

Accelerating Data Transformation with IBM DB2 Analytics Accelerator for z/OS

Ute Baumbach

Patric Becker

Uwe Dennerer

Eberhard Hechler

Wolfgang Hengstler

Steffen Knoll

Frank Neumann

Guenter Georg Schoellmann

Khadija Souissi

Timm Zimmermann



Information Management



In partnership with
IBM Academy of Technology



International Technical Support Organization

**Accelerating Data Transformation with IBM DB2
Analytics Accelerator for z/OS**

December 2015

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

First Edition (December 2015)

This edition applies to DB2 for z/OS Version 10, IBM DB2 Analytics Accelerator for z/OS Version 4.1 PTF 5.1, IBM DB2 Analytics Accelerator for z/OS Version 5.1, QMF for z/OS Version 11.2.

© Copyright International Business Machines Corporation 2015. 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

Notices	vii
Trademarks	viii
IBM Redbooks promotions	ix
Preface	xi
Authors	xii
Acknowledgement	xiii
Now you can become a published author, too	xiv
Comments welcome	xiv
Stay connected to IBM Redbooks	xiv
Chapter 1. Analytics on an IBM z Systems environment	1
1.1 Traditional data warehouse and analytics concepts	3
1.1.1 Persistency layers	3
1.1.2 Information supply chain	3
1.1.3 Analytics on z Systems aspects	4
1.2 IBM Transaction and Analytics Processing	4
1.2.1 Logical data warehouse	5
1.2.2 Technical approaches	5
1.2.3 Real-time analytics on z Systems	6
1.2.4 Analytics on operational data	6
1.3 Transformation patterns	6
1.3.1 Traditional ETL	6
1.3.2 Accelerating data transformation	7
1.3.3 New transformation patterns	7
1.4 New use cases	8
1.4.1 Use case description	8
1.5 Strategic outlook	10
Chapter 2. Accelerator-only tables	11
2.1 Concepts and architecture	12
2.1.1 Non-accelerator DB2 table	13
2.1.2 Accelerator-shadow table	14
2.1.3 Accelerator-archived table and partition	15
2.1.4 Accelerator-only table (AOT)	15
2.2 Software level prerequisites	17
2.3 Syntax and capabilities	18
2.3.1 Creating accelerator-only tables	18
2.3.2 Inserting data into accelerator-only tables	19
2.3.3 Removing accelerator-only tables	21
2.3.4 Using accelerator-only tables in queries or DML statements	22
2.4 Transactional considerations	22
2.5 Lifecycle management	24
2.6 Limitations and restrictions	25
2.7 Performance considerations	26
2.7.1 INSERT SELECT performance considerations	29

Chapter 3. Use cases that are enabled by accelerator-only tables and in-database analytics	31
3.1 The four use cases of the DB2 Analytics Accelerator	32
3.2 How accelerator-only tables and in-database analytics extend use cases	33
3.3 Acceleration of existing business critical queries	34
3.4 Derive business insight from z/OS transaction systems	37
3.5 Reduce IT sprawl for analytics initiatives	40
3.6 Improve access to historical data and lower storage costs	43
3.7 Summary	46
Chapter 4. Multistep reporting	47
4.1 Concepts of multistep reporting	48
Chapter 5. Using IBM DB2 QMF to store query results and import tables	53
5.1 QMF and IBM z Systems	54
5.2 QMF for z/OS and DB2 Analytics Accelerator	55
5.2.1 Accelerator-only table support in QMF for z/OS V11.2	56
5.3 Running queries and saving results using a QMF procedure	58
5.3.1 Queries used in the sample scenario	58
5.3.2 Running the procedure in DB2 for z/OS	61
5.3.3 Running the procedure in DB2 for z/OS and DB2 Analytics Accelerator	62
5.3.4 Running the procedure in DB2 Analytics Accelerator using accelerator-only tables created by the SAVE DATA AS command	63
5.3.5 Running the procedure in DB2 Analytics Accelerator using accelerator-only tables created by the RUN QUERY command	65
5.4 Importing tables as accelerator-only tables	69
5.5 Preferred practices	70
Chapter 6. Accelerating IBM Campaign processing	73
6.1 What is IBM Campaign	74
6.2 Components and architecture	75
6.2.1 IBM Campaign and DB2 for z/OS	76
6.2.2 IBM Campaign performance considerations and usage of temp tables	76
6.2.3 Defining temporary tables	78
6.3 Our IBM Campaign environment	79
6.4 Campaign example scenario used in this chapter	82
6.4.1 Flowchart for car insurance campaign	82
6.4.2 Using temp tables for car insurance campaign	83
6.4.3 Using and enabling accelerator-only tables for car insurance campaign	84
Chapter 7. In-database transformations	87
7.1 In-database transformations	88
7.2 Custom transformation and extract, transform, and load processes	91
7.2.1 Consolidation and optimization	93
7.2.2 Performance	95
7.3 Accelerator and accelerator-only table usage in IBM InfoSphere DataStage	96
7.3.1 IBM InfoSphere Information Server	96
7.3.2 Things to consider for configuration	101
7.3.3 Basic accelerator usage within Data Stage	106
7.3.4 Accelerator maintenance through DataStage	113
7.3.5 Optimizing existing DataStage jobs	130
7.3.6 Pitfalls	144
7.3.7 Loading large amounts of data through DataStage to the Accelerator	147

Chapter 8. Accelerator-only tables supporting data scientists’ ad hoc analysis	151
8.1 Data science and ad hoc analysis	152
8.2 Interactive analysis with notebooks: Python and Jupyter	154
8.3 Example with insurance claim data in DB2 for z/OS	154
8.3.1 Sample data layout in DB2 for z/OS	156
8.3.2 Accessing DB2 for z/OS data in Python	157
8.3.3 Data analysis examples with accelerator-only tables	159
8.4 More data scientist aspects	164
Chapter 9. Integrating more data sources and archiving for analytics	165
9.1 DB2 Analytics Accelerator Loader for z/OS	166
9.2 General function overview	166
9.2.1 Group Consistent Load	166
9.2.2 Dual load	167
9.2.3 Image Copy load	168
9.3 Support for accelerator-only tables	169
9.3.1 Loading accelerator-only tables using Accelerator Loader	170
9.4 Integrating other DBMS data with DB2 for z/OS data	170
9.5 Architecture pattern: Incremental archiving with accelerator-only tables	172
Chapter 10. In-database analytics	175
10.1 Reasons to use in-database analytics with IBM DB2 Analytics Accelerator	176
10.2 Executing analytical functions using an Accelerator	177
10.3 Enable IBM SPSS modeling for acceleration	178
10.4 Technical implementation	180
Appendix A. Description of IBM z Systems environment used for this publication	183
A.1 IBM DB2 Analytics Accelerator	184
A.2 IBM DB2 for z/OS	185
A.3 IBM System z system environment	192
A.4 IBM InfoSphere Information Server DataStage	193
Related publications	195
IBM Redbooks	195
Other publications	195
Online resources	195
Help from IBM	196

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 about 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 grant 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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites 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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

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


COPYRIGHT LICENSE:

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

Trademarks

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

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

AIX®	IBM PureData®	Redbooks®
BigInsights®	IBM z™	Redbooks (logo)  ®
CICS®	IBM z Systems™	SPSS®
Cognos®	IBM z13™	System z®
CPLEX®	IMS™	Tivoli®
DataStage®	Informix®	Unica®
DB2®	InfoSphere®	WebSphere®
DB2 Connect™	MVS™	z Systems™
DB2 Universal Database™	OS/390®	z/OS®
DRDA®	PureData®	z/VM®
ECKD™	pureXML®	z/VSE®
FICON®	QMFT™	z13™
FlashCopy®	QualityStage®	zEnterprise®
i5/OS™	Query Management Facility™	
IBM®	RACF®	

The following terms are trademarks of other companies:

Netezza, and N logo are trademarks or registered trademarks of IBM International Group B.V., an IBM Company.

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

Microsoft, Windows, 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 Oracle and/or its affiliates.

