadlocking each other.

When nicknames must be created for views, it is recommended to find out which columns of the tables in the view are indexed at the data source, and to find out the number of rows that result from the view. The nickname can then be created with an SQL script that also includes statements to add the index information and update the statistics for the nickname. Example 14-13 lists such a script using information from the scenario shown in this chapter.

---

*Example 14-13 Creating a nickname with index statistics*

---

```
create nickname sc07.soldbypart for db2udb.sc07.soldbypart ;
create unique index sc07.soldx1 on sco7.soldbypart (l_partkey, total_sold)
specification only;
update sysstat.tables set card = 200000 where tabschema='SC07' and
tabname='SOLDBYPART' ;
```

---

## Using materialized query tables

In this chapter we describe how queries can take advantage of materialized query tables (MQTs). MQTs allow you to store the answer set of queries against local tables, nicknames, or a combination of the two. The optimizer is able to transparently use the previously stored results to satisfy other similar queries. When MQTs are defined by queries involving nicknames (potentially on multiple sources), they become a valuable way to cache remote data locally on DB2 Information Integrator.

The chapter is structured as:

- ▶ Overview
- ▶ Accessing data without MQT
- ▶ Accessing data with MQT

## 15.1 Overview

DB2 Information Integrator's cost-based optimizer considers not only standard database statistics, such as cardinality and indexes, but also network and server resources, and the query power available in the data source engine.

A materialized query table (MQT) is a table that is defined based on the result of a query. The MQT mechanism allows administrators to define materialized views of data in a set of underlying tables, or nicknames. For certain classes of queries, the database can automatically determine whether the MQT can answer a query, without accessing the base tables. With MQTs you can automatically and transparently route read-only queries to the materialized views, while you transparently route updates to the database.

You can define MQTs on nicknames. By using a nickname to reference an MQT, the local DB2 instance can cache the remote data. This capability results in better performance for federated queries, because the queries access the remote data locally. If for any reason the remote source table becomes unavailable, DB2 can still use the MQT that is defined on it, if it meets the routing criteria. This results in both improved availability and performance.

This is possible and acceptable whenever the user does not require the most up-to-date data. That is, the application can deal with data, which can be refreshed periodically, rather than updated synchronously.

By using MQTs, you can speed queries that join multiple tables across several servers on the LAN. MQTs allow you to save the answer set on a single server, and update that materialized view when data changes in any of the servers. The update of the data in the MQTs that refer to nicknames needs to be driven at the federated server. The federated server has no way of knowing when the tables at the data sources has changed; there is no mechanism that automatically updates data in MQTs for nicknames when data at data sources changes.

Any type of federated relational data can be a candidate for these types of joins. This includes Oracle, SQL Server, and other *relational* data sources.

We now want to describe a query without and with MQT in the next two sections. The query accesses a nickname on four different data sources (DB2 AIX, DB2 for z/OS, Oracle and SQL Server). The query counts the number of orders placed between certain dates by customers who are not in Asia, and who are not part of the Building market segment.

In Example 15-1, you see the query we want to perform.

*Example 15-1 MQT - Sample query*

---

```
SELECT COUNT(*)
FROM
    DB2DAT.ORDERS,
    MSSQL.CUSTOMER,
    DB2ZOS.NATION,
    ORA.REGION
WHERE
    N_NATIONKEY = C_NATIONKEY
    AND N_REGIONKEY = R_REGIONKEY
    AND R_NAME <> 'ASIA'
    AND O_CUSTKEY = C_CUSTKEY
    AND C_MKTSEGMENT <> 'BUILDING'
    AND O_ORDERDATE BETWEEN DATE('1994-07-01') AND DATE('1994-09-30')
```

---

## 15.2 Accessing data without MQT

We executed the query listed in Example 15-1 with the response time in Example 15-2.

*Example 15-2 MQT - Query response time*

---

```
1
-----
      36751

Number of rows retrieved is:      1
Number of rows sent to output is: 1

Prepare Time is:      0.077      seconds
Execute Time is:      8.266      seconds
Fetch Time is:        0.000      seconds
Elapsed Time is:      8.343      seconds
-----
```

---

You can see in Figure 15-1 the access plan from the Visual Explain tool. We need to join multiple tables across several servers on the LAN. The SHIP operators that retrieve data from for Oracle, DB2 for z/OS, and SQL Server obtain a list of customers that meet the conditions before performing the final nested loop join to the ORDERS DB2-source nickname. To avoid retrieving all of this customer data each time, we could create an MQT on the federated database that pre-joins the first three nicknames.

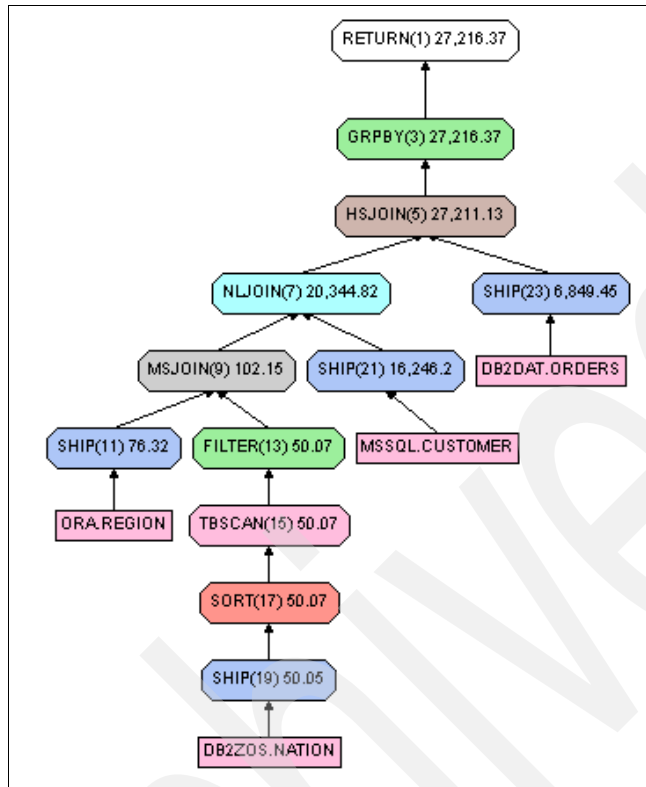


Figure 15-1 MQT - Sample query access plan

## 15.3 Accessing data with MQT

For our next step, we created an MQT as detailed in Example 15-3. The materialized table contains a list of all non-BUILDING customers by region. Note that an MQT is just like a regular table in many ways. It is a good idea to do RUNSTATS on it, and you can also create indexes to provide better access paths.

Example 15-3 MQT - Create table

```
-- Create MQT
CREATE TABLE DB2FED32.CUST_MQT AS
  (SELECT
    C_CUSTKEY, R_NAME
  FROM
    MSSQL.CUSTOMER,
    DB2ZOS.NATION,
```

```

ORA.REGION
WHERE
    N_NATIONKEY = C_NATIONKEY
    AND N_REGIONKEY = R_REGIONKEY
    AND C_MKTSEGMENT <> 'BUILDING')
DATA INITIALLY DEFERRED REFRESH DEFERRED;
-- Populate MQT
REFRESH TABLE DB2FED32.CUST_MQT;
-- Create index on MQT
CREATE INDEX DB2FED32.CUST_MQT_IDX1 ON DB2FED32.CUST_MQT (C_CUSTKEY, R_NAME);
-- Update statistics on MQT
RUNSTATS ON TABLE DB2FED32.CUST_MQT FOR INDEXES ALL;

```

---

The clause **DATA INITIALLY DEFERRED** means that data is not inserted into the table as part of the **CREATE TABLE** statement. Instead, you have to execute a **REFRESH TABLE** statement to populate the table. The clause **REFRESH DEFERRED** means that the data in the table only reflects the result of the query as a snapshot at the time you issue the **REFRESH TABLE** statement. For more information on creating MQTs, see the *IBM DB2 Universal Database SQL Reference Volume 2 Version 8*, SC09-4845. When we were ready to populate the MQT we have just created, we issued the **REFRESH TABLE** statement, created appropriate indexes (optional), and updated the statistics.

When you create a MQT, you can specify the clause **REFRESH IMMEDIATE** or **DEFERRED**. The clause **REFRESH IMMEDIATE** is *not* supported when the full select includes a reference to a nickname.

The optimizer may choose to use an MQT with the **REFRESH DEFERRED** option if **CURRENT REFRESH AGE** setting is set to **ANY**. **CURRENT QUERY OPTIMIZATION** and **CURRENT REFRESH AGE** settings are described in detail in *IBM DB2 Universal Database SQL Reference Volume 2 Version 8*, SC09-4845.

To enable the optimizer to choose the MQT automatically, the **ENABLE QUERY OPTIMIZATION** must be in effect (it is the default).

In our case, we used the clause **REFRESH DEFERRED**. In order to allow DB2 to choose this MQT, which may be out of date with respect to the actual remote data on which it is based, you need to change **CURRENT REFRESH AGE** to **ANY**. You can either change the default configuration of your federated database with a DB2 command, as in Example 15-4, then shut down and restart DB2, or you can use the **SET CURRENT REFRESH AGE** statement of Example 15-4 before executing the query.

Example 15-4 MQT - Enabling MQT use

```
UPDATE DATABASE CONFIGURATION FOR <dbname> USING DFT_REFRESH_AGE ANY
or
SET CURRENT REFRESH AGE ANY
```

Now the optimizer automatically chooses to use the MQT, eliminating the need to join the three nicknames that produce the list of qualifying customers. The only join that remains is from the MQT to the ORDERS table on the remote DB2 source. See the Explain in Figure 15-2. The estimated cost of 7292 with the MQT compares favorably with that of 27216 in the previous plan shown in Example 15-1.

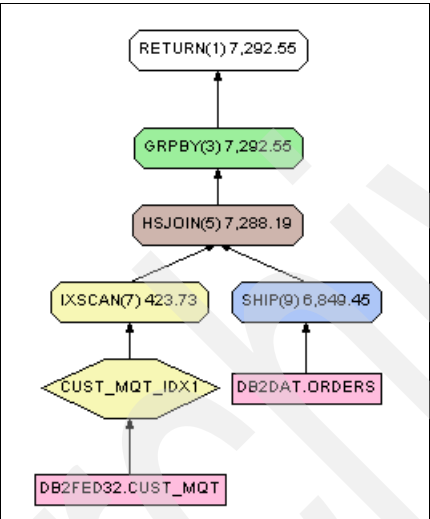


Figure 15-2 MQT - Explain

We then executed the query with the new access plan, and we reported the response time in Example 15-5. The query now takes 0.62 second instead of 8.34 seconds.

Example 15-5 MQT - Response time with new access plan

```
1
-----
      36751

Number of rows retrieved is:      1
Number of rows sent to output is: 1

Prepare Time is:      0.051      seconds
Execute Time is:      0.566      seconds
```



Fetch Time is:	0.000	seconds
Elapsed Time is:	0.616	seconds

---

So, in summary, using MQTs may yield performance advantages when the data from the tables:

- ▶ Is very frequently joined
- ▶ Is rarely, if ever, updated
- ▶ Is very frequently calculated, such as in SUMs

We conclude that MQTs if applied properly are very useful in various situations. The above example illustrates how MQTs can be applied for improving a query with multiple tables across several servers on the LAN. While MQTs are very convenient, they come at the price of additional disk space.

**Note:** In this chapter, we concentrated only on the MQT solution. Alternatively, we could decide to use the DB2 Information Integrator replication facility to copy the remote tables to one remote data source and facilitate the query execution. For general information on DB2 Information Integrator replication functions and examples, refer to the manual *IBM DB2 Universal Database Replication Guide and Reference Version 8, Release 1, SC27-1121*, and the redbook *A Practical Guide to DB2 UDB Data Replication V8, SG24-6828*.

### 15.3.1 Refreshing data in MQTs with nicknames

When REFRESH TABLE is executed for an MQT that includes nicknames in its definition, the process that DB2 uses to refresh the data in the MQT is the same as the process for refreshing data in MQTs that only have tables in their definition:

1. DB2 deletes the data from the MQT.
2. DB2 executes an `insert into` MQT with a sub-select that contains the SQL statement that was used in the definition of the MQT.

To understand the performance of the MQT refresh:

- ▶ Explain the SQL statement that was used in the MQT definition to see the access plan that Information Integrator will use to fetch the data for the refresh
- ▶ Keep in mind that normal DELETE and INSERT SQL operations are used to refresh the MQT. These operations are logged by DB2, and if there are indexes on the MQT (which is recommended), these indexes will be updated as part of the refresh process.