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

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

Find and read thousands of IBM Redbooks publications

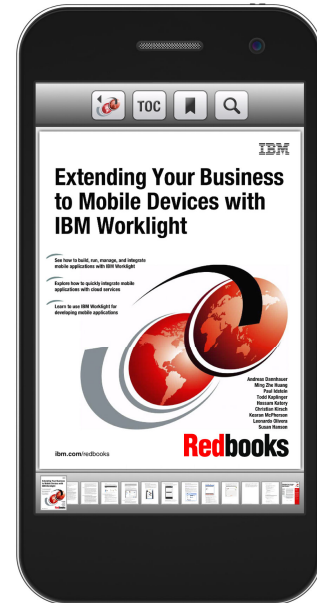
- ▶ Search, bookmark, save and organize favorites
- ▶ Get up-to-the-minute Redbooks news and announcements
- ▶ Link to the latest Redbooks blogs and videos

Get the latest version of the Redbooks Mobile App



Download
Now

iOS



Promote your business in an IBM Redbooks publication

Place a Sponsorship Promotion in an IBM® Redbooks® publication, featuring your business or solution with a link to your web site.

Qualified IBM Business Partners may place a full page promotion in the most popular Redbooks publications. Imagine the power of being seen by users who download millions of Redbooks publications each year!



ibm.com/Redbooks

About Redbooks → Business Partner Programs

THIS PAGE INTENTIONALLY LEFT BLANK

Preface

Transforming data from operational data models to purpose-oriented data structures has been commonplace for the last decades. Data transformations are heavily used in all types of industries to provide information to various users at different levels. Depending on individual needs, the transformed data is stored in various different systems.

Sending operational data to other systems for further processing is then required, and introduces much complexity to an existing information technology (IT) infrastructure. Although maintenance of additional hardware and software is one component, potential inconsistencies and individually managed refresh cycles are others.

For decades, there was no simple and efficient way to perform data transformations on the source system of operational data. With IBM® DB2® Analytics Accelerator, DB2 for z/OS is now in a unique position to complete these transformations in an efficient and well-performing way. DB2 for z/OS completes these while connecting to the same platform as for operational transactions, helping you to minimize your efforts to manage existing IT infrastructure.

Real-time analytics on incoming operational transactions is another demand. Creating a comprehensive scoring model to detect specific patterns inside your data can easily require multiple iterations and multiple hours to complete. By enabling a first set of analytical functionality in DB2 Analytics Accelerator, those dedicated mining algorithms can now be run on an accelerator to efficiently perform these modeling tasks.

Given the speed of query processing on an accelerator, these modeling tasks can now be performed much quicker compared to traditional relational database management systems. This speed enables you to keep your scoring algorithms more up-to-date, and ultimately adapt more quickly to constantly changing customer behaviors.

This IBM Redbooks® publication describes the new table type that is introduced with DB2 Analytics Accelerator V4.1 PTF5 that enables more efficient data transformations. These tables are called *accelerator-only* tables, and can exist on an accelerator only.

The tables benefit from the accelerator performance characteristics, while maintaining access through existing DB2 for z/OS application programming interfaces (APIs). Additionally, we describe the newly introduced analytical capabilities with DB2 Analytics Accelerator V5.1, putting you in the position to efficiently perform data modeling for online analytical requirements in your DB2 for z/OS environment.

This book is intended for technical decision-makers who want to get a broad understanding about the analytical capabilities and accelerator-only tables of DB2 Analytics Accelerator. In addition, you learn about how these capabilities can be used to accelerate in-database transformations and in-database analytics in various environments and scenarios, including the following scenarios:

- ▶ Multi-step processing and reporting in IBM DB2 Query Management Facility™, IBM Campaign, or Microstrategy environments
- ▶ In-database transformations using IBM InfoSphere® DataStage®
- ▶ Ad hoc data analysis for data scientists
- ▶ In-database analytics using IBM SPSS® Modeler

Authors

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

Ute Baumbach has been a Software Engineer at the IBM Research and Development Laboratory in Boeblingen, Germany for 25 years, where she works in various software development projects and roles. Most of her projects were based on DB2 for Linux, UNIX, and Windows, and now also on DB2 for z/OS. For two and a half years, she has worked as a member of the Analytics on IBM z Systems™ Center of Excellence team, focusing on DB2 Analytics Accelerator proofs of concepts, customer deployment support, and education.

Patric Becker is a Software Architect in Analytics on z Systems Center of Excellence at the IBM Boeblingen Lab. The team has specialized in DB2 Analytics Accelerator, and performs IBM-internal and external education for this product, supports customer installations, and provides proofs-of-concepts. Patric has more than 18 years of experience with DB2 for z/OS. He has co-authored five IBM Redbooks publications, focusing on DB2 for z/OS, high availability (HA), performance, and data warehousing.

Uwe Denneler is an IT Specialist at the IBM Client Center in the IBM Boeblingen Lab, Germany. He has more than 20 years of experience in the mainframe and IBM z/OS® field. He worked with independent software vendor (ISV) and Customer projects on z Systems (z/OS, IBM z/VM®, IBM z/VSE®, Linux on z Systems, and various subsystems). He also prepares demonstrations on IBM z Systems.

Eberhard Hechler is an Executive Architect from the IBM Germany Research and Development Lab. He is a member of DB2 Analytics Accelerator development. In his 2.5 years at the IBM Kingston Lab in New York, Eberhard has worked in software development, performance optimization, IT solution architecture and design, Hadoop and Spark integration, and Master Data Management. He then worked with DB2 for IBM MVST™, focusing on testing and performance measurements. He has worked worldwide with IBM clients from various industries on a vast number of topics, such as data warehouse and business intelligence (BI) solutions, information architectures, and industry solutions. From 2011-2014, he was at IBM Singapore, working as the Lead Big Data Architect in the Communications Sector of IBM Software Group (SWG). He is a member of the IBM Academy of Technology and co-authored the following books:

- ▶ *Enterprise MDM*, Pearson, 2008, ISBN: 0132366258
- ▶ *The Art of Enterprise Information Architecture*, Pearson, 2010, ISBN: 0137035713
- ▶ *Beyond Big Data*, Pearson, 2014, ISBN: 013350980X

Wolfgang Hengstler is the Product Manager for the DB2 Analytics Accelerator. During his 34 years with IBM, Wolfgang has held positions in development, architecture, product, and market management in different projects in IBM Software Group, IBM Storage Group, IBM Tivoli®, and IBM Global Services. Among them were developing operating system components, system automation products, and application hosting services. Wolfgang works in the IBM Research and Development Lab in Boeblingen, Germany.

Steffen Knoll is a Software Engineer in the IBM Research and Development Lab in Boeblingen. He has 10 years of experience in software development and Data Warehousing on z Systems. He holds a degree in Applied Computer Science. His areas of expertise include developing in-memory databases, such as IBM Smart Analytics Accelerator, and integrating Netezza® into IBM DB2 Analytics Accelerator.

Frank Neumann is a Senior Software Engineer and technical team lead of the Analytics on z Systems Center of Excellence, in the IBM Research and Development Lab in Boeblingen, Germany. As part of the development organization, his responsibilities include customer consultation, proofs of concepts, deployment support, and education for DB2 Analytics Accelerator. During his 17 years with IBM, Frank worked as a developer, team leader, and architect for software components and solutions for IBM WebSphere® and Information Management products.

Guenter Georg Schoellmann is an Advisory Project Manager and IT specialist at the Analytics on z Systems Center of Excellence at the IBM Research and Development Lab in Boeblingen, Germany. As part of the development organization, his responsibilities include customer consultation, proofs of concepts, deployment support, and education for DB2 Analytics Accelerator. During his more than 30 years with IBM, Guenter has worked as a developer, team leader, customer advocate, and project manager in DB2 for z/OS utilities and other information management products.

Khadija Souissi is a certified IT Specialist and Consultant at the IBM Client Center - Systems and Software, IBM Germany Lab in Boeblingen. She holds a Master's degree in Electrical Engineering. During her 14 years with IBM, Khadija has worked as an IBM System z® Firmware developer, Field Technical Sales Specialist, and Business Analytics subject matter expert (SME). She gained experience in System z, IBM Power, storage, data warehousing, information governance, and Business Analytics Solutions on System z with deep hands-on skills in IBM Cognos® BI, SPSS, and DB2 Analytics Accelerator for z/OS. Her responsibilities include customer consultation, proofs of concepts, enablement workshops, demonstration development, and delivery worldwide. With various language skills, Khadija can provide special support during briefings, events, and customer engagements, especially for French-speaking and Arabic-speaking customers from Europe, Middle East, and Africa (EMEA), by working in their native language.

Timm Zimmermann is a Technical Advisor with Rocket Software. In his previous role with IBM, Timm was a Software Developer and a member of the DB2 for z/OS Development SWAT team. In this role, he consults for worldwide customers on various technical topics, including implementation and migration, design for high performance and high availability, performance and tuning, system health checks, and disaster recovery (DR). Before this role, Timm was a DB2 Engine developer, with extensive knowledge and experience with DB2 Real-Time Statistics and DB2 Utilities.

Acknowledgement

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

Toni Bollinger
Karl-Heinz (Charly) Fesenbeckh
IBM Germany

Marie-Laure Pessoa Caucat
IBM France

Gary W. Crupi
Andy Perkins
Dino Tonelli
Robin Zalud
IBM USA

Whei-Jen Chen, ITSO project leader
International Technical Support Organization, Boeblingen Center

Now you can become a published author, too

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time. Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Learn more about the residency program, browse the residency index, and apply online:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us.

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

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

ibm.com/redbooks

- Send your comments in an email:

redbooks@us.ibm.com

- Mail your comments:

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

Stay connected to IBM Redbooks

- Find us on Facebook:

<http://www.facebook.com/IBMRedbooks>

- Follow us on Twitter:

<http://twitter.com/ibmredbooks>

- Look for us on LinkedIn:

<http://www.linkedin.com/groups?home=&gid=2130806>

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>

- Stay current on recent Redbooks publications with RSS Feeds:

<http://www.redbooks.ibm.com/rss.html>



Analytics on an IBM z Systems environment

This chapter describes the context of why IBM DB2 Analytics Accelerator-only tables play such a pivotal role in Enterprise Data Warehouse (EDW) and analytics on IBM z Systems scenarios. An accelerator-only table (AOT) is an object (a place to store data) that can only exist on the DB2 Analytics Accelerator. Access to AOTs is strictly through IBM DB2 for z/OS. However, except for the metadata of an AOT, no data is stored in DB2 for z/OS at any time.

Within this introductory chapter, we briefly elaborate the following areas:

- ▶ Traditional data warehouse and analytics concepts
- ▶ IBM Transaction and Analytics Processing
- ▶ Transformation patterns
- ▶ New use cases
- ▶ Strategic outlook

Acknowledging this broader context leads us to recall traditional data warehousing (DWH) and analytics on z Systems concepts, specifically regarding the naturally accompanying complex data persistency layer and cumbersome information supply chain. In the past, data from heterogeneous source systems had to be transformed and persisted in various data containers, such as a staging area, a system of record, data marts, and even various data cubes.

Advances in IBM z Systems hardware, online transaction processing (OLTP), and online analytical processing (OLAP) capabilities have made it possible to improve the DWH and analytics landscape by reducing the number of required data persistency containers and simplifying the data flow. There have also been advances in new functions in products, such as DB2 for z/OS with the DB2 Analytics Accelerator, IBM SPSS Modeler, and IBM Operational Decision Manager (ODM), just to name a few.

IBM Transaction and Analytics Processing is a term to describe the capability of a single database (DB) that can accommodate both OLTP and OLAP paradigms for real-time operational analytics and transactional workload management on a single system.

Real-time analytics is an enabling capability for many use case scenarios across various industries, which is closely linked to the IBM Transaction and Analytics Processing concept. DB2 Analytics Accelerator-only tables represent a major step toward IBM Transaction and Analytics Processing, real-time analytics, and Operational Data Stores (ODS).

Data transformation with its associated characteristics, such as data quality improvements, data governance, metadata management, and data provisioning and loading, constitutes a vital component in any DWH environment and analytics scenario. In this introductory chapter, we highlight the essential role of DB2 Analytics Accelerator-only tables, enabling simplified transformation patterns that go beyond the traditional extract, transform, and load (ETL) pattern.

Because today's complexity of data transformation still represents the key reason for DWH and business intelligence (BI) project failures, DB2 Analytics Accelerator-only tables contribute significantly to simplifying DWH and analytics on z Systems solution architectures, and mitigating project execution risks.

In the past, the value of the DB2 Analytics Accelerator was well-founded by its significant performance improvement of existing Structured Query Language (SQL) workloads. With the introduction of the AOTs, several additional use case scenarios are possible. This introductory chapter highlights these use cases, which are described in much greater detail in subsequent chapters.

Accelerator-only tables portray one particular aspect of analytics on z Systems. This introductory chapter concludes with a strategic outlook in further improving the DWH and analytics on z Systems value proposition, accomplishing IBM Transaction and Analytics Processing in its full scale over time, and eventually enabling even more use case scenarios.

1.1 Traditional data warehouse and analytics concepts

Traditional DWH and analytics environments were characterized by data latency issues. The often-cumbersome data flow through various data persistency layers, such as the staging area, the system of record, several data marts, and even several data cubes, prevented low latency analytics and timely availability of reports. In the past, analytics on z Systems was frequently characterized by the analytics layer itself implemented on distributed platforms.

1.1.1 Persistency layers

Complementing the OLTP systems, most customers today still operate a traditional DWH and analytics systems landscape with data persistency containers that serve different purposes:

Staging area	To maintain a work area to cleanse and transform source data.
ODS	To enable operational analytics.
System of record (SOR)	To serve as a consistent base for generating data marts.
Data marts	To address Line of Business (LOB)-specific analytical needs.
Data cubes	To enable analytical scenarios, such as business planning or enabling simple drill-down and drill-through capabilities.

Often, customers even maintained several of these data persistency containers for specific business purposes, for example, maintaining a LOB-specific ODS or SOR, without any Enterprise-wide implementation of a single SOR. This systems landscape does result in several negative ramifications:

- ▶ High complexity in both system management and application development
- ▶ Difficulties in supporting real-time analytics
- ▶ Inability to match the ever more demanding service level agreement (SLA) requirements
- ▶ High total cost of ownership

As you see throughout this book, AOTs are a vehicle to address some of these issues.

1.1.2 Information supply chain

Traditional DWH environments were characterized by rather cumbersome information supply chains, where data from heterogeneous, mainly OLTP source systems and other application systems had to be accessed, cleansed, transformed, and loaded into the DWH system. There are several historical reasons for that, such as different access patterns and the negative effect on performance if the BI and analytical workload would have been run on the transactional systems.

Furthermore, the performance characteristics of OLTP systems with the corresponding data structures were not adequate to accommodate BI and analytical workloads. Consequently, source data had to be transformed and aggregated into data structures (that is, star schemas, OLAP cubes) that were suitable for running these workloads. Lastly, different lifecycle characteristics required the implementation of a data integration hub, which the EDW served the purpose of. The existing information supply chain is characterized by the following issues:

- ▶ Coexistence of numerous languages, such as Cobol, SQL, and information integration tools, such as IBM InfoSphere DataStage and IBM InfoSphere QualityStage®
- ▶ Maintenance issues and a high degree of complexity of the entire data flow
- ▶ Latency aspects related to available and actionable insight
- ▶ Missing metadata management capability, including data lineage and data provenance

Accelerator-only tables provide in-database transformation, and enable several data transformation patterns that greatly simplify the overall information supply chain, and can address some of the previous issues.

1.1.3 Analytics on z Systems aspects

IBM z Systems and DB2 for z/OS have traditionally been associated with OLTP and transactional workloads. However, with the introduction of DB2 Analytics Accelerator, customers are increasingly running analytical workloads on z Systems, and are deploying DWH systems using DB2 for z/OS with the DB2 Analytics Accelerator. As we see later in this chapter, there are other technical capabilities that have greatly increased the overall analytics on z Systems value proposition.

1.2 IBM Transaction and Analytics Processing

Performance limitations of existing OLTP systems were the primary reason to build separate DWH systems. As a consequence, multiple copies of the same data had to be maintained, using different data models to address the performance requirements of OLAP and analytics workloads. Recent advances in IBM z Systems hardware, enhanced OLTP and OLAP capabilities, and innovative new product functions enable transactional processing and analytics to operate on the same database.

IBM Transaction and Analytics Processing is a concept that eliminates the need for multiple copies of the same data. It provides the foundation for real-time and operational analytics, reduces the number of data marts or even eliminates the need for data marts entirely, and enables the logical data warehouse. DB2 for z/OS with the DB2 Analytics Accelerator is a key enabler for this concept.