## Part 5

# Appendixes





## Case study

This appendix has a list of the scripts and definitions that we worked with during the setup of our case study:

- ▶ DDL for DB2 for AIX  
This describes the Data Definition Language (DDL) statements for setting up the database objects for a data source. We create tables and indexes with this script.
- ▶ Script for building the case study  
This describes the setup of all federated database objects we used. We created wrappers, servers, user mappings, and nicknames for all our different data sources.
- ▶ Data sources/wrapper/server/schema names
- ▶ DB2 list db directory
- ▶ DB2 list dcs directory
- ▶ DB2 list Node Directory
- ▶ /home/informix/ids92/etc/sqlhosts
- ▶ /oracle9ir2/network/admin/tnsnames.ora
- ▶ /home/db2fed32/.odbc.ini
- ▶ /home/db2fed32/sqllib/cfg/db2dj.ini
- ▶ ldb2set -all

► env

## A.1 DDL for DB2 for AIX

Example A-1 lists the DDL statements for setting up a DB2 data source used in our case study. You can perform this script by using the following command on your AIX shell prompt:

```
db2 -tvf db2udbdd1.sql
```

*Example: A-1 DB2 data source - Data definition language*

---

```
CONNECT TO DATDB user db2dat64 using db2dat64;
```

```
-----  
-- DDL Statements for table "DATDB"."NATION"  
-----
```

```
CREATE TABLE "DATDB"."NATION" (  
    "N_NATIONKEY" INTEGER NOT NULL ,  
    "N_NAME" CHAR(25) NOT NULL ,  
    "N_REGIONKEY" INTEGER NOT NULL ,  
    "N_COMMENT" VARCHAR(152) NOT NULL WITH DEFAULT ''  
);  
ALTER TABLE "DATDB"."NATION" LOCKSIZE TABLE;
```

```
-- load data before creating indexes.  
-- load command looks like this:  
-- load from nation.tbl of del modified by coldel| replace into DATDB.nation ;
```

```
-- DDL Statements for indexes on Table "DATDB"."NATION"  
CREATE UNIQUE INDEX "DATDB"."N_NATKEYNAM" ON "DATDB"."NATION"  
    ("N_NATIONKEY" ASC)  
    INCLUDE ("N_NAME" ASC) PCTFREE 0 ;
```

```
-- DDL Statements for indexes on Table "DATDB"."NATION"  
CREATE UNIQUE INDEX "DATDB"."N_REGKEYNATKEYNAM" ON "DATDB"."NATION"  
    ("N_REGIONKEY" ASC,  
    "N_NATIONKEY" ASC)  
    INCLUDE ("N_NAME" ASC) PCTFREE 0 ;
```

```
RUNSTATS ON TABLE DATDB.NATION FOR INDEXES ALL;
```

```
-----  
-- DDL Statements for table "DATDB"."REGION"  
-----
```

```
CREATE TABLE "DATDB"."REGION" (  
    "R_REGIONKEY" INTEGER NOT NULL ,  
    "R_NAME" CHAR(25) NOT NULL ,  
    "R_COMMENT" VARCHAR(152) NOT NULL WITH DEFAULT ''
```

```

;
ALTER TABLE "DATDB"."REGION" LOCKSIZE TABLE;

-- load data before creating indexes.
-- load command looks like this:
-- load from region.tbl of del modified by coldel| replace into DATDB.region ;

-- DDL Statements for indexes on Table "DATDB"."REGION"
CREATE UNIQUE INDEX "DATDB"."R_NAMREGKEY" ON "DATDB"."REGION"
  ("R_NAME" ASC,
   "R_REGIONKEY" ASC)
  PCTFREE 0 ;

RUNSTATS ON TABLE DATDB.REGION FOR INDEXES ALL;

-----
-- DDL Statements for table "DATDB"."PART"
-----

CREATE TABLE "DATDB"."PART" (
  "P_PARTKEY" INTEGER NOT NULL ,
  "P_NAME" VARCHAR(55) NOT NULL ,
  "P_MFGR" CHAR(25) NOT NULL ,
  "P_BRAND" CHAR(10) NOT NULL ,
  "P_TYPE" VARCHAR(25) NOT NULL ,
  "P_SIZE" INTEGER NOT NULL ,
  "P_CONTAINER" CHAR(10) NOT NULL ,
  "P_RETAILPRICE" DOUBLE NOT NULL ,
  "P_COMMENT" VARCHAR(23) NOT NULL WITH DEFAULT '')
;
ALTER TABLE "DATDB"."PART" LOCKSIZE TABLE;

-- load data before creating indexes.
-- load command looks like this:
-- load from part.tbl of del modified by coldel| replace into DATDB.part ;

-- DDL Statements for indexes on Table "DATDB"."PART"
CREATE UNIQUE INDEX "DATDB"."PK_P_PARTKEY" ON "DATDB"."PART"
  ("P_PARTKEY" ASC)
  PCTFREE 0 ;

RUNSTATS ON TABLE DATDB.PART FOR INDEXES ALL;

-----
-- DDL Statements for table "DATDB"."SUPPLIER"
-----

CREATE TABLE "DATDB"."SUPPLIER" (

```



```

        "S_SUPPKEY" INTEGER NOT NULL ,
        "S_NAME" CHAR(25) NOT NULL ,
        "S_ADDRESS" VARCHAR(40) NOT NULL ,
        "S_NATIONKEY" INTEGER NOT NULL ,
        "S_PHONE" CHAR(15) NOT NULL ,
        "S_ACCTBAL" DOUBLE NOT NULL ,
        "S_COMMENT" VARCHAR(101) NOT NULL WITH DEFAULT ''
    ;
    ALTER TABLE "DATDB"."SUPPLIER" LOCKSIZE TABLE;

-- load data before creating indexes.
-- load command looks like this:
-- load from supplier.tbl of del modified by coldel| replace into
DATDB.supplier ;

-- DDL Statements for indexes on Table "DATDB"."SUPPLIER"
CREATE UNIQUE INDEX "DATDB"."PK_S_SUPPKEY" ON "DATDB"."SUPPLIER"
("S_SUPPKEY" ASC)
PCTFREE 0 ;

-- DDL Statements for indexes on Table "DATDB"."SUPPLIER"
CREATE UNIQUE INDEX "DATDB"."S_NAT_SKEY_REG" ON "DATDB"."SUPPLIER"
("S_NATIONKEY" ASC,
 "S_SUPPKEY" ASC)
PCTFREE 0 ;

-- DDL Statements for indexes on Table "DATDB"."SUPPLIER"
CREATE UNIQUE INDEX "DATDB"."UXS_SKNK" ON "DATDB"."SUPPLIER"
("S_SUPPKEY" ASC)
INCLUDE ("S_NATIONKEY" ASC) PCTFREE 0 ;

RUNSTATS ON TABLE DATDB.SUPPLIER FOR INDEXES ALL ;

-----
-- DDL Statements for table "DATDB"."PARTSUPP"
-----

CREATE TABLE "DATDB"."PARTSUPP" (
    "PS_PARTKEY" INTEGER NOT NULL ,
    "PS_SUPPKEY" INTEGER NOT NULL ,
    "PS_AVAILQTY" INTEGER NOT NULL ,
    "PS_SUPPLYCOST" DOUBLE NOT NULL ,
    "PS_COMMENT" VARCHAR(199) NOT NULL WITH DEFAULT ''
)
;
ALTER TABLE "DATDB"."PARTSUPP" LOCKSIZE TABLE;

-- load data before creating indexes.
-- load command looks like this:

```

```

-- load from partsupp.tbl of del modified by coldel| replace into
DATDB.partsupp ;

-- DDL Statements for indexes on Table "DATDB"."PARTSUPP"
CREATE UNIQUE INDEX "DATDB"."UXPS_PK2KSC" ON "DATDB"."PARTSUPP"
("PS_PARTKEY" ASC,
 "PS_SUPPKEY" ASC)
INCLUDE ("PS_SUPPLYCOST" ASC) PCTFREE 0 ;

RUNSTATS ON TABLE DATDB.PARTSUPP FOR INDEXES ALL;

-----
-- DDL Statements for table "DATDB"."CUSTOMER"
-----

CREATE TABLE "DATDB"."CUSTOMER" (
    "C_CUSTKEY" INTEGER NOT NULL ,
    "C_NAME" CHAR(25) NOT NULL ,
    "C_ADDRESS" VARCHAR(40) NOT NULL ,
    "C_NATIONKEY" INTEGER NOT NULL ,
    "C_PHONE" CHAR(15) NOT NULL ,
    "C_ACCTBAL" DOUBLE NOT NULL ,
    "C_MKTSEGMENT" CHAR(10) NOT NULL ,
    "C_COMMENT" VARCHAR(117) NOT NULL WITH DEFAULT ''
);
ALTER TABLE "DATDB"."CUSTOMER" LOCKSIZE TABLE;

-- load data before creating indexes.
-- load command looks like this:
-- load from customer.tbl of del modified by coldel| replace into
DATDB.customer ;

-- DDL Statements for indexes on Table "DATDB"."CUSTOMER"
CREATE UNIQUE INDEX "DATDB"."C_CK" ON "DATDB"."CUSTOMER"
("C_CUSTKEY" ASC)
PCTFREE 0 ;

-- DDL Statements for indexes on Table "DATDB"."CUSTOMER"
CREATE UNIQUE INDEX "DATDB"."C_NAT_CKEY_REG" ON "DATDB"."CUSTOMER"
("C_NATIONKEY" ASC,
 "C_CUSTKEY" ASC)
PCTFREE 0 ;

RUNSTATS ON TABLE DATDB.CUSTOMER FOR INDEXES ALL ;

-----
-- DDL Statements for table "DATDB"."ORDERS"
-----

```

```

CREATE TABLE "DATDB"."ORDERS" (
    "O_ORDERKEY" INTEGER NOT NULL ,
    "O_CUSTKEY" INTEGER NOT NULL ,
    "O_ORDERSTATUS" CHAR(1) NOT NULL ,
    "O_TOTALPRICE" DOUBLE NOT NULL ,
    "O_ORDERDATE" DATE NOT NULL ,
    "O_ORDERPRIORITY" CHAR(15) NOT NULL ,
    "O_CLERK" CHAR(15) NOT NULL ,
    "O_SHIPPRIORITY" INTEGER NOT NULL ,
    "O_COMMENT" VARCHAR(79) NOT NULL WITH DEFAULT '')
;
ALTER TABLE "DATDB"."ORDERS" APPEND ON;

-- load data before creating indexes.
-- load command looks like this:
-- load from orders.tbl of del modified by coldel| replace into DATDB.orders ;

-- DDL Statements for indexes on Table "DATDB"."ORDERS"
CREATE UNIQUE INDEX "DATDB"."O_CK_OD_OK_SP" ON "DATDB"."ORDERS"
("O_CUSTKEY" ASC,
 "O_ORDERDATE" ASC,
 "O_ORDERKEY" ASC)
INCLUDE ("O_SHIPPRIORITY" ASC) PCTFREE 2 ;

RUNSTATS ON TABLE DATDB.ORDERS FOR INDEXES ALL ;

-----
-- DDL Statements for table "DATDB"."LINEITEM"
-----

CREATE TABLE "DATDB"."LINEITEM" (
    "L_ORDERKEY" INTEGER NOT NULL ,
    "L_PARTKEY" INTEGER NOT NULL ,
    "L_SUPPKEY" INTEGER NOT NULL ,
    "L_LINENUMBER" INTEGER NOT NULL ,
    "L_QUANTITY" DOUBLE NOT NULL ,
    "L_EXTENDEDPRICE" DOUBLE NOT NULL ,
    "L_DISCOUNT" DOUBLE NOT NULL ,
    "L_TAX" DOUBLE NOT NULL ,
    "L_RETURNFLAG" CHAR(1) NOT NULL ,
    "L_LINESTATUS" CHAR(1) NOT NULL ,
    "L_SHIPDATE" DATE NOT NULL ,
    "L_COMMITDATE" DATE NOT NULL ,
    "L_RECEIPTDATE" DATE NOT NULL ,
    "L_SHIPINSTRUCT" CHAR(25) NOT NULL ,
    "L_SHIPMODE" CHAR(10) NOT NULL ,
    "L_COMMENT" VARCHAR(44) NOT NULL WITH DEFAULT '')
;
ALTER TABLE "DATDB"."LINEITEM" APPEND ON;