Figure 1-1 is a high-level depiction of the IBM Transaction and Analytics Processing concept.

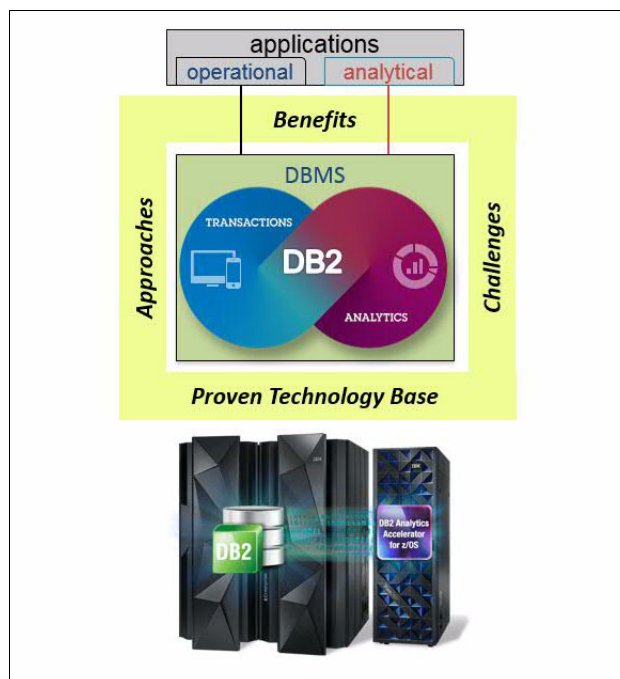


Figure 1-1 IBM Transaction and Analytics Processing

1.2.1 Logical data warehouse

DB2 Analytics Accelerator and specifically the AOTs constitute a key step toward a uniform access to data for any type of application. DB2 for z/OS serves as an *entry platform* to run transactional and analytical workloads, and data transformation jobs. With the in-DB transformation capability of the Accelerator-only tables, latency between data creation and data consumption can be significantly reduced or even eliminated.

IBM Transaction and Analytics Processing represents an opportunity to reduce redundancy of data by removing and consolidating some of the traditional DWH layers. For instance, the number of data marts might be significantly reduced with fewer aggregations to be built within the remaining data marts. Depending on the DWH and analytical requirements, the data mart layer might even be completely removed.

The transactional and analytical capabilities become integrated with DB2 for z/OS and the DB2 Analytics Accelerator as the single IBM Transaction and Analytics Processing database management system (DBMS) engine. Therefore, the DWH can be seen as a logical data warehouse, where efficient data movement is performed within the system, not between systems, often not even involving the network.

Last, compared to the traditional DWH and analytics landscape, IBM Transaction and Analytics Processing provides uniform policies and procedures for security, high availability (HA), disaster recovery (DR), and systems monitoring, using the same tools and skills.

1.2.2 Technical approaches

Delivering against the IBM Transaction and Analytics Processing promise, there are a series of technical challenges that must be addressed. First, there need to be mixed workload management capabilities to prioritize workload execution against defined SLAs.

Furthermore, continuous availability, security, and reliability have to be ensured. In order to accommodate data and workload growth, seamless scale-up and scale-out capabilities need to be provided. Last, IBM Transaction and Analytics Processing systems need to provide universal processing capabilities to deliver best performance for both transactional and analytical workloads, without the need for excessive tuning.

The following sections provide information about some of the technical approaches that can be taken to build IBM Transaction and Analytics Processing systems. DB2 for z/OS, with its vast scope of technical functions and capabilities, the DB2 Analytics Accelerator, and specifically the new accelerator-only tables in concert with the superior technology of other IBM z Systems products, address most of these areas already:

- ▶ Large random access memory (RAM) size
- ▶ In-memory database techniques
- ▶ Massively parallel processing
- ▶ Large number of sockets, processor cores, and servers
- ▶ Vector processing
- ▶ Hardware acceleration through special-purpose processors
- ▶ Field Programmable Gate Arrays (FPGA), graphics processing units (GPU), and other specialized processing units
- ▶ Columnar data stores
- ▶ Special-purpose appliances

1.2.3 Real-time analytics on z Systems

With the introduction of the DB2 Analytics Accelerator, z Systems, and specifically DB2 for z/OS, clients can broaden the processing scope to include advanced analytics, even with low latency to real-time characteristics. With the advanced analytics capabilities of the new IBM z13™, and the availability of the IBM SPSS Modeler with Scoring Adapter for IBM zEnterprise® V15.0, clients can implement several use cases that require real-time scoring of transactional data in DB2 for z/OS.

DB2 for z/OS with the DB2 Analytics Accelerator specifically with its accelerator-only tables contribute to several real-time analytics on z Systems solutions that can be complemented with the following additional offerings:

- ▶ Advanced analytics with IBM z13
- ▶ Business rule management system (BRMS) for z/OS clients
- ▶ SPSS Modeler with Scoring Adapter for zEnterprise
- ▶ Availability of the IBM CPLEX® standard for z/OS

1.2.4 Analytics on operational data

The concept of ODS has existed for several years. However, clients are still struggling to implement operational analytics. An ODS typically involves data cleansing, some level of information integration, and resolution of data redundancy. It also includes business rules validation for business integrity purposes. An ODS typically only contains a small time window of data, and relates to a subset of operational systems that are reflected in building and maintaining an ODS, such as for customer care, retail banking, or marketing purposes.

The following list describes some application examples:

- ▶ Status of a customer order
- ▶ Credit score calculation
- ▶ Real-time fraud validation
- ▶ Real-time validation for account opening processes

With accelerator-only tables, an ODS can easily be built within the DB2 for z/OS OLTP transactional landscape. DB2 for z/OS and the DB2 Analytics Accelerator enable even more complex ODS systems to be built that involve low latency update requirements and some more complex transformation and aggregation.

1.3 Transformation patterns

The new accelerator-only tables enable several new transformation patterns that complement the traditional data movement-centric ETL.

1.3.1 Traditional ETL

The traditional ETL procedure requires a dedicated information integration server, where source data must be *extracted* from a heterogeneous source systems landscape, *transformed* on the information integration server, and then eventually *loaded* into the target data systems. Depending on the source systems layer, this traditional ETL procedure implies movement of potentially huge data volumes from source to target systems.

This prevents an agile data flow that requires low latency or real-time characteristics, and is contrary to the often-cited imperative to process data where it resides. In combination with the large number of different data persistency layers in the conventional DWH and analytics on a z Systems landscape, traditional ETL therefore circumvents the implementation of IBM Transaction and Analytics Processing.

1.3.2 Accelerating data transformation

With the Accelerator-only tables and in-DB transformation capabilities, data transformation can be simplified and accelerated. Data does not necessarily have to be moved to another information integration server for transformation and cleansing purposes. The data can now be transformed within the DB2 Analytics Accelerator. With InfoSphere DataStage Balanced Optimization¹, IBM does provide in-DB transformation on other platforms, such as DB2 for z/OS, DB2 for Linux, UNIX, and Windows, IBM PureData® for Analytics (PDA), Teradata, Oracle, and Hadoop.

Optimization pushes processing functionality and related data input/output (I/O) into database sources or targets, or Hadoop clusters, depending on the optimization options that you choose. Using Hadoop clusters, InfoSphere DataStage Balanced Optimization writes Big Data File Stage (BDFS) jobs into JavaScript Object Notation (JSON) query language (Jaql). BDFS operates on IBM InfoSphere BigInsights®. BDFS was introduced in InfoSphere Information Server V9.1.

1.3.3 New transformation patterns

With DB2 Analytics Accelerator V4.1 PTF 5, InfoSphere DataStage Balanced Optimization² supports AOTs and therefore, works on the DB2 Analytics Accelerator. For an in-depth discussion about DataStage usage of AOTs, see *Reliability and Performance with IBM DB2 Analytics Accelerator V4.1*, SG24-8213.

The following list illustrates some of the additional transformation patterns:

- ▶ **ELT.** The source data is *extracted* from several source systems, *loaded* into the DB2 Analytics Accelerator, and *transformed* there to be used by analytics applications directly accessing the transformed data through DB2 for z/OS. Accelerator-only tables can be used to store transformed data *only* in the DB2 Analytics Accelerator. This enables ODS deployments on z Systems, and constitutes a vital step toward implementation of IBM Transaction and Analytics Processing paradigms.
- ▶ **ELTEL.** In addition to the ELT steps in the previous pattern, the already transformed data is *extracted* from the DB2 Analytics Accelerator using DB2 for z/OS, and *loaded* into another target system for consumption by analytics applications. This presents a transformation pattern that enables the existing DWH and analytics landscape to remain unchanged, and yet simplifies and streamlines the entire information supply chain.