```

```

-- load data before creating indexes.
-- load command looks like this:
-- load from lineitem.tbl of del modified by coldel| replace into
DATDB.lineitem ;

-- DDL Statements for indexes on Table "DATDB"."LINEITEM"
CREATE INDEX "DATDB"."L_PKSKOKEPDSQN" ON "DATDB"."LINEITEM"
("L_PARTKEY" ASC,
 "L_SUPPKEY" ASC,
 "L_ORDERKEY" ASC,
 "L_EXTENDEDPRICE" ASC,
 "L_DISCOUNT" ASC,
 "L_QUANTITY" ASC)
PCTFREE 4 ;

RUNSTATS ON TABLE DATDB.LINEITEM FOR INDEXES ALL;

```

---

## A.2 Script for building the case study

The script in Example A-2 builds all federated database objects that we used in our case study. Perform this script by using following command on your AIX shell prompt:

**db2 -tvf DB2\_FederatedDBObjects.sql**

*Example: A-2 Case study - Creating federated database objects*

---

```
connect to feddb user db2fed32 using db2fed32;
```

```
-----
----- DB2 Family -----
```

```
-- Creating DB2 UDB Family Wrapper:
CREATE WRAPPER "DB2DRDA" LIBRARY 'libdb2drda.a';
```

```
-----
-----DB2 for z/OS-----
```

```
-- Create DB2/zSeries Server:
CREATE SERVER DB2ZSERIES TYPE DB2/390 VERSION '7' WRAPPER "DB2DRDA"
AUTHID "paolor1" PASSWORD "itsosj"
OPTIONS( ADD DBNAME 'DCSDB2G', PASSWORD 'Y');
```

```
-- Create User Mapping:
CREATE USER MAPPING FOR "DB2FED32" SERVER "DB2ZSERIES"
OPTIONS ( ADD REMOTE_AUTHID 'paolor1', ADD REMOTE_PASSWORD 'itsosj');
```

```

-- Create Nicknames with schema update:
CREATE NICKNAME DB2ZOS.LINEITEM FOR DB2ZSERIES.PAOLOR4.LINEITEM;
CREATE NICKNAME DB2ZOS.ORDERS FOR DB2ZSERIES.PAOLOR4.ORDER;
CREATE NICKNAME DB2ZOS.PARTSUPP FOR DB2ZSERIES.PAOLOR4.PARTSUPP;
CREATE NICKNAME DB2ZOS.PART FOR DB2ZSERIES.PAOLOR4.PART;
CREATE NICKNAME DB2ZOS.CUSTOMER FOR DB2ZSERIES.PAOLOR4.CUSTOMER;
CREATE NICKNAME DB2ZOS.SUPPLIER FOR DB2ZSERIES.PAOLOR4.SUPPLIER;
CREATE NICKNAME DB2ZOS.NATION FOR DB2ZSERIES.PAOLOR4.NATION;
CREATE NICKNAME DB2ZOS.REGION FOR DB2ZSERIES.PAOLOR4.REGION;

-----
----- DB2 iSeries -----

-- Create DB2/iSeries Server:
CREATE SERVER DB2ISERIES TYPE DB2/400 VERSION '5.2' WRAPPER "DB2DRDA"
AUTHID "sanjose" PASSWORD "itso"
OPTIONS( ADD DBNAME 'DCSOS400', PASSWORD 'Y');

-- Create User Mapping:
CREATE USER MAPPING FOR "DB2FED32" SERVER "DB2ISERIES"
OPTIONS ( ADD REMOTE_AUTHID 'sanjose', ADD REMOTE_PASSWORD 'itso') ;

-- Create Nickname with schema update:
CREATE NICKNAME DB2OS400.CUSTOMER FOR DB2ISERIES.SAMPLEDB.CUSTOMER;
CREATE NICKNAME DB2OS400.LINEITEM FOR DB2ISERIES.SAMPLEDB.LINEITEM;
CREATE NICKNAME DB2OS400.NATION FOR DB2ISERIES.SAMPLEDB.NATION;
CREATE NICKNAME DB2OS400.ORDERS FOR DB2ISERIES.SAMPLEDB.ORDERS;
CREATE NICKNAME DB2OS400.PART FOR DB2ISERIES.SAMPLEDB.PART;
CREATE NICKNAME DB2OS400.PARTSUPP FOR DB2ISERIES.SAMPLEDB.PARTSUPP;
CREATE NICKNAME DB2OS400.REGION FOR DB2ISERIES.SAMPLEDB.REGION;
CREATE NICKNAME DB2OS400.SUPPLIER FOR DB2ISERIES.SAMPLEDB.SUPPLIER;

-----
----- DB2 UDB -----

-- Create DB2 UDB Server:
CREATE SERVER DB2UDB TYPE DB2/UDB VERSION '8.1' WRAPPER "DB2DRDA"
AUTHID "db2dat64" PASSWORD "db2dat64"
OPTIONS( ADD DBNAME 'datdb', PASSWORD 'Y');

-- Create User Mapping:
CREATE USER MAPPING FOR "DB2FED32" SERVER "DB2UDB"
OPTIONS ( ADD REMOTE_AUTHID 'db2dat64', ADD REMOTE_PASSWORD 'db2dat64') ;

CREATE NICKNAME DB2DAT.NATION FOR DB2UDB.DATDB.NATION;
CREATE NICKNAME DB2DAT.REGION FOR DB2UDB.DATDB.REGION;
CREATE NICKNAME DB2DAT.PART FOR DB2UDB.DATDB.PART;

```

```

CREATE NICKNAME DB2DAT.SUPPLIER FOR DB2UDB.DATDB.SUPPLIER;
CREATE NICKNAME DB2DAT.PARTSUPP FOR DB2UDB.DATDB.PARTSUPP;
CREATE NICKNAME DB2DAT.CUSTOMER FOR DB2UDB.DATDB.CUSTOMER;
CREATE NICKNAME DB2DAT.ORDERS FOR DB2UDB.DATDB.ORDERS;
CREATE NICKNAME DB2DAT.LINEITEM FOR DB2UDB.DATDB.LINEITEM;

-----
----- Informix -----

-- Creating Informix Wrapper
CREATE WRAPPER "INFORMIX" LIBRARY 'libdb2informix.a';

-- Create Informix Server:
CREATE SERVER IDS9 TYPE INFORMIX VERSION '9' WRAPPER "INFORMIX"
OPTIONS( ADD NODE 'demo_on', DBNAME 'tpc',
PASSWORD 'Y', IUD_APP_SVPT_ENFORCE 'Y');

-- Create User Mapping:
CREATE USER MAPPING FOR "DB2FED32" SERVER "IDS9"
OPTIONS ( ADD REMOTE_AUTHID 'informix', ADD REMOTE_PASSWORD 'informix') ;

-- Create Nickname with schema update:
CREATE NICKNAME INFX.NATION FOR IDS9."informix"."nation";
CREATE NICKNAME INFX.REGION FOR IDS9."informix"."region";
CREATE NICKNAME INFX.PART FOR IDS9."informix"."part";
CREATE NICKNAME INFX.SUPPLIER FOR IDS9."informix"."supplier";
CREATE NICKNAME INFX.PARTSUPP FOR IDS9."informix"."partsupp";
CREATE NICKNAME INFX.CUSTOMER FOR IDS9."informix"."customer";
CREATE NICKNAME INFX.ORDERS FOR IDS9."informix"."orders";
CREATE NICKNAME INFX.LINEITEM FOR IDS9."informix"."lineitem";

-----
----- Oracle -----

-- Create Oracle Net8 Wrapper:
CREATE WRAPPER "ORACLE" LIBRARY 'libdb2net8.a';

-- Create Oracle Server:
CREATE SERVER ORACLE9 TYPE ORACLE VERSION '9' WRAPPER "ORACLE"
OPTIONS( ADD NODE 'ORAFED', PASSWORD 'Y');

-- Create User Mapping:
CREATE USER MAPPING FOR "DB2FED32" SERVER "ORACLE9"
OPTIONS ( ADD REMOTE_AUTHID 'tpch', ADD REMOTE_PASSWORD 'tpch') ;

-- Create Nickname with schema update:
CREATE NICKNAME ORA.CUSTOMER FOR ORACLE9.TPCH.CUSTOMER;
CREATE NICKNAME ORA.LINEITEM FOR ORACLE9.TPCH.LINEITEM;
CREATE NICKNAME ORA.NATION FOR ORACLE9.TPCH.NATION;

```

```

CREATE NICKNAME ORA.ORDERS FOR ORACLE9.TPCH.ORDERS;
CREATE NICKNAME ORA.PART FOR ORACLE9.TPCH.PART;
CREATE NICKNAME ORA.PARTSUPP FOR ORACLE9.TPCH.PARTSUPP;
CREATE NICKNAME ORA.REGION FOR ORACLE9.TPCH.REGION;
CREATE NICKNAME ORA.SUPPLIER FOR ORACLE9.TPCH.SUPPLIER;

-----
----- SQL Server -----

-- Create MSSQLServer Wrapper:
CREATE WRAPPER "MSSQL" LIBRARY 'libdb2mssql3.a';

-- Create MSSQLServer Server:
CREATE SERVER MSSQL2000 TYPE MSSQLSERVER VERSION '2000' WRAPPER "MSSQL"
OPTIONS( ADD NODE 'RADONSQL', DBNAME 'sqlservdb', PASSWORD 'Y');

-- Create User Mapping:
CREATE USER MAPPING FOR "DB2FED32" SERVER "MSSQL2000"
OPTIONS ( ADD REMOTE_AUTHID 'sqlserv', ADD REMOTE_PASSWORD 'sqlserv' );

-- Create Nickname with schema update:
CREATE NICKNAME MSSQL.NATION FOR MSSQL2000."sqlserv".NATION;
CREATE NICKNAME MSSQL.REGION FOR MSSQL2000."sqlserv".REGION;
CREATE NICKNAME MSSQL.PART FOR MSSQL2000."sqlserv".PART;
CREATE NICKNAME MSSQL.SUPPLIER FOR MSSQL2000."sqlserv".SUPPLIER;
CREATE NICKNAME MSSQL.PARTSUPP FOR MSSQL2000."sqlserv".PARTSUPP;
CREATE NICKNAME MSSQL.CUSTOMER FOR MSSQL2000."sqlserv".CUSTOMER;
CREATE NICKNAME MSSQL.ORDERS FOR MSSQL2000."sqlserv".ORDERS;
CREATE NICKNAME MSSQL.LINEITEM FOR MSSQL2000."sqlserv".LINEITEM;

-----
----- Flat Files -----
CREATE WRAPPER "FLATFILES" LIBRARY 'libdb2lsfile.a';
CREATE SERVER FLATSERV WRAPPER "FLATFILES";

CREATE NICKNAME FLAT.REGION (
    R_REGIONKEY INTEGER NOT NULL ,
    R_NAME CHARACTER (25) NOT NULL ,
    R_COMMENT VARCHAR (152) NOT NULL )
FOR SERVER "FLATSERV"
OPTIONS(COLUMN_DELIMITER '|',
    FILE_PATH '/exchange/flatfiles/region.txt');

CREATE NICKNAME FLAT.NATION (
    N_NATIONKEY INTEGER NOT NULL ,
    N_NAME CHARACTER (25) NOT NULL ,
    N_REGIONKEY INTEGER NOT NULL ,
    N_COMMENT VARCHAR(152) NOT NULL )
FOR SERVER "FLATSERV"

```

```

OPTIONS(COLUMN_DELIMITER '|' ,
        FILE_PATH '/exchange/flatfiles/nation.txt');

CREATE NICKNAME FLAT.PART (
    P_PARTKEY INTEGER NOT NULL ,
    P_NAME VARCHAR(55) NOT NULL ,
    P_MFGR CHARACTER(25) NOT NULL ,
    P_BRAND CHARACTER(10) NOT NULL ,
    P_TYPE VARCHAR(25) NOT NULL ,
    P_SIZE INTEGER NOT NULL ,
    P_CONTAINER CHARACTER(10) NOT NULL ,
    P_RETAILPRICE DOUBLE NOT NULL ,
    P_COMMENT VARCHAR(23) NOT NULL )
FOR SERVER "FLATSERV"
OPTIONS(COLUMN_DELIMITER '|' ,
        FILE_PATH '/exchange/flatfiles/part.txt');

CREATE NICKNAME FLAT.SUPPLIER (
    S_SUPPKEY INTEGER NOT NULL ,
    S_NAME CHARACTER(25) NOT NULL ,
    S_ADDRESS VARCHAR(40) NOT NULL ,
    S_NATIONKEY INTEGER NOT NULL ,
    S_PHONE CHARACTER(15) NOT NULL ,
    S_ACCTBAL DOUBLE NOT NULL ,
    S_COMMENT VARCHAR(101) NOT NULL )
FOR SERVER "FLATSERV"
OPTIONS(COLUMN_DELIMITER '|' ,
        FILE_PATH '/exchange/flatfiles/supplier.txt');

CREATE NICKNAME FLAT.CUSTOMER (
    C_CUSTKEY INTEGER NOT NULL ,
    C_NAME CHARACTER(25) NOT NULL ,
    C_ADDRESS VARCHAR(40) NOT NULL ,
    C_NATIONKEY INTEGER NOT NULL ,
    C_PHONE CHARACTER(15) NOT NULL ,
    C_ACCTBAL DOUBLE NOT NULL ,
    C_MKTSEGMENT CHARACTER(10) NOT NULL ,
    C_COMMENT VARCHAR(117) NOT NULL )
FOR SERVER "FLATSERV"
OPTIONS(COLUMN_DELIMITER '|' ,
        FILE_PATH '/exchange/flatfiles/customer.txt');

-----
----- XML -----
CREATE WRAPPER "XML" LIBRARY 'libdb2lxml.a';

CREATE SERVER XMLSERV WRAPPER "XML";

```



```

CREATE NICKNAME XML.REGION (
    R_REGIONKEY INTEGER NOT NULL OPTIONS(XPATH './r_regionkey/text()'),
    R_NAME CHARACTER (25) NOT NULL OPTIONS(XPATH './r_name/text()'),
    R_COMMENT VARCHAR (152) NOT NULL OPTIONS(XPATH './r_comment/text()'))
FOR SERVER "XMLSERV"
OPTIONS(XPATH '//region' ,
        FILE_PATH '/exchange/xml/region.xml');

```

```

CREATE NICKNAME XML.NATION (
    N_NATIONKEY INTEGER NOT NULL OPTIONS(XPATH './n_nationkey/text()'),
    N_NAME VARCHAR (25) NOT NULL OPTIONS(XPATH './n_name/text()'),
    N_REGIONKEY INTEGER NOT NULL OPTIONS(XPATH './n_regionkey/text()'),
    N_COMMENT VARCHAR (152) NOT NULL OPTIONS(XPATH './n_comment/text()'))
FOR SERVER "XMLSERV"
OPTIONS(XPATH '//nation' ,
        FILE_PATH '/exchange/xml/nation.xml');

```

```

CREATE NICKNAME XML.SUPPLIER (
    S_SUPPKEY INTEGER NOT NULL OPTIONS(XPATH './S_SUPPKEY/text()'),
    S_NAME VARCHAR (25) NOT NULL OPTIONS(XPATH './S_NAME/text()'),
    S_ADDRESS VARCHAR (40) NOT NULL OPTIONS(XPATH './S_ADDRESS/text()'),
    S_NATIONKEY INTEGER NOT NULL OPTIONS(XPATH './S_NATIONKEY/text()'),
    S_PHONE VARCHAR (15) NOT NULL OPTIONS(XPATH './S_PHONE/text()'),
    S_ACCTBAL DOUBLE NOT NULL OPTIONS(XPATH './S_ACCTBAL/text()'),
    S_COMMENT VARCHAR (101) NOT NULL OPTIONS(XPATH './S_COMMENT/text()'))
FOR SERVER "XMLSERV"
OPTIONS(XPATH '//supplier' ,
        FILE_PATH '/exchange/xml/supplier.xml');

```

-----  
 ----- EXCEL Files -----

```

-- Create ODBC Wrapper:
CREATE WRAPPER "ODBCXLS" LIBRARY 'libdb2rcodbc.a'
OPTIONS( ADD MODULE '/openlink/lib/oplodb2.so');

-- Create ALLXLS Server:
CREATE SERVER ALLSRV TYPE ODBC VERSION '3.0' WRAPPER "ODBCXLS"
OPTIONS( ADD NODE 'ALLXLS', PASSWORD 'N');

-- Create Nicknames:
CREATE NICKNAME XLS."CUSTOMER$" FOR ALLSRV."Customer$";
--CREATE NICKNAME XLS."LINEITEM$" FOR ALLSRV."LineItem$";
CREATE NICKNAME XLS."NATION$" FOR ALLSRV."Nation$";
--CREATE NICKNAME XLS."ORDERS$" FOR ALLSRV."Orders$";
CREATE NICKNAME XLS."PART$" FOR ALLSRV."Part$";

```

```
CREATE NICKNAME XLS."PARTSUPP$" FOR ALLSRV."PartSupp$";
CREATE NICKNAME XLS."REGION$" FOR ALLSRV."Region$";
CREATE NICKNAME XLS."SUPPLIER$" FOR ALLSRV."Supplier$";
```

---

## A.3 Data sources/wrapper/server/schema names

As a summary for A.2, “Script for building the case study” on page 382, Table A-1 lists all names for wrappers, servers, and schemas used in our case study.

*Table A-1 Data source - Wrapper-Server-Schema name - Matrix*

Data source	Wrapper name	Server name	Schema name
DB2 UDB	DB2DRDA	DB2UDB	DB2DAT
DB2 for z/OS	DB2DRDA	DB2ZSERIES	DB2ZOS
DB2 for iSeries	DB2DRDA	DB2ISERIES	DB2OS400
Informix	INFORMIX	IDS9	INFX
Oracle	ORACLE	ORACLE9	ORA
Microsoft SQL Server	MSSQL	MSSQL2000	MSSQL
Table structured files	FLATFILES	FLATSERV	FLAT
XML	XML	XMLSERV	XML
Excel	ODBCXLS	ALLSRV	XLS

## A.4 DB2 list db directory

In Example A-3 we list our System Database Directory.

*Example: A-3 System Database Directory*

---

Database 1 entry:

Database alias	= FEDDB
Database name	= FEDDB
Local database directory	= /datafed
Database release level	= a.00
Comment	=
Directory entry type	= Indirect
Catalog database partition number	= 0

Database 2 entry:

Database alias	= DCS0S400
Database name	= DCS0S400
Node name	= DB20S400
Database release level	= a.00
Comment	=
Directory entry type	= Remote
Authentication	= SERVER
Catalog database partition number	= -1

Database 3 entry:

Database alias	= DATDB
Database name	= DATDB
Node name	= DB2DAT64
Database release level	= a.00
Comment	= Datdb
Directory entry type	= Remote
Authentication	= SERVER
Catalog database partition number	= -1

Database 4 entry:

Database alias	= DCSDB2G
Database name	= DCSDB2G
Node name	= DB2ZSRV
Database release level	= a.00
Comment	=
Directory entry type	= Remote
Authentication	= SERVER
Catalog database partition number	= -1

---

## A.5 DB2 list dcs directory

In Example A-4 we list our Database Connection Services (DCS) directory. The number of entries in the directory is two.

*Example: A-4 DCS directory*

---

DCS 1 entry:

Local database name	= DCSDB2G
Target database name	= DB2G
Application requestor name	=
DCS parameters	=

Comment	=
DCS directory release level	= 0x0100

DCS 2 entry:

Local database name	= DCS0S400
Target database name	= ITS0
Application requestor name	=
DCS parameters	=
Comment	=
DCS directory release level	= 0x0100

---

## A.6 DB2 list Node Directory

In Example A-5 we list our Node Directory. The number of entries in the directory is three.

*Example: A-5 Node Directory*

---

Node 1 entry:

Node name	= DB2DAT64
Comment	=
Directory entry type	= LOCAL
Protocol	= LOCAL
Instance name	= db2dat64

Node 2 entry:

Node name	= DB20S400
Comment	=
Directory entry type	= LOCAL
Protocol	= TCPIP
Hostname	= 9.5.92.35
Service name	= 446

Node 3 entry:

Node name	= DB2ZSRV
Comment	=
Directory entry type	= LOCAL
Protocol	= TCPIP
Hostname	= 9.12.6.8
Service name	= 33378

---

## A.7 /home/informix/ids92/etc/sqlhosts

In Example A-6 we list our Informix etc/sqlhosts file.

*Example: A-6 etc/sqlhosts*

---

```
demo_on  onsoctcp  nanping  30001
```

---

## A.8 /oracle9ir2/network/admin/tnsnames.ora

In Example A-7 we list our Oracle tnsnames.ora file.

*Example: A-7 tsnames.ora*

---

```
ORAFED.ALMAIDEN.IBM.COM =  
  (DESCRIPTION =  
    (ADDRESS_LIST =  
      (ADDRESS = (PROTOCOL = TCP)(HOST = lead.almaiden.ibm.com)(PORT = 1521))  
    )  
    (CONNECT_DATA =  
      (SERVICE_NAME = orafed.II.almaiden.ibm.com)  
    )  
  )
```

---

## A.9 /home/db2fed32/.odbc.ini

In Example A-8 we list the contents of our odbc.ini file.

*Example: A-8 .odbc.ini*

---

```
[ODBC Data Sources]  
RADONSQL=DataDirect 4.20 SQL Server Wire Protocol  
ALLXLS      = OpenLink Generic ODBC Driver  
  
[ODBC]  
IANAAppCodePage=4  
InstallDir=/datadirect/odbc4.2  
Trace=0  
TraceDll=/datadirect/odbc4.2/lib/odbctrac.so  
TraceFile=odbctrace.out  
UseCursorLib=0  
  
[RADONSQL]  
Driver=/datadirect/odbc4.2/lib/ivmsss19.so  
Description=DataDirect 4.20 SQL Server Wire Protocol
```

```

Address=radon,1433

[Default]
Driver = /openlink/lib/oplodbcs.so

[Communications]
BrokerTimeout = 30
ReceiveTimeout = 120
RetryTimeout = 5
ReceiveSize = 16000
SendSize = 4096
ShowErrors = Yes
DataEncryption = No

[ODBC]
DebugFile = /tmp/openlink.log

[ALLXLS]
Description = Excel all tables
Host = radon:5000
ServerType = ODBC
Driver = /openlink/lib/oplodbcs.so
Database = ALL
ReadOnly = Yes
NoLoginBox = No
FetchBufferSize = 99

```

---

## A.10 /home/db2fed32/sqllib/cfg/db2dj.ini

In Example A-9 we list the contents of our db2dj.ini file.

*Example: A-9 db2dj.ini*

---

```

INFORMIXDIR=/home/informix/ids94
INFORMIXSERVER=demo_on
ORACLE_HOME=/oracle9ir2
DJX_ODBC_LIBRARY_PATH=/datadirect/odbc4.2/lib
ODBCINI=/home/db2fed32/.odbc.ini

```

---

## A.11 ldb2set -all

In Example A-10 we list the contents of our db2set file.

*Example: A-10 db2set -all*

---

```
[i] DB2LIBPATH=/datadirect/odbc4.2/lib
[i] DB2ENVLIST=LIBPATH
[i] DB2COMM=tcPIP
[i] DB2AUTOSTART=YES
[g] DB2SYSTEM=baltic
[g] DB2ADMINSERVER=dasusr1
```

---

## A.12 env

In Example A-11 we list our DB2 env file.

*Example: A-11 env*

---

```
_=/usr/bin/env
LANG=en_US
LOGIN=db2fed32
PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:/home/db2fed32/bin:/usr/bin/X11:/sbin:./
/home/db2fed32/sqllib/bin:/home/db2fed32/sqllib/adm:/home/db2fed32/sqllib/misc
LC_FASTMSG=true
CGI_DIRECTORY=/var/docsearch/cgi-bin
EDITOR=vi
CLASSPATH=/home/db2fed32/sqllib/java/db2java.zip:/home/db2fed32/sqllib/java/db2
jcc.jar:/home/db2fed32/sqllib/java/sqlj.zip:/home/db2fed32/sqllib/function:/hom
e/db2fed32/sqllib/java/db2jcc_license_cisuz.jar:/home/db2fed32/sqllib/java/db2j
cc_license_cu.jar:.
LOGNAME=db2fed32
MAIL=/usr/spool/mail/db2fed32
LOCPATH=/usr/lib/nls/loc
PS1=${WHOAMI}:${PWD}>
VWSPATH=/home/db2fed32/sqllib
DOCUMENT_SERVER_MACHINE_NAME=localhost
USER=db2fed32
AUTHSTATE=compat
DEFAULT_BROWSER=netscape
SHELL=/usr/bin/ksh
ODMDIR=/etc/objrepos
DOCUMENT_SERVER_PORT=49213
HOME=/home/db2fed32
DB2INSTANCE=db2fed32
LD_LIBRARY_PATH=./home/db2fed32/sqllib/lib
TERM=ansi
```

```
MAILMSG=[YOU HAVE NEW MAIL]
ITECONFIGSRV=/etc/IMNSearch
PWD=/home/db2fed32
DOCUMENT_DIRECTORY=/usr/docsearch/html
TZ=PST8PDT
ITECONFIGCL=/etc/IMNSearch/clients
ITE_DOC_SEARCH_INSTANCE=search
A_z=! LOGNAME
LIBPATH=/usr/lib:/lib:/home/db2fed32/sql/lib/lib
NLSPATH=/usr/lib/nls/msg/%L/%N:/usr/lib/nls/msg/%L/%N.cat
```

---



# Unicode tutorial

This appendix provides a brief introduction to the Unicode encoding.