With the stunning performance characteristics of the DB2 Analytics Accelerator, the complexity of the transformation jobs can be significantly reduced, and in some cases, even eliminated. Depending on the specific set of requirements for a particular DWH and analytics project, additional customized transformation patterns can be designed and deployed.

¹ For more information about InfoSphere DataStage Balanced Optimization, see the following websites:
https://www.ibm.com/support/knowledgecenter/#!/SSZJPZ_11.3.0/com.ibm.swg.im.iis.ds.parjob.dev.doc/toc/pics/introductiontobalancedoptimization.html and
<http://www.ibm.com/developerworks/data/library/techarticle/dm-1402optimizebdfs>

² Balanced Optimization is a licensed add-on to InfoSphere DataStage, which is a product of the IBM InfoSphere Information Server platform.

1.4 New use cases

The DB2 Analytics Accelerator already enables several use case scenarios that complement the traditional OLTP transactional environment with analytics on z Systems capabilities. With every new DB2 Analytics Accelerator version or program temporary fix (PTF), this set of use cases is improved, and new ones are enabled.

1.4.1 Use case description

Table 1-1 provides an overview of some of the most prominent use case scenarios. The last column in the table indicates those use cases that benefit or are enabled by the accelerator-only tables and in-DB transformation capabilities, and are therefore related to the data flow optimization and simplifications.

As can be seen by this table, there is quite a persuasive number of use cases, which can even be deployed in combination. In concrete client situations, several of these use cases are related to an initiative and are intertwined.

Table 1-1 Use case scenarios value proposition

#	Use case	Key characteristics	Data flow relevance
1	Radical performance improvements	<ul style="list-style-type: none">▶ Typically orders of magnitude (up to ~2000)▶ No need for tuning (leveraging PDA technology, powered by Netezza)▶ Significant reduction of processor use cost▶ No need to restrict users from running data-intensive queries when they want	NO
2	Enterprise Data Warehouse	<ul style="list-style-type: none">▶ Taking advantage of homogeneous environment▶ HA, DR, security, monitoring, tooling, skills, and so on▶ Optimize/simplify ETL, turn into ELT using DataStage Balanced Optimization	YES
3	Making data marts obsolete	<ul style="list-style-type: none">▶ No need for multiple copies of data▶ No need for ETL to create data marts▶ Centralized administration	YES
4	Operational reporting	<ul style="list-style-type: none">▶ Analytics on transactional data▶ Real-time availability of transformed/cleansed data▶ Enabling data reservoirs	NO
5	Reduced storage cost	<ul style="list-style-type: none">▶ Archiving without query performance penalty▶ Using High Performance Storage System (HPSS) for DB2 for z/OS historical data	NO
6	SQL on IBM IMS™, VSAM, and sequential data	<ul style="list-style-type: none">▶ Using IBM DB2 Analytics Accelerator Loader with DataStage to load sequential/VSAM data into IBM DB2 Analytics Accelerator^a▶ No sequential/Virtual Storage Access Method (VSAM) data on DB2 for z/OS (just schema)	YES
7	z13 embedded analytics	<ul style="list-style-type: none">▶ Using in-IBM DB2 Analytics Accelerator predictive analytics (SPSS integration) in IBM DB2 Analytics Accelerator V5▶ With real-time scoring adapter in DB2 for z/OS	NO

#	Use case	Key characteristics	Data flow relevance
8	Data flow simplification	<ul style="list-style-type: none"> ▶ Reduce the complexity of the Enterprise-wide information supply chain ▶ With DB2 Analytics Accelerator and with the in-DB transformation in the IBM DB2 Analytics Accelerator V4.1 PTF 5 ▶ Using DataStage Balanced Optimization on DB2 Analytics Accelerator, DB2 for z/OS, Oracle, and so on ▶ Supporting several patterns: ELT, TELT, or TETLT 	YES
9	SMF Logs analytics	<ul style="list-style-type: none"> ▶ IBM DB2 Analytics Accelerator can be used in the context of IBM Tivoli Decision Support for z/OS (TDSz) and CMAz analytics ▶ Using IBM DB2 Analytics Accelerator Loader for loading IBM System Management Facility (SMF) Log records directly into DB2 Analytics Accelerator without ETL ▶ No SMF Log records on DB2 for z/OS (just schema) 	NO
10	Data scientist work area	<ul style="list-style-type: none"> ▶ Deeper insight into customers and markets ▶ Data Scientists are creating temporary DB objects for ad hoc analysis ▶ Access control through personal DB in DB2 for z/OS ▶ Accelerator-only tables to process and store filtered and transformed results of source transactional data 	YES
11	Improved performance for iterative campaign tuning	<ul style="list-style-type: none"> ▶ Accelerated campaign tuning for IBM Campaign (formerly Unica®) ▶ Multi-step campaign management ▶ Using accelerator-only tables for storing large intermediate results in the DB2 Analytics Accelerator without data movement 	YES
12	Improved performance for iterative reporting	<ul style="list-style-type: none"> ▶ Deeper insight into operational status through faster reporting ▶ MicroStrategy, IBM QMF™, home grown applications, and so on ▶ Multi-step reporting applications, such as QMF, Microstrategy, and Statistical Analysis System (SAS) ▶ Using temporary objects to store results, that is, temporary tables for intermediate query results ▶ Accelerator-only tables provide an alternative to Declared Global Temporary Tables (DGTs) or regular DB2 tables, particularly for large intermediate results 	YES

a. For an in-depth discussion about the IBM DB2 Analytics Loader capabilities, see Chapter 9, *Reliability and Performance with IBM DB2 Analytics Accelerator V4.1*, SG24-8213.

Depending on the deployment of individual or combined use cases, the value proposition ranges from reduction of the total cost of operations (TCO) in operating the entire DWH and analytics infrastructure, which includes the entire information supply chain. Some clients have achieved great value through some of these use cases by simply reducing the movement of data, or even avoiding data transformation at all, therefore keeping z data on z Systems and being able to run analytical workloads on the z Systems platform.

The optimization and simplification of the information supply chain results in low latency data transformation aspects. Reducing data latency in generating analytical insight and making this insight available to consuming systems, even in real-time, represents a key business value.

For instance, contextual marketing campaign management initiatives can be operated with lowest latency, increasing customer retention and loyalty. Credit card fraud detection and avoidance is enabled through real-time analytics that is underpinned by the latest capabilities of the DB2 Analytics Accelerator. Instantaneous, on-demand customer insight and service can be delivered, increasing the overall customer experience in service delivery. Operational analytics can be increasingly performed on *z data*, limiting or even avoiding data movement.

Through its latest capabilities, the DB2 Analytics Accelerator enables clients to embark on the strategic IBM Transaction and Analytics Processing journey, where the value proposition reaches far beyond the traditional IT value areas, such as TCO reduction, storage savings, and so on. It now enables transformational business opportunities for all *z Systems* clients.

1.5 Strategic outlook

Before we dive into the details of Accelerator-only tables in the subsequent chapters, we provide a short strategic outlook. Further improving operational and real-time analytics and enhancing IBM Transaction and Analytics Processing capabilities are certainly recognized as key strategic imperatives. This includes support for additional business-relevant use case scenarios by further optimizing in-DB transformation capabilities and complementing this with in-DB and advanced analytics.

Improving existing features, such as enabling more query acceleration, increasing the DB2 Analytics Accelerator transparency, delivering deeper *z Systems* integration, and using the PDA technology evolution are additional strategic investment areas. Figure 1-2 is a high-level depiction of the strategic direction of the DB2 Analytics Accelerator.

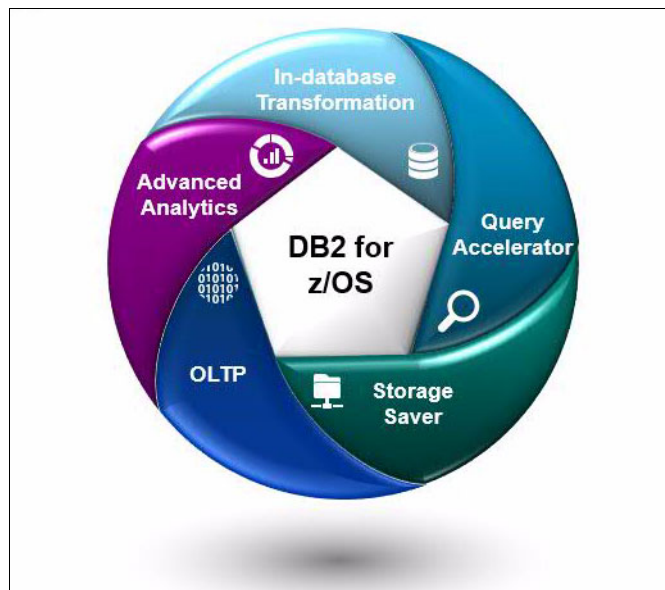


Figure 1-2 Strategic outlook

Delivering true IBM Transaction and Analytics Processing, ODS, and real-time analytics on *z Systems* capabilities, ultimately requires consolidation and unification of transactional and analytical data stores. This eliminates data movement and enables true application transparency.

The subsequent chapters provide further details of Accelerator-only tables and in-database transformation, including corresponding use case scenarios.



Accelerator-only tables

This chapter explains the technical details of accelerator-only tables (AOTs) and describes the following topics:

- ▶ Concepts and architecture
- ▶ Software level prerequisites
- ▶ Syntax and capabilities
- ▶ Transactional considerations
- ▶ Lifecycle management
- ▶ Limitations and restrictions
- ▶ Performance considerations

2.1 Concepts and architecture

Accelerator-only tables (AOTs) are tables that do not originate from DB2 base tables. Their data exists on the IBM DB2 Analytics Accelerator (Accelerator) only. Queries and Data Manipulation Language (DML) statements against accelerator-only tables are always routed to the Accelerator. The special register must be set to **ENABLE**, **ELIGIBLE**, or **ALL**. If a Structured Query Language (SQL) statement does not satisfy the conditions for acceleration, the statement fails.

Data in accelerator-only tables can only be modified by **INSERT**, **UPDATE**, and **DELETE** (I/U/D) statements. How data is modified by DML statements is explained in 2.3, “Syntax and capabilities” on page 18.

The main purpose of accelerator-only tables is to store intermediate or final results of data transformation or reporting processes. At the same time, AOTs accelerate queries on these results, either on intermediate results during the processes or on final results after the processes complete. The benefits are described in the following list:

- ▶ Accelerate in-database data transformation and data movement processes
- ▶ Reduced need of data movement processes to other platforms for data transformation purposes
- ▶ Enables multistep reporting on the Accelerator
- ▶ Saves disk space and processor (CPU) use cost on IBM z Systems currently used for transformations and reporting steps
- ▶ Enable data preparation steps for data mining and other advanced analytics to run on the Accelerator

The benefits and use cases for accelerator-only tables are further discussed in detail in Chapter 3, “Use cases that are enabled by accelerator-only tables and in-database analytics” on page 31.

An accelerator-only table is a new table type among others that you work with in a DB2 Analytics Accelerator environment. It is important to understand the characteristics of each of them to use them to your best advantage in your DB2 Analytics Accelerator environment.

Figure 2-1 on page 13 shows the characteristics of all table types. The details are explained in the following sections.

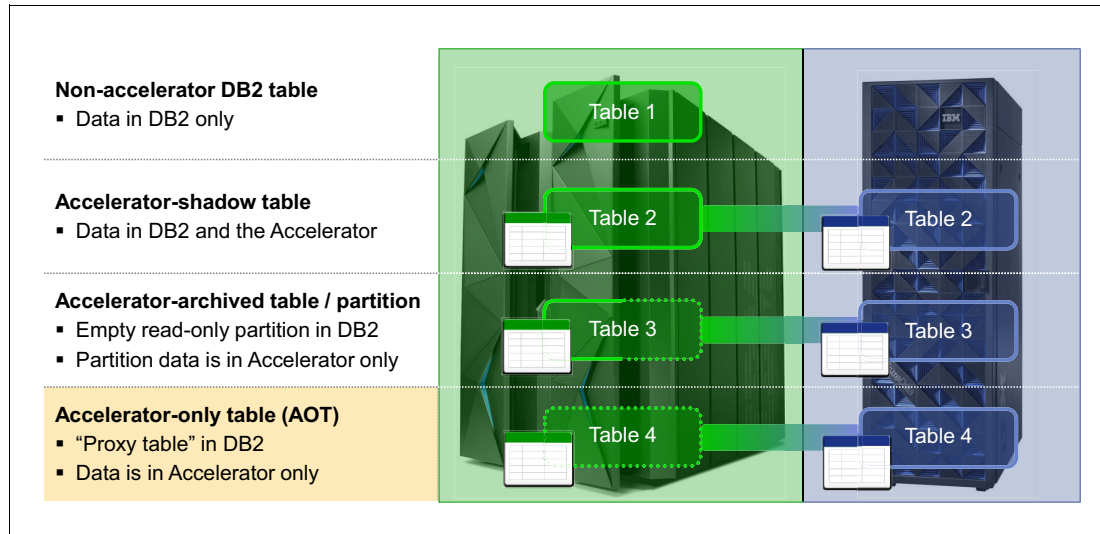


Figure 2-1 Accelerator table types

2.1.1 Non-accelerator DB2 table

Figure 2-2 shows the architecture of a non-accelerator DB2 table.

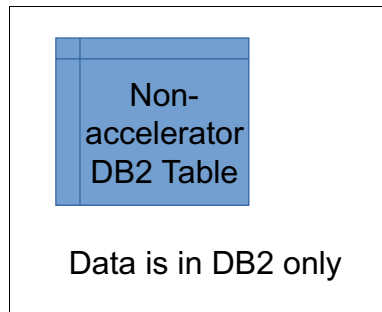


Figure 2-2 Non-accelerator DB2 table

A non-accelerator DB2 table and its data exists in DB2 only. Queries run against that data are always run in DB2. A short summary with explanations of all different DB2 table types can be found in the following IBM Knowledge Center:

http://www.ibm.com/support/knowledgecenter/SSEPEK_11.0.0/com.ibm.db2z11.doc.intro/src/tpc/db2z_typesoftables.dita

2.1.2 Accelerator-shadow table

Figure 2-3 shows the architecture of an accelerator-shadow table.

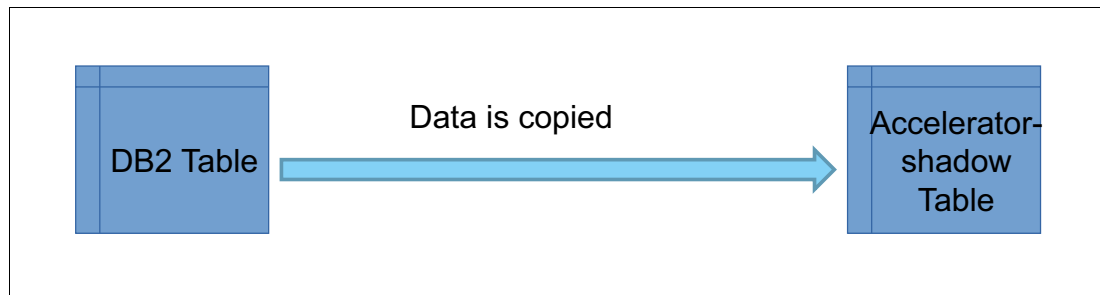


Figure 2-3 Accelerator-shadow table

An accelerator-shadow table is a table on the Accelerator that is a copy of a DB2 table. The accelerator-shadow table contains all or a subset of the DB2 table columns. For example, columns with column types that are not supported by DB2 Analytics Accelerator are omitted. After the table is defined on the Accelerator, you can load data into that table by copying data from the original DB2 table to the corresponding table on the Accelerator.

Alternatively, you can replicate changed data continuously from the DB2 table to the accelerator-shadow table. Another method is to use the DB2 Analytics Accelerator Loader to load data into IBM DB2 for z/OS and the Accelerator at the same time.

The data exists in both tables, in the DB2 table and in the accelerator-shadow table. However, dependent on load or replication frequencies, the accelerator-shadow table contains a snapshot of the data of the DB2 table at a certain point in time.