## B.1 Unicode

Fundamentally, computers just deal with numbers. They store letters and other characters by assigning a number for each one. Before Unicode was invented, there were hundreds of different encoding systems for assigning these numbers. No single encoding system could contain enough characters. The European Union alone requires several different encodings to cover all its languages. Even for a single language like English, no single encoding was adequate for all the letters, punctuation, and technical symbols in common use.

These legacy encoding systems conflict with one another; that is, two encodings can use the same number for two different characters or use different numbers for the same character. Moreover, for encodings such as Shift-JIS (a popular encoding in Japan), there are dozens of different variants. Even if a program knows that the encoding is Shift-JIS, there is usually no way to find out at run time precisely which variant is being referenced. Any given computer, especially any server, needs to support many different encodings; yet, whenever data is passed between different encodings or platforms, that data always runs the risk of corruption.

Unicode was invented to address this situation. It provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language. The Unicode Standard has been adopted by such industry leaders as Apple, HP, IBM, JustSystem, Microsoft, Oracle, SAP, Sun, Sybase, and many others. Unicode is required by modern standards such as XML, Java, ECMAScript (JavaScript), LDAP 3.0, CORBA 3.0, WML, etc. It is the official way to implement ISO/IEC 10646. It is supported in many operating systems, all modern browsers, and many other products.

Incorporating Unicode into client-server or multitiered applications and Web sites offers significant cost savings over the use of legacy encoding systems. Unicode enables a single software product or a single Web site to be targeted across multiple platforms, languages, and countries without re-engineering. It allows data to be transported through many different systems without corruption. Even where the client applications still depend on legacy encodings, Unicode can be used as a “lingua franca” on the server and database side, so that the user’s data can always be stored without corruption, no matter what the original encoding.

There are three different ways to encapsulate Unicode for use on a system: UTF-8, UTF-16, and UTF-32. Each of the UTFs can be useful in different environments. For systems that only offer 8-bit strings currently, but are multibyte enabled, UTF-8 may be the best choice. For systems that do not care about storage requirements, UTF-32 may be best. For systems such as Windows, Java, or ICU that use UTF-16 strings already, UTF-16 is the obvious choice. The

XML specification requires that all conformant XML parsers accept both UTF-8 and UTF-16.

*Table B-1 Unicode specifications and requirements*

UTF	Length per character	Estimated average storage required per page (3000 characters)
UTF-8	1, 2, 3, or 4 bytes	3 KB (1999) 5 KB (2003)  On average, English takes slightly over one unit per code point. Most Latin-script languages take about 1.1 bytes. Greek, Russian, Arabic, and Hebrew take about 1.7 bytes, and most others (including Japanese, Chinese, Korean and Hindi) take about 3 bytes.
UTF-16	2 or 4 bytes	6 KB  All of the most common characters in use for all modern writing systems are already represented with 2 bytes. Characters in surrogate space take 4 bytes, but as a proportion of all world text they will always be very rare.
UTF-32	4 bytes	12 KB  All take 4 bytes.

Conversion from one UTF to another is extremely fast. Unlike converting to and from legacy encodings like Latin-2, conversion between UTFs does not require table lookups. Since the conversion between the different UTFs is so rapid, all programs are strongly encouraged to interchange all of the UTFs, regardless of which UTF is used internally.

In general, components that communicate data to outside systems must be prepared to handle any of the UTF formats on input and output, as well as the common legacy code pages. Internal storage can be limited to a particular UTF, and communication between components of a multitier system can be limited to a particular UTF. For example, when using such components written in Java or using ICU, two components can communicate together with UTF-16. If a component were to communicate with an outside system, it must be prepared to convert the encoding that the outside system uses into and out of UTF-16.

For more detailed information about IBM work with Unicode, see:

<http://www-4.ibm.com/software/developer/library/utfencodingforms/index.html>

and some other articles on:  
<http://www.ibm.com/developer/unicode/>

Information is also available on the Unicode Web site at:  
<http://www.unicode.org>

The following document provides a detailed description of Unicode support as stated from a high level perspective in this architecture imperatives document.

## **International Components for Unicode (ICU)**

Supporting Unicode correctly and efficiently is important for e-business applications. It is also complex. Recognizing these facts, IBM has been supporting an open-source cross-platform C/C++/Java library for supporting Unicode, called International Components for Unicode (ICU). This effort has been received enthusiastically by the development community both inside and outside IBM. A number of fully-tested reference releases have been posted and used in a wide variety of projects. IBM platforms have all committed to making ICU services available on their platforms. ICU includes support for working with Unicode strings (collation, iteration, character classification) as well as efficient conversions to and from a very wide set of other encodings. ICU also includes Unicode-based locale support for standard locale-related features, such as formatting and parsing dates, times, currencies, and other numeric formats. The code is available in three different forms: as a C-callable library, as a C++ library, and as extensions to the JRE internationalization classes in Java.

The data to support the conversions, the locale-specific operations, and the general Unicode manipulations is available in independent localization packs.

The Globalization whitepaper currently calls for all IBM platforms to provide access to ICU. Complete compliance is planned by the middle of 2001. For platforms that do not provide this support, programs can install ICU themselves. The ICU design allows applications to control their own ICU files, so multiple applications can either choose to use the platform ICU, their own ICU, or an ICU that is shared with other cooperative applications.

More information can be found at:  
<http://oss.software.ibm.com/icu>

## **Unicode levels**

In the IBM e-business globalization architecture, programs can achieve Unicode compliance at two different levels: simple data integrity or complete standards compliance across a broad character repertoire. Programs should support the complete level, but as an intermediate step it is important to support at least the simple level.

## No data corruption

For basic data integrity, programs must be able to store and retrieve Unicode character data without losing data. Since Unicode contains a much larger set of characters than legacy character sets do, data integrity is simplest to achieve if the native character encoding used by the implementation is Unicode itself. It is possible to achieve data integrity with other native encodings; for example, an XML file can be in a different encoding but still provide data integrity by representing Unicode with numeric character references (NCR). So, `&#x3B1;` can be used to represent a Greek alpha. However, analyzing and manipulating the characters in such operations as sorting is done far more efficiently if Unicode is used as a native encoding technique.

“No data loss” does not necessarily equal “no data change.” In some cases, Unicode allows multiple ways of expressing the same character. These are called *canonical equivalents*. Programs may change characters from one canonical equivalent form to another. This may happen as the result of moving the Unicode characters through standard processing libraries such as ICU. For example, the sequence *latin small letter a* (Unicode 0061) followed by *combining acute accent* (Unicode 0301) may be transformed into *latin small letter a with acute* (Unicode 00E0).

Note that some systems will produce *fallbacks* when mapping from Unicode to legacy character encodings. For example, many of the Windows mappings will use a pound symbol to represent a lira if the lira sign is not available in the target encoding. This represents a corruption of the data unless it is used only for display.

## Unicode Standards Compliance

Programs that need to operate on Unicode strings, rather than simply storing and passing them on, have a higher compliance level to achieve. The requirements for Unicode-compliant programs are spelled out extensively in the Unicode standard, especially in the section on conformance. This section of the standard describes how Unicode values can be interpreted and transformed in processes that implement Unicode.

## Repertoire

The Unicode standard allows programs to support any non-empty subset of the Unicode character repertoire and still be considered compliant. However, support for e-business requires support for all of the characters used in all of the world's major languages. The consistent use of Unicode libraries such as ICU will allow programs to process the entire repertoire of Unicode values in compliance with the Unicode standard. Programs that need to interact directly with end-users will need additional support to work with some ranges of Unicode values that require specialized input or output methods.

## Native Unicode

To achieve standards compliance with a broad Unicode character repertoire requires programs to use some internal encoding that can encompass that repertoire without many-to-one round-tripping losses. As well, correctly implementing many of the Unicode algorithms is far more straightforward using actual Unicode values rather than other encodings. Finally, using Unicode internally allows programs to easily take advantage of other standard libraries and programs that work with Unicode, such as ICU. For all of these reasons, the architecture team strongly recommends the use of Unicode as the encoding system for internal processing of strings, in addition to being the encoding system used at external boundaries. This consistent use of Unicode for both program data and user data, at the API level and internally, is sometimes called Native Unicode. We highly recommend the use of Native Unicode for efficiency and consistent program operation.

The ability to successfully use a Native Unicode strategy in a program sometimes depends on the Unicode support of the platform the program is running on. Programs written for platforms that have broad support for Unicode, such as Windows NT® or Java, should take advantage of that support to become native Unicode applications. Programs that run on non-Unicode platforms, which need to rely on platform support for processing some of their textual data, may need to make compromises. Such programs may need to limit the range of Unicode repertoire they support to stay within limits imposed by the platform. Or, for performance reasons, they may need to use an encoding that matches the platform encoding. Or both sorts of compromises may need to be made. Development teams facing these sorts of limits should explore short-term alternatives, such as using non-platform services for Unicode support. Use of portable services such as ICU is one good solution that is available now. All of the IBM platforms are finding ways to make ICU services available, so programs that write to the ICU interfaces can use platform services with minimal changes when ICU services are available on the platform.

## File systems

File systems that are accessible to clients should allow arbitrary Unicode characters in file names (except syntax characters such as path delimiters). While this capability is not required for middleware and server file systems, it is recommended since it prevents errors when there is a mismatch between the abilities of programs like Java (which can use Unicode file systems) and the native system. To promote interoperability, both \ and / should be acceptable as delimiters.

## Data conversions

Unicode provides a way of encompassing the data that is currently in legacy code pages. However, it is vital to know exactly which code page is being used in

conversion. Otherwise, programs will get the wrong result. Unfortunately, code pages do not have unique names. There are a good dozen code pages that all advertise themselves as being Shift-JIS. Choose the wrong ones and data corruption results. As an added complication, the contents of many code pages change over time.

To deal with these problems, IBM needs absolutely precise information about the mappings used on its own platforms, and on all other platforms where data might originate or might end up. We require a compilation of code page mapping data that:

- ▶ Is source-code controlled, with versioning and archival capabilities
- ▶ Is uniquely identified
- ▶ Is programmatically accessible over the Internet
- ▶ Serves up XML format for data
- ▶ Has content that is rigorously cross-checked against all IBM platforms and non-IBM platforms (for instance Windows, Java) to validate the data
- ▶ Includes information for converting visual-order and visual-shaping BiDi characters to Unicode and back

### Encapsulating Unicode

Document formats such as XML or HTML and programming languages such as Java or Perl can provide a mechanism that allows for all Unicode characters to be represented, even if the file encoding is a legacy encoding, using a so-called “escaped-form.” For example, the Unicode character hex 2029 can be represented in XML or HTML as `&#x2029;` in Java as `\u2029`, and in Perl as `\x{2029}`. Such escape mechanisms can also be used in a minimal Unicode implementation for encapsulating Unicode text for legacy programs. It does, however, require considerably more storage (6-7 bytes per Unicode character) if more than a few Unicode characters are escaped.

## B.2 Locale

A *locale* defines the subset of a user’s environment that depends on language and cultural convention.

An internationalized system has no built-in assumptions or dependencies on code set, character classification, character comparison rules, character collation order, monetary formatting, numeric punctuation, date and time formatting, or the text of messages.

Because of cultural variety, people have different expectations about the results of many of the most basic computer-human interactions. The extent to which global software is possible is the extent to which these differences can be reasonably specified in ways that can be dealt with by programs.

One type of specification is needed for interactions for which expectations vary. Various computer languages and associated standards bodies have categorized culturally-expected computer behavior. They have addressed problems such as numeric formatting, the language used in the UI, and the results of comparing and sorting.