How tables are loaded is explained in the *IBM DB2 Analytics Accelerator Version 4.1 User's Guide*, SH12-7040, found at the following link:

<http://publibfp.dhe.ibm.com/epubs/pdf/h1270404.pdf>

In addition, for an explanation about what a load strategy might look like, see *Reliability and Performance with IBM DB2 Analytics Accelerator V4.1*, SG24-8213:

<http://www.redbooks.ibm.com/redbooks/pdfs/sg248213.pdf>

The value of the **QUERY ACCELERATION** setting determines whether queries using these tables are routed to the Accelerator to run on the accelerator-shadow tables or stay on DB2 to run on the original DB2 tables. The value must be set to ELIGIBLE, ENABLE, ENABLE WITH FAILBACK, or ALL. The value is set by one of the following methods, as shown in Example 2-1.

Example 2-1 Options to set the QUERY ACCELERATION setting

-
- QUERY_ACCELERATION subsystem parameter
 - CURRENT QUERY ACCELERATION special register
 - QUERYACCELERATION bind option
 - Connection properties for applications connecting using JDBC or ODBC
 - Profile tables for remote applications
-

The following IBM Redbooks publications explain the concepts of Query Acceleration Management in more detail:

- *Reliability and Performance with IBM DB2 Analytics Accelerator V4.1*, SG24-8213:

<http://www.redbooks.ibm.com/redbooks/pdfs/sg248213.pdf>

- *Hybrid Analytics Solution using IBM DB2 Analytics Accelerator for z/OS V3.1*, SG24-8151:
<http://www.redbooks.ibm.com/redbooks/pdfs/sg248151.pdf>

Only read queries are routed to the Accelerator to run on accelerator-shadow tables. It is not possible to run DML statements on accelerator-shadow tables. If you use **INSERT** from **SELECT** statements, the **SELECT** part can be routed to the Accelerator and run on accelerator-shadow tables if the **ZPARM QUERY_ACCEL_OPTION** is set to option 2. Otherwise, the **SELECT** part is run in DB2. The **INSERT** part is always run in DB2.

2.1.3 Accelerator-archived table and partition

Figure 2-4 shows the architecture of an accelerator-archived table and partition.

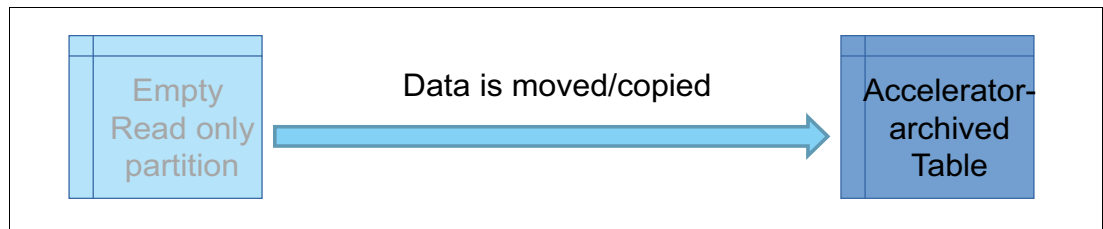


Figure 2-4 Accelerator-archived table and partition

An accelerator-archived table is a table on the Accelerator for which partitions of a range-partitioned DB2 table, or the entire range-partitioned table, have been moved (archived) to the Accelerator. The accelerator-archived tables are used primarily for historical data that is no longer actively used or maintained.

When the data is archived on the Accelerator, the original DB2 partitions are emptied to save storage space on IBM System z. The table spaces of DB2 table partitions that are associated with an accelerator-archived table are set to a persistent read-only state so that the original DB2 table can no longer be changed. An image copy of the data is also created as part of the archive process, to enable restoring the data in DB2.

To include accelerator-archived tables in your queries, set the **GET_ACCEL_ARCHIVE** special register, **ZPARM**, or **BIND** option to YES in addition to setting the **QUERY ACCELERATION** setting to a value other than NONE. How to set the **QUERY ACCELERATION** setting is explained in Example 2-1 on page 14.

2.1.4 Accelerator-only table (AOT)

Figure 2-5 shows the architecture of an accelerator-only table.

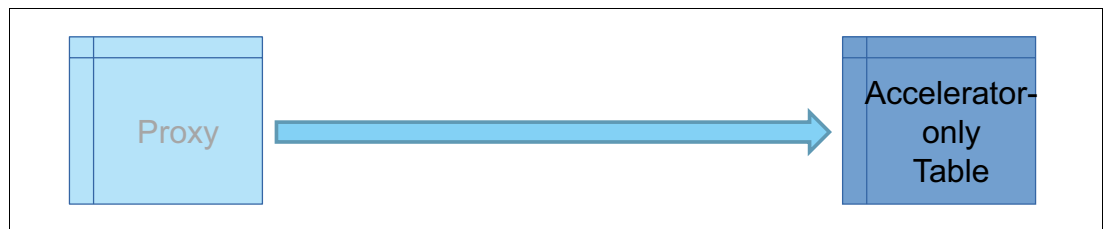


Figure 2-5 Accelerator-only table

An accelerator-only table is a table that stores rows only in the Accelerator, not in DB2. The table and column definition of the accelerator-only table is contained in DB2 catalog tables. The SYSIBM.SYSTABLES DB2 catalog table reflects an accelerator-only table by setting **TYPE** to D (definition-only). Additional entries are contained in the SYSIBM.SYSCOLUMNS and SYSACCEL.SYSACCELERATEDTABLES catalog tables, for example. the SYSACCELERATEDTABLES table contains the information about which Accelerator the accelerator-only table is located.

An accelerator-only table is created when the **CREATE TABLE** statement is issued in DB2 with the **IN ACCELERATOR <name>** clause where <name> determines the name of the Accelerator. The **CREATE** statement is explained in more detail in 2.3, “Syntax and capabilities” on page 18.

An accelerator-only table is dropped using the **DROP TABLE** statements. To create or drop an accelerator-only table the Accelerator must be started, but the **QUERY ACCELERATION** setting (see Example 2-1 on page 14) can have any value.

The **QUERY ACCELERATION** setting must be set to **ENABLE**, **ELIGIBLE**, or **ALL** to enable execution of queries or DML statements on accelerator-only tables.

Any queries that reference accelerator-only tables must be run in the accelerator. These queries can also join other tables on the Accelerator, either accelerator-shadow tables, accelerator-archived tables, or other accelerator-only tables. However, a query referencing an accelerator-only table can never join a non-accelerator DB2 table. If a query that references an accelerator-only table is not eligible for query acceleration, or if the **QUERY ACCELERATION** setting is not set correctly, an SQL error -4742 is issued (see Example 2-4 on page 20).

To change the contents of an accelerator-only table, the DML statement must be run in the Accelerator. This DML statement can reference other tables on the Accelerator, either accelerator-shadow tables, accelerator-archived tables or other accelerator-only tables. However, this DML statement can never reference a non-accelerator DB2 table.

Running queries and DML statements on the Accelerator can significantly speed up SQL statements, such as **INSERT** from **SELECT** statements. The complete **INSERT** from **SELECT** statement can run on the Accelerator, or must run on the Accelerator if the target table is an accelerator-only table. **INSERT** from **SELECT** statements targeting an accelerator-only table do not need the **ZPARM QUERY_ACCEL_OPTIONS** set. They are required for **INSERT** from **SELECT** statements that target DB2 tables, but that run the **SELECT** part on the Accelerator.

Accelerator-only tables are different from temporary tables. The DB2 catalog does not store a description of a declared temporary table for example. Therefore, the description and the instance of the table are not persistent. Multiple application processes can refer to the same declared temporary table by name, but they do not actually share the same description or instance of the table.

2.2 Software level prerequisites

Accelerator-only tables have been introduced together with other enhancements with DB2 Analytics Accelerator V4.1 PTF-5. The relevant authorized program analysis report (APAR) numbers and a complete list of prerequisites and co-requisites can be found on the following website:

<http://www.ibm.com/support/docview.wss?uid=swg27039487#IDAA%20V4%20PTF5>

For details about the delivered enhancements, see the Release Note on the following website:

<http://www.ibm.com/support/docview.wss?uid=swg27045592>

Accelerator-only table functionality also requires you to install the DB2 10 or DB2 11 PTFs listed in Table 2-1.