The second type of specification is needed for people whose expectations vary. Some of the most important dimensions along which expectations vary are language and country. Because these differences tend to be distributed in geographic clumps, the idea of locale has been borrowed from geography. Locale is used in the software industry in general to mean any of the following related concepts:

- ▶ The set of people who share a set of common expectations about their computer interactions
- ▶ The common expectations of computer behaviors that those people share
- ▶ The name given to one of those particular sets of expectations or people
- ▶ The computer-readable data (and sometimes code) that encapsulates those behaviors

A locale model contains assumptions about all four of these aspects of locale. In particular, any adequate locale model used in a global e-business system needs to meet these requirements.

Locale is defined by these language and cultural conventions. An internationalized system processes information correctly for different locations. For example, in the United States, the date format, 9/6/2002, is interpreted to mean the sixth day of the ninth month of the year 2002. The United Kingdom interprets the same date format to mean the ninth day of the sixth month of the year 2002. The formatting of numeric and monetary data is also country-specific, for example, the U.S. dollar and the U.K. pound. Rules for uppercase and lowercase usage also vary according to the language and the country.

All locale information must be accessible to programs at run time so that data is processed and displayed correctly for your cultural conventions and language. This process is called localization. Localization consists of developing a database containing locale-specific rules for formatting data and an interface to obtain the rules.



## B.2.1 Understanding locale

The locale model accounts for at least language and country, and it has some additional ways of specifying variants. It includes support for the major categories of locale-dependent computing. It provides for hierarchical fall-back behavior at either the source or run-time levels. It allows different locales to be set per client. For multi-client server software, this means there must be a way to have different locale processing for each client context (which may be per thread, depending on the cIA locale comprises the language, territory, and code set combination used to identify a set of language conventions). These conventions include information on collation, case conversion, and character classification, the language of message catalogs, date-and-time representation, the monetary symbol, and numeric representation.

Locale information contained in the locale definition source files must first be converted into a locale database by the **localdef** command. The **setlocale** subroutine can then access this information and set locale information for applications. To deal with locale data in a logical manner, locale definition source files are divided into six categories. Each category contains a specific aspect of the locale data. The **LC\_\*** environment variables and the **LANG** environment variable can be used to specify the desired locale. For more information on locale categories, see Appendix B.2.1, “Understanding locale” on page 403.

### Typical user scenarios

Users might encounter several NLS-related scenarios on the system. This section lists common scenarios with suggested actions to be taken.

- ▶ User keeps default code set

The user might be satisfied with the default code set for language-territory combinations even where more than one code set is supported. The user might keep the default code set if the current user environment uses that code set, or if the user is new and has no code set preference.

The language-territory selected at system installation time is defaulted to the appropriate locale based on the default code set. The default keyboard mappings, default font, and message catalogs are all established around the default code set. This scenario requires no special action from the user.

- ▶ User changes code set from the default code set

Users of a Latin-1 or Japanese locale might want to migrate their data and NLS environment to a different (non default) code set. This can be done in the following fashion:

- When the user has existing data that requires conversion

Flat text files that require conversion to the preferred code set can be converted through the Users application in Web-based System Manager,

the SMIT Manage the Language Environment® menu, or the **iconv** utility. User-defined structured files require conversion through user-written conversion tools that use the iconv library functions to convert the desired text fields within the structured files.

- When the user wants to change to the other code set

Where more than one code set is supported for a language-territory combination, the user may change to a non default locale by using:

- The user's application in Web-based System Manager
- The SMIT Manage Language Environment menu
- The **chlang**, **chkbd**, and **chfont** commands

## Locale naming conventions

Each locale is named by its locale definition source file name. These files are named for the language, territory, and code set information they describe. The following format is used for naming a locale definition file:

```
language[_territory][.codeset][@modifier]
```

For example, the locale for the Danish language spoken in Denmark using the ISO8859-1 code set is `da_DK.ISO8859-1`. The `da` stands for the Danish language and the `DK` stands for Denmark. The short form of `da_DK` is sufficient to indicate this locale. The same language and territory using the ISO8859-15 code set is indicated by `da_DK.8859-15`.

System-defined locale definition files are provided to show the format of locale categories and their keywords. The `/usr/lib/nls/loc` directory contains the locale definition files for system-defined locales. The C, or POSIX, locale defines the ANSI C-defined standard locale inherited by all processes at startup time. To obtain a list of system-defined locale definition source files, enter the following on the command line:

```
/usr/lib/nls/lsmle -c
```

## Installation default locale

The installation default locale refers to the locale selected at installation. For example, when prompted, a user can specify the French language as spoken in Canada during the installation process. The code set automatically defaults to the ISO8859-1 code set. With this information, the system sets the value of the default locale, specified by the `LANG` environment variable, to `fr_CA` (fr for ISO8859-1 French and CA for Canada). Every process uses this locale unless the `LC_*` or `LANG` environment variables are modified. The default locale can be changed by using the Manage Language Environment menu in SMIT. For more information see System Management Interface Tool (SMIT) Overview in AIX

5L™ Version 5.2 System Management Concepts: Operating System and Devices.

There are two basic ways to name locales programmatically:

- ▶ A string, with delimiters to separate the different parts
- ▶ An integer, with certain bits for each part

The POSIX (Portable Operating System Interface) standard uses delimited strings for locale names. For examples: `de` specifies just the language (German), while `de_AT` also specifies a country (German, Austria). For more on the POSIX locale standards, see:

<http://www.opennc.org/onlinepubs/7908799/xbd/locale.html>

In Internet protocols, instead of an underscore a hyphen is used. ISO standards 639-2 for language codes and 3166 for country codes are widely used as the basis for locale names. ICU and Java use the same basic mechanism for identification of locales. The Unicode Web site has a comparison chart with ISO, Windows, and Macintosh country and language codes, at:

<http://www.unicode.org/onlinedat/online.html>

It also has a list of languages and the scripts used to write them.

Microsoft Windows used 32-bit integers for locales. These have 10 bits for the language code, 6 bits for the country code, and 16 bits for variants like special collations. There are fewer such codes defined by Microsoft than there are abbreviations defined in the ISO standards and Microsoft controls this registry. If there is no code defined, then a certain locale cannot be supported portably. Microsoft .NET introduces the use for RFC 3066-based locale IDs, which look much like the POSIX or Java/ICU locale IDs.

DB2 for z/OS uses the information associated with a locale to execute UPPER, LOWER, and TRANSLATE functions in a culturally correct manner.



# Glossary

**Compensation.** The ability of DB2 to process SQL that is not supported by a data source.

**Database instance.** A specific independent implementation of a DBMS in a specific environment.

**Dynamic SQL.** SQL that is interpreted during execution of the statement.

**Extenders.** The program modules that provide extended capabilities for DB2, and are tightly integrated with DB2.

**Federation.** Providing a unified interface to diverse data.

**Function mapping.** The mapping between a data source function and a DB2 UDB function.

**Function template.** A virtual function definition for a data source function that cannot be executed on DB2.

**Index specification.** An index catalog entry for a nickname.

**Instance.** A complete database environment

**Materialized query table.** A table where the results of a query are stored, for later reuse.

**Nickname.** An identifier that is used to reference the data set located at the data source, mapped to rows and columns in DB2.

**Optimization.** The capability to enable a process to execute and perform in such a way as to maximize performance, minimize resource utilization, and minimize the process execution response time delivered to the end user.

**Option.** An additional attribute specific to each data source to customize an object.

**Passthru.** A special session mode that allows users to submit SQL statements directly to a data source without being changed by the federation support.

**Pushdown.** The act of optimizing a data operation by pushing the SQL down to the lowest point in the federated architecture where that operation can be executed.

**Server.** A specific data source, such as a database on a DB2 instance.

**Shared nothing.** A data management architecture where nothing is shared between processes. Each process has its own processor, memory, and disk space.

**Static SQL.** SQL that has been compiled prior to execution. Typically provides best performance.

**Type mapping.** The mapping of a specific data source type to a DB2 UDB data type.

**User mapping.** The information needed to connect to a specific server.

**Virtual database.** A federation of multiple heterogeneous relational databases.

**Wrapper.** The wrapper code module. The means by which a data federation engine interacts with heterogeneous sources of data. Wrappers take the SQL that the federation engine uses and maps it to the API of the data source to be accessed. For example, they take DB2 SQL and transform it to the language understood by the data source to be accessed.

**xtree.** A query-tree tool that allows you to monitor the query plan execution of individual queries in a graphical environment.



# Abbreviations and acronyms

<b>AIX</b>	Advanced Interactive eXecutive from IBM	<b>CM</b>	compatibility mode
<b>AMI</b>	application messaging interface	<b>COBOL</b>	Common Business Oriented Language
<b>APAR</b>	authorized program analysis report	<b>CPU</b>	central processing unit
<b>API</b>	application programming interface	<b>CRLF</b>	carriage return and line feed
<b>AQR</b>	automatic query re-write	<b>CSA</b>	common storage area
<b>AR</b>	access register	<b>CSV</b>	comma separated variables
<b>ARM</b>	automatic restart manager	<b>CTT</b>	created temporary table
<b>ART</b>	access register translation	<b>DAD</b>	document access definition
<b>ASCII</b>	American National Standard Code for Information Interchange	<b>DASD</b>	direct access storage device
<b>BI</b>	business intelligence	<b>DAT</b>	dynamic address translation
<b>BLOB</b>	binary large object	<b>DB2 PM</b>	DB2 performance monitor
<b>BO</b>	Business Objects	<b>DBA</b>	database administrator
<b>BTQ</b>	broadcast table queue	<b>DBAT</b>	database access thread
<b>CCA</b>	client configuration assistant	<b>DBCLOB</b>	double byte character large object
<b>CCSID</b>	coded character set identifier	<b>DBD</b>	database descriptor
<b>CD</b>	compact disk	<b>DBET</b>	database exception tables states
<b>CDR</b>	Call Detail Record	<b>DBID</b>	database identifier
<b>CEC</b>	central electronics complex	<b>DBMS</b>	database management system
<b>CF</b>	coupling facility	<b>DBRM</b>	database request module
<b>CFCC</b>	coupling facility control code	<b>DCL</b>	data control language
<b>CFRM</b>	coupling facility resource management	<b>DCS</b>	database connection services
<b>CICS</b>	Customer Information Control System	<b>DDCS</b>	distributed database connection services
<b>CLI</b>	call level interface	<b>DDF</b>	distributed data facility
<b>CLI</b>	call level interface	<b>DDL</b>	data definition language
<b>CLOB</b>	character large object	<b>DLL</b>	dynamic load library
<b>CLP</b>	command line processor	<b>DML</b>	data manipulation language
		<b>DNS</b>	domain name server

<b>DPSI</b>	data partitioned secondary index	<b>HPJ</b>	high performance Java
<b>DRDA</b>	distributed relational database architecture	<b>HTTP</b>	Hypertext Transport Protocol
<b>DSC</b>	dynamic statement cache, local or global	<b>I/O</b>	input/output
<b>DTD</b>	data type definition	<b>IBM</b>	International Business Machines Corporation
<b>DTT</b>	declared temporary tables	<b>ICF</b>	integrated catalog facility
<b>DWH</b>	data warehouse	<b>ICF</b>	integrated coupling facility
<b>EA</b>	extended addressability	<b>ICMF</b>	internal coupling migration facility
<b>EAR</b>	Enterprise Application Archive	<b>IDS</b>	Informix Dynamic Server
<b>EBCDIC</b>	extended binary coded decimal interchange code	<b>IFCID</b>	instrumentation facility component identifier
<b>ECS</b>	enhanced catalog sharing	<b>IFI</b>	instrumentation facility interface
<b>ECSA</b>	extended common storage area	<b>IM</b>	Intelligent Miner™
<b>EDM</b>	environment descriptor manager	<b>IMS</b>	Information Management System
<b>ENFM</b>	enabling new function mode	<b>IPLA</b>	IBM Program Licence Agreement
<b>ERP</b>	enterprise resource planning	<b>IRLM</b>	internal resource lock manager
<b>ESA</b>	Enterprise Systems Architecture	<b>IRWW</b>	IBM Relational Warehouse Workload
<b>ESE</b>	Enterprise Server Edition	<b>ISPF</b>	interactive system productivity facility
<b>ESS</b>	Enterprise Storage Server®	<b>ISV</b>	independent software vendor
<b>ETL</b>	extract transform load	<b>ITR</b>	internal throughput rate, a processor time measure, focuses on system capacity
<b>ETR</b>	external throughput rate, an elapsed time measure, focuses on system capacity	<b>ITSO</b>	International Technical Support Organization
<b>FQEP</b>	federated query execution plan	<b>IVP</b>	installation verification process
<b>FTP</b>	File Transfer Protocol	<b>J2EE</b>	Java 2 Enterprise Edition
<b>GB</b>	gigabyte (1,073,741,824 bytes)	<b>JDBC</b>	Java Database Connectivity
<b>GBP</b>	group buffer pool	<b>JFS</b>	journaled file systems
<b>GRS</b>	global resource serialization	<b>JIT</b>	just in time (Java compiler)
<b>GUI</b>	graphical user interface	<b>JNI</b>	Java Native Interface
<b>HA</b>	host adapter	<b>JVM</b>	Java Virtual Machine
<b>HFS</b>	hierarchical file system		



<b>KB</b>	kilobyte (1,024 bytes)	<b>PMML</b>	Predictive Model Markup Language
<b>LAN</b>	local area network	<b>PSID</b>	pageset identifier
<b>LCU</b>	logical control unit	<b>PSP</b>	preventive service planning
<b>LDAP</b>	Lightweight Directory Access Protocol	<b>PTF</b>	program temporary fix
<b>LOB</b>	large object	<b>PUNC</b>	possibly uncommitted
<b>LPAR</b>	Logical Partition	<b>QBIC®</b>	query by image content
<b>LPL</b>	logical page list	<b>QMF</b>	Query Management Facility
<b>LRECL</b>	logical record length	<b>RACF®</b>	Resource Access Control Facility
<b>LRSN</b>	log record sequence number	<b>RBA</b>	relative byte address
<b>LVM</b>	logical volume manager	<b>RBF</b>	Radial Basis Function
<b>MB</b>	megabyte (1,048,576 bytes)	<b>RDBMS</b>	Relational DataBase Management System
<b>MIME</b>	Multipurpose Internet Mail Extensions	<b>RECFM</b>	record format
<b>MIS</b>	Management Information System	<b>RID</b>	record identifier
<b>MQI</b>	MQSeries application programming interface	<b>ROI</b>	Return On Investment
<b>MQT</b>	materialized query table	<b>ROT</b>	rule of thumb
<b>MSM</b>	Multidimensional Storage Manager	<b>RPC</b>	Remote Procedural Call
<b>NFM</b>	new function mode	<b>RPD</b>	remote pushdown (operator)
<b>NLS</b>	national language support	<b>RR</b>	repeatable read
<b>NPI</b>	non-partitioning index	<b>RRSAF</b>	resource recovery services attach facility
<b>NVS</b>	Non-Volatile Storage	<b>RS</b>	read stability
<b>ODB</b>	object descriptor in DBD	<b>RSM</b>	Real Storage Manager
<b>ODBC</b>	Open Data Base Connectivity	<b>RTS</b>	real time statistics
<b>ODS</b>	Operational Data Store	<b>RVA</b>	RAMAC® Virtual Array
<b>OLAP</b>	Online Analytical Processing	<b>SBCS</b>	single-byte character set
<b>OLTP</b>	OnLine Transactional Processing	<b>SDK</b>	software developers kit
<b>OS/390®</b>	Operating System/390®	<b>SMIT</b>	System Management Interface Tool
<b>PAV</b>	Parallel Access Volume	<b>SMS</b>	Short Message (Text) Services received usually on a cellular phone
<b>PDS</b>	partitioned data set	<b>SMTP</b>	Simple Mail Transfer Protocol
<b>PIB</b>	parallel index build	<b>SOA</b>	Service-Oriented Architecture
<b>PITR</b>	point-in-time recovery		

<b>SOAP</b>	Simple Object Access Protocol
<b>SPE</b>	small program enhancement
<b>SQL</b>	Structured Query Language
<b>SVL</b>	IBM Silicon Valley Laboratory
<b>TCB</b>	Task control block
<b>UDB</b>	Universal DataBase
<b>UDDI</b>	Universal Description, Discovery, and Integration
<b>UDF</b>	User Defined Function
<b>UDT</b>	User Defined Type
<b>URL</b>	Uniform Resource Locator
<b>URN</b>	Uniform Resource Names
<b>USS</b>	UNIX System Services
<b>VSAM</b>	Virtual Storage Access Method
<b>W3C</b>	World Wide Web Consortium
<b>WAS</b>	WebSphere Application Service
<b>WLM</b>	Workload Manager
<b>WMQI</b>	WebSphere MQ Integrator
<b>WSAD</b>	WebSphere Studio Application Developer
<b>WSDL</b>	Web Services Description Language
<b>XML</b>	eXtensible Markup Language
<b>XSD</b>	XML Schema Definition or XML Schema Language

# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

## IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 415. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *A Practical Guide to DB2 Data Replication V8*, SG24-6828
- ▶ *Getting Started in Integrating Your Information*, SG24-6892
- ▶ *XML for DB2 Information Integration*, SG24-6994
- ▶ *IBM Life Sciences Solutions: Turning Data into Discovery with DiscoverLink*, SG24-6290
- ▶ *IBM Informix: Integration Through Data Federation*, SG24-7032
- ▶ *Moving Data Across the DB2 Family*, SG24-6905
- ▶ *The IBM Enterprise Information Portal: A Practical Approach*, SG24-6101
- ▶ *The IBM Enterprise Information Portal: A Primer*, SG24-5749
- ▶ *The IBM Enterprise Information Portal: A Cookbook*, SG24-6125

## Other publications

These publications are also relevant as further information sources:

- ▶ *IBM Systems Journal Vol. 41, No. 4, 2002, Information Integration*, G321-01473
- ▶ *IBM DB2 Information Integrator Developer's Guide Version 8*, SC18-7359
- ▶ *IBM DB2 Information Integrator Federated Systems Guide Version 8*, SC18-7364
- ▶ *IBM DB2 Information Integrator Installation Guide Version 8*, SC18-7036
- ▶ *IBM DB2 Information Integrator Migration Guide Version 8*, SC18-7360
- ▶ *IBM DB2 Information Integrator Solutions Guide Version 8*, SC18-7037

- ▶ *IBM DB2 Information Integrator Data Source Configuration Guide Version 8*, available as softcopy only from the Web site:  
<http://www.ibm.com/software/data/integration/solution>
- ▶ *IBM DB2 Information Integrator Installation Release Notes Version 8*, available as softcopy only from the Web site:  
<http://www.ibm.com/software/data/integration/solution>
- ▶ *IBM DB2 Universal Database Administration Guide: Planning Version 8*, SC09-4822
- ▶ *IBM DB2 Universal Database Command Reference, Version 8*, SC09-4828
- ▶ *IBM DB2 Universal Database Message Reference Volume 1 Version 8*, GC09-4840
- ▶ *IBM DB2 Universal Database Replication Guide and Reference Version 8*, Release 1, SC27-1121
- ▶ *IBM DB2 Universal Database SQL Reference Volume 1 Version 8*, SC09-4844
- ▶ *IBM DB2 Universal Database SQL Reference Volume 2 Version 8*, SC09-4845
- ▶ *IBM DB2 Universal Database System Monitor Guide and Reference Version 8*, SC09-4847
- ▶ *IBM Systems Journal Vol. 41, No. 4, 2002, Information Integration*, G321-01473
- ▶ *Using the federated database technology of IBM DB2 Information Integrator*, white paper by Anjali Grover, Eileen Lin and Ioana Ursu, available from the Web site:  
<http://www.ibm.com/software/data/pubs/papers/#iipapers>

## Online resources

These Web sites and URLs are also relevant as further information sources:

- ▶ DB2 Information Integration products  
<http://www.ibm.com/software/data/integration/solution>
- ▶ DB2 offerings information  
<http://www.ibm.com/common/ssi/>
- ▶ DB2 Information Integration support site  
<http://www.ibm.com/software/data/integration/db2ii/support.html>
- ▶ Systems Journal on Information Integration

<http://www.research.ibm.com/journal/sj41-4.html>

- ▶ IBM DiscoveryLink offering  
<http://www.ibm.com/solutions/lifesciences/solutions/discoverylink.html>
- ▶ DataDirect ODBC driver for UNIX 4.2  
<http://www.datadirect.com>
- ▶ OpenLink MultiTier ODBC Client 5.0  
<http://www.openlink.co.uk>
- ▶ Samba Server 2.2  
<http://www.samba.org>
- ▶ IBM developerWorks  
<http://www.ibm.com/developerworks/db2>

## How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, 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)



# Index

## A

- access plan 313
- ACCOUNTING\_STRING 101
- AIX xxiii, 428
- ASCII 294
- ASCII encoding 35
- AST 278
- AUTHENTICATION 88
- Automatic Summary Tables 278
- automation 5
- autonomic concepts 20

## B

- baltic 67
- blank padding 41
- BLOCKING ALL 281
- bottlenecks in federated environment 275
- buffer pool 272
- BUFFPAGE 91
- built-in function 29, 43
- business challenges 4
- business need 4
- Business Objects 31

## C

- cache 12
- case study 53
- cast 335
- CCSID 294
- character set 294
- code points 294
- codepage 294
- COLLATING\_SEQUENCE 261
- collating\_sequence 323
  - recommendation 327
- column cardinality 262
- column options 40, 45
- COMM\_RATE 250, 261, 280
- communication bandwidth 244
- compensation 246
- configuring DB2 for use with Microsoft SQL Server
- data sources 167

- configuring DB2 for use with Oracle data sources 167
- connectivity 123
- consolidation 10
- Control Center 13, 39, 87
- cost 10
- cost optimization 249
- COT(x) 246
- CPU 244
- CPU\_RATIO 249, 261, 280
- CREATE FUNCTION 343
- CREATE FUNCTION MAPPING 342
- CREATE NICKNAME
  - RANGE option 220
- CURRENCY 219
- CURRENT REFRESH AGE 369
- Custom Installation 75

## D

- data access 71
- data distribution 70
- DATA INITIALLY DEFERRED 369
- data model 53, 68
- data modelling 33
- data source 26
- data source client 122
- data type mapping 42, 334
- data type mappings 29
- data types
  - DB2 to ODBC wrapper 219
- database server 29
- database size 69
- DataDirect Connect for ODBC UNIX 122
- DataDirect ODBC driver 165
- DataJoiner 13, 19
- DATE 334, 338
- DATETIME 219
- DB2
  - CHAR 42
  - DATE 40
  - DECIMAL 42
  - DOUBLE 42
  - TIMESTAMP 42

- DB2 Command Line Processor 297
- DB2 Common Server V2 20
- DB2 Connect 55
- DB2 Control Center 73
- DB2 data types 219
- DB2 Enterprise Server Edition 26
- DB2 for AIX
  - altering DB2 for z/OS server definition and server options 98
- DB2 for z/OS data source 59
- DB2 for z/OS nicknames 102
- DB2 for z/OS server 94
- DB2 for z/OS server definition 98
- DB2 for z/OS user mappings 99, 101
- DB2 for z/OS wrapper 93
- DB2 Information Integrator 14
  - altering Microsoft SQL Server definition and server options 189
  - altering Microsoft SQL Server user mappings 192
  - altering ODBC nicknames 232
  - altering Oracle nicknames 181
  - altering Oracle server definition and server options 176
  - altering Oracle user mappings 178
  - altering table-structured file nicknames 217
  - altering the ODBC server 227
  - altering XML nicknames 209
  - components 26
  - configuring the wrappers 162
  - creating Microsoft SQL Server nicknames 192, 195
  - creating Microsoft SQL Server server 186
  - creating Microsoft SQL Server user mappings 190
  - creating Microsoft SQL Server wrapper 185
  - creating Oracle nicknames 179
  - creating Oracle user mappings 176
  - creating the ODBC nickname 228
  - creating the ODBC server 225
  - creating the ODBC wrapper 223
  - creating the Oracle server 172
  - creating the Oracle wrapper 171
  - creating the table-structured file nicknames 213
  - creating the table-structured file server 212
  - creating the table-structured file wrapper 212
  - creating the XML nicknames 199
  - creating the XML server 198
  - creating the XML wrapper 197
  - installation 125
  - installation hints and tips 159
  - integrating Microsoft Excel 218
  - integrating Microsoft SQL Server 2000 181
  - integrating table-structured (flat) files 209
  - integrating XML data source 195
  - maintenance and operational issues 232
  - non relational wrappers setup 142
  - offerings 25
  - overview 22
  - prerequisites 122
  - relational wrappers setup 130
  - response files examples 157
  - software prerequisites 122
  - trouble shooting 235
- DB2 Information Integrator for Content 15
- DB2 Information Integrator integrating Oracle 9i 168
- DB2 instance 137
- DB2 Net Search Extender 26
- DB2 on AIX
  - altering DB2 for z/OS nicknames 105
  - altering DB2 for z/OS user mappings 101
  - Altering IDS nickname 118
  - altering IDS server definition and server options 112
  - altering IDS user mappings 115
  - configuring for federation 87
  - creating DB2 for z/OS nicknames 102
  - creating DB2 for z/OS user mappings 99
  - creating IDS nicknames 115
  - creating IDS user mapping 113
  - creating the DB2 for z/OS server 94
  - creating the DB2 for z/OS wrapper 93
  - creating the IDS server 109
  - creating the IDS wrapper 108
  - defining Informix Dynamic Server support 85
  - installation instructions 76
  - installation requirements 75
  - integrating DB2 for z/OS 91
  - integrating DB2 UDB for iSeries 118
  - integrating Informix Dynamic Server 105
  - setting up 74
- DB2 snapshot monitor 276
- DB2 tools catalog 83
- DB2 UDB Enterprise Server Edition 56
- DB2 UDB for iSeries 118
- DB2\_DJ\_COMM 90
- DB2\_DJ\_INI 90



- DB2\_MAXIMAL\_PUSHDOWN 36, 98–99, 113, 176, 189–190
- db2\_maximal\_pushdown 318
- db2batch 276, 308, 352
- DB2CODEPAGE 297
- db2diag.log 237
- db2dj.ini 89, 123, 171, 238, 296
- DB2ENVLIST 90
- DB2EXFMT 255
- db2exfmt 285, 308–309
- db2expln 286
- db2isetup 137, 162
- db2iupdt 125
- db2level 238
- DB2LIBPATH 90
- db2look 238
- db2set 184
- db2set variables 90
- DDL for DB2 data source 377
- DELETE 251
- DETERMINISTIC 345
- Discover 115
- DiscoveryLink 13, 18
- DISPLAY 76
- distributed query performance 266
- distributed two-source query 264
- distribution transparency 11
- djxlink 106, 125, 160, 167, 235
- djxlink Oracle script 236
- DOUBLE 334
- DRDA 73
- DRDA wrapper 55
- dynexpln 286

## E

- EBCDIC encoding 35
- EBCDIC/ASCII conversion 31
- e-business 5
- ENABLE QUERY OPTIMIZATION 369
- encoding scheme 294
- enterprise federated system 25
- Enterprise Information Portal xxiv, 19
- environment
  - connecting a flat file data source 64
  - connecting Excel data source via ODBC 65
  - connecting Microsoft SQL Server 62
  - connecting Oracle data source 61
  - connecting XML data source 63

- including DB2 for z/OS and Informix data sources 59
- installing Information Integrator 60
- the final configuration 66
- error logging 160
- Excel 348
  - pushdown 349
- Excel data source 65
- Excel data types 219
- execution plan 265
- Explain 308
- Explain script 287
- Explain tables 286
- Explaining queries 287

## F

- FED\_NO\_AUTH 88
- FEDERATED 87
- federated data 8, 55
- federated data solution
  - configuring 51
- federated database 22, 29
- federated environment 53
- federated query compiler flow 247
- federated query execution plan 251
- federated server 30
- federated system 11, 22, 53–54, 73
  - architecture 23
- Federation 10
- FILE\_PATH 207
- FixPak 208
- FixPak 2 232
- FixPak 3 125, 209, 233
- flat file data source 64
- FLOAT 334
- FOR 230
- FOR FETCH ONLY 281
- fragment 246
- function mapping 43
- function template 345

## G

- get\_stats 277, 307, 312
- global catalog 43
- global catalog views 44
- global catalog 29
- good query 253
- GRANT 37, 262

grid computing 3, 6

## H

health monitor 84  
Heterogeneity transparency 11  
high2key 262  
HSJOIN 259

## I

I/O 244  
iisetaup.bin 134, 148  
IMS xxiv, 17  
index specification 356  
index statistics to nicknames for views 364  
Information 3  
information integration 3–4, 9  
Information Integrator  
    family of products 13  
Informix 20  
    BOOLEAN 42  
    DATETIME 42  
    MONEY 42  
Informix Client Installation Directory 85  
Informix Client SDK 87  
Informix data source 59  
Informix data source support 85  
Informix Dynamic Server 105  
Informix nicknames 115  
Informix server 109  
Informix user mappings 113  
Informix wrapper 108  
INSERT 251  
Installation hints 159  
installation requirements 75  
integration 5  
IO\_RATIO 249, 261, 280  
iSeries 55, 73, 88–89, 118

## J

join 24, 30, 32, 195, 210, 242–244, 247, 258  
join of five tables 259

## K

kit for Informix client software development 85  
kit for wrapper development 34

## L

lack of statistics 358  
Latency 10  
lead 68  
legacy sources 17  
license agreemen 127  
Life Sciences 17  
Life Sciences Data Connect 13, 18  
LIKE 35  
local system 54  
locale 296  
LOGICAL 219  
Lotus Extended Search xxiv, 16  
low2key 262

## M

mapping used defined functions 343  
materialized query tables 13, 25  
MGJOIN 259  
Microsoft Excel 218  
    supported by two wrappers 218  
Microsoft SQL Server 2000 181  
Microsoft SQL Server nicknames 192  
Microsoft SQL Server server 186  
Microsoft SQL server user mapping 190  
Microsoft SQL Server wrapper 185  
Microsoft SQL Server wrapper libraries 182  
Microsoft Visual Studio.Net 31  
move and join 270  
MQT 242  
MQT performance 273

## N

nanping 67  
Net.Search xxiv  
new index after nickname creation 357  
nickname 28, 38  
Nickname statistics  
    recommendations 315  
nickname statistics 262, 305  
nicknames for remote views 357  
NLJOIN 259  
non relational data sources 57  
non relational wrappers 26, 158  
    installation 143  
NUMBER 219  
NUMERIC\_STRING 40

## O

- ODBC data types 219
- ODBC nickname 228
- ODBC server options 227
- ODBC servers 225
- ODBC wrapper 218, 223, 348
- ODBC wrapper mapping to DB2 data types 218
- odbc.ini 183
- on demand 5
- Open Data Base Connectivity 123
- open standards 6
- optimize for n rows 282
- optimizer factors 245
- Oracle 34
  - COLLATE USING IDENTITY 35
  - DATE 40
- Oracle 9i 168
- Oracle 9i wrapper 171
- Oracle nicknames 179
- Oracle server 172
- Oracle server option 327
- Oracle server options 174
- Oracle SQL function
  - RPAD 326
- Oracle user mappings 176
- Oracle wrapper libraries 169
- ORACLE\_HOME 164
- outer join 246

## P

- partitioned tables 271
- pass through 32
- Passthru 48, 407
- passthru 32, 37, 343, 358
- Passthru privileges 45
- PATH 90
- patterns xxiv
- PDA 249
- performance
  - federated server options 260
- performance analysis of a single remote source query 263
- performance factors 242
- performance tutorial 239
- PLAN\_HINTS 176
- prerequisites 122
- protocol 26
- PUSHDOWN 226

- Pushdown 175, 188, 407
- pushdown 30, 36, 97, 111, 175, 188, 242–243, 245, 248, 251, 315, 348, 350
  - importance of 250
- pushdown analysis 249
- pushdown examples 244
- pushdown factors 243
- PUSHDOWN set to N 349
- PUSHDOWN set to Y 350

## Q

- QMF 297
  - default font 303
- query optimization 246
- query performance 242, 272
- query performance tuning 275
- query rewrite 247
  - example 258
- query tuning on a single data source 253

## R

- radon 68
- RANGE option 220
- read 23
- read access scenarios 23
- read only mode 71
- Redbooks Web site 415
  - Contact us xxviii
- REFRESH DEFERRED 369
- REFRESH IMMEDIATE 369
- REFRESH TABLE 369
- Relational Connect 13, 18
- relational data sources 56
- relational wrappers 26
  - configuring 162
  - installation 136
  - response file 157
- replication xxiv, 25, 45
- response file 157–158
- RETURN 255
- RMQTX 263
- RPAD 326
- RPD 255, 278
- RQRIOLBK 282
- rqriolbk 281
- RQUERY 255
- RTRIM 330
- RUNSTATS 277, 368

## S

Samba 63, 201, 211, 415  
schema name 70  
script for building the case study 382  
SELECT 251  
server 28, 34  
server options 35, 317  
SET SERVER OPTION 36, 98  
SHEAPTHRES 89  
SHIP 251, 255, 278  
snapshot monitor 276  
sored procedures 24  
SORTED 'Y' 352  
SORTED 'N' 351  
SORTED parameter, 348  
SORTHEAP 91, 272  
sortheap 276  
SQL server data source 62  
SQL\_BIT 219  
SQL\_DOUBLE 219  
SQL\_NUMERIC 219  
SQL\_TIMESTAMP 219  
SQL\_VARCHAR 219  
SQL0332N 300  
SQL1822N 237  
sqlhosts 85  
SQLSTATE=57017 300  
statistics 306  
status report 140  
STREAMING 208  
SUM 43  
SYSCAT.COLOPTIONS 44  
SYSCAT.COLUMNS 44  
SYSCAT.FUNCMAPOPTIONS 45  
SYSCAT.FUNCMAPPARMOPTIONS 45  
SYSCAT.FUNCMAAPPINGS 45, 342  
SYSCAT.FUNCTIONS 45  
SYSCAT.INDEXES 44  
SYSCAT.INDEXOPTIONS 44  
SYSCAT.KEYCOLUSE 44  
SYSCAT.PASSTHRUAUTH 45  
SYSCAT.ROUTINES 45  
SYSCAT.SERVEROPTIONS 44  
SYSCAT.SERVERS 44  
SYSCAT.TABLES 44  
SYSCAT.TABOPTIONS 44  
SYSCAT.TYEMAPPINGS 45  
SYSCAT.WRAPOPTIONS 44  
SYSCAT.WRAPPERS 44

SYSIBM.USERNAMES 88–89  
SYSSTAT 306  
SYSSTAT.COLUMNS 44, 262, 306  
SYSSTAT.INDEXES 44, 306  
SYSSTAT.TABLES 44, 306

## T

table-structured file nicknames 213, 217  
table-structured file server 212  
table-structured file wrapper 212  
table-structured files 209, 348  
    sorting 351  
team xxvi  
TEMP space 272  
template function 344  
TEMPSPACE1 91  
territory 296  
TEXT 219  
the environment 58  
TIMESTAMP 40, 334, 337  
tnsnames.ora 170  
tools 31  
tools catalog 83  
Trailing blanks 329  
Transparency 11  
transparency 11  
troubleshooting 235  
tuning query on multiple data sources 264

## U

UCS-2 301  
Unicode 294  
UPDATE 251  
update 23  
user defined functions 343  
user mapping 29, 37  
UTF-8 301  
utilizing an existing index 361

## V

VARCHAR 328  
VARCHAR\_NO\_TRAILING\_BLANKS 40, 261  
varchar\_no\_trailing\_blanks 327  
varchar\_no\_trailing\_blanks server option 327  
VARCHAR2 40  
virtualization 5  
Visual Explain 251, 278, 285

VSAM xxiv, 17

## **W**

Web service 17

Web Services xxiv

WebSphere xxiv

WebSphere MQ 17, 32

WebSphere Studio 31

Windows xxiii, 428

wrapper 28, 74

wrapper development kit 47

wrappers 33

write 25

WSAD 412

wtsc63 68

## **X**

XML 4, 195

memory requirement 208

XML data source 63

XML nicknames 199

XML server 198

XML wrapper 197

STREAMING option 208

XPATH 207

XQuery 20

## **Z**

z/OS xxiii, 55, 68, 73, 428





## Data Federation with IBM DB2 Information Integrator V8.1

(0.5" spine)  
0.475" <-> 0.875"  
250 <-> 459 pages









**Redbooks**

# Data Federation with IBM DB2 Information Integrator V8.1

**Introducing  
information  
integration and data  
federation**

**Setting up DB2  
Information  
Integrator  
environments**

**Operating and tuning  
with DB2 Federated  
Data**

Federation and integration are open technologies, which extend your investment on existing data, and relational or non relational data by bringing it all together in a single virtual location for on demand application access. IBM DB2 Information Integrator V8.1 is the framework supporting federated data solutions.

This IBM Redbook provides implementation guidelines for deploying a federated access solution with IBM DB2 Information Integrator V8.1. Topics include configuring, tuning, and debugging a federated system; use of federation in a business intelligence-like environment with read access to relational (DB2 family, Oracle, SQL Server) and non relational sources (flat files, XML, ODBC), and usage through queries or reporting tools.

This redbook will help you create a federated data solution with a case study, which accesses several data sources across the AIX, Windows, and z/OS platforms.

**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)