Table 2-1 APAR/PTF numbers

DB2 version	APAR / PTF number
DB2 10	PI30375 / UI26402 PI30376 / UI26404 PI30377 / UI26406 PI31445 / UI26408
DB2 11	PI35817 / UI29036 PI35818 / UI29037 PI35819 / UI29038 PI35820 / UI29039 PI35821 / UI29040

2.3 Syntax and capabilities

When planning to work with accelerator-only tables, the first step is to create them. Inserting data into accelerator-only tables or removing accelerator-only tables is also discussed in this chapter.

2.3.1 Creating accelerator-only tables

The **CREATE TABLE** DDL statement was enhanced, and Figure 2-6 shows the **CREATE TABLE** statement with the new **IN ACCELERATOR** clause.

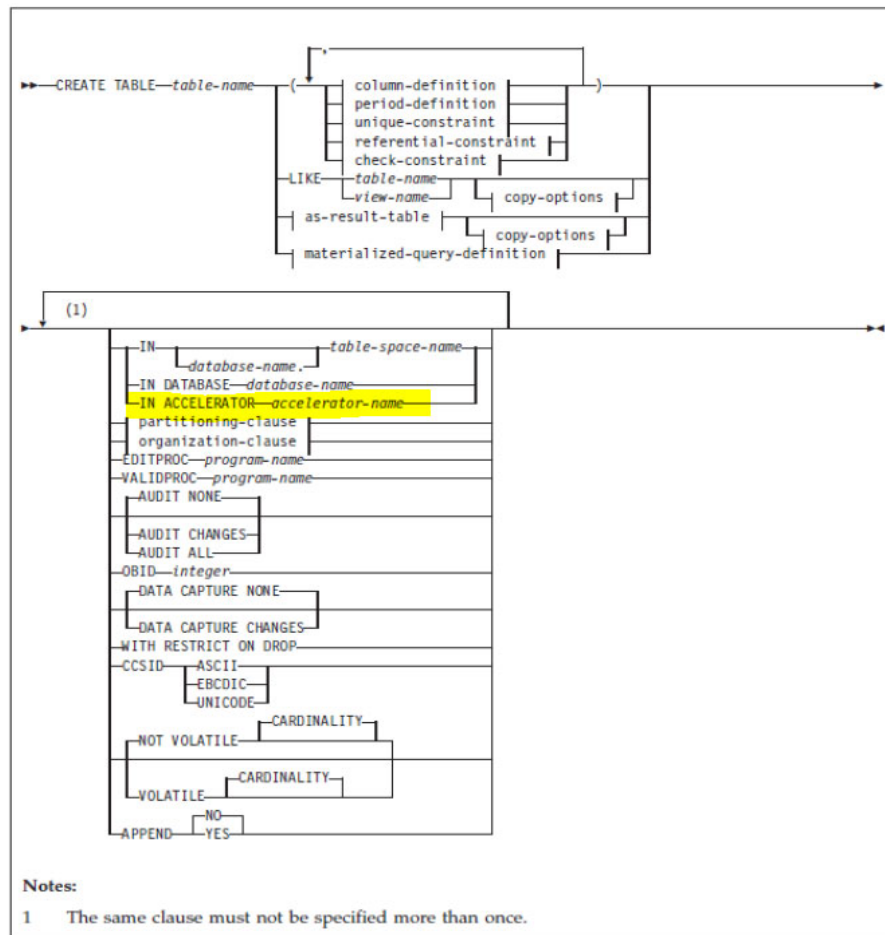


Figure 2-6 **CREATE TABLE** statement with **IN ACCELERATOR** clause

In Example 2-2, the table MYAOT1 is created as an accelerator-only table in the Accelerator ACCEL1. This is driven by the new **IN ACCELERATOR** clause, which specifies that the table is an accelerator-only table, and *<accelerator-name>* specifies the name of the Accelerator that the table will be defined in.

Example 2-2 Create accelerator-only table

```
CREATE TABLE MYAOT1 (..column definitions..) IN ACCELERATOR ACCEL1 IN DATABASE
AOTDB;
COMMIT;
```

For the complete **CREATE TABLE** statement, see the following IBM Knowledge Center:

http://www.ibm.com/support/knowledgecenter/SSEPEK_11.0.0/com.ibm.db2z11.doc.sqlref/src/tpc/db2z_sql_createtable.dita?lang=en

Specify a database name, for example AOTDB, to avoid the implicit creation of a database name by DB2. This simplifies the management of accelerator-only tables (see also 2.5, “Lifecycle management” on page 24).

As already mentioned in 2.1.4, “Accelerator-only table (AOT)” on page 15, the setting of **QUERY ACCELERATION** is not important for the **CREATE TABLE** and **DROP TABLE** statement. I/U/D operations require a setting of ELIGIBLE, ENABLE, or ALL.

Only by using the **CREATE TABLE** statement can an accelerator-only table be created. Column type restrictions of accelerator-shadow tables apply to accelerator-only tables as well. Accelerator-only tables cannot be created using the “Add tables” dialog in DB2 Analytics Accelerator Studio. After an accelerator-only table is created, it is shown in the list of tables in DB2 Analytics Accelerator Studio. A dedicated icon and the operational state identifies a table as an accelerator-only table, as shown in Figure 2-7.

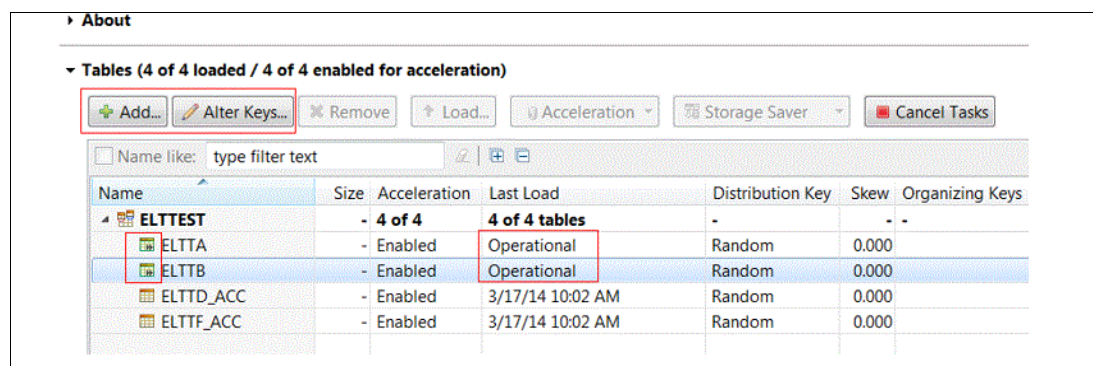


Figure 2-7 Data Studio view of accelerator-only tables

Each accelerator-only table creation statement must be in one unit of work (UOW). Therefore, Example 2-3 does *not* work, in contrast to Example 2-2 on page 18. It is missing a **COMMIT** statement between the two **CREATE TABLE** statements, and would result in SQLCODE = -571, ERROR: THE STATEMENT WOULD RESULT IN A MULTIPLE SITE UPDATE.

Example 2-3 Create statement not working (missing COMMIT statement)

```
CREATE TABLE MYAOT1 (..column definitions..) IN ACCELERATOR ACCEL1 IN DATABASE AOTDB;
CREATE TABLE MYAOT2 (..column definitions..) IN ACCELERATOR ACCEL1IN DATABASE AOTDB;
COMMIT;
```

2.3.2 Inserting data into accelerator-only tables

Now that the accelerator-only table MYAOT1 is defined in the Accelerator ACCEL1, it does not contain any rows. Accelerator-only tables can be populated with data that already exists on the Accelerator, which means from accelerator-shadow tables, accelerator-archived tables, or other accelerator-only tables. Alternatively, you can insert single rows with specified values.

Traditional methods (for example, running the stored procedure **ACCEL_LOAD_TABLES**) do not work.

Use one of the two DB2 **INSERT** statement forms to insert rows into the accelerator-only tables:

- ▶ The **INSERT** using **VALUES** form is used to insert a single row into the table using the values provided or referenced. At the time of writing, multi-row insert is not supported.
- ▶ The **INSERT** using **SELECT** form is used to insert one or more rows into the table using values from other tables on the Accelerator.

The possibility to use any kind of **SELECT** statement eligible for acceleration opens the horizon for any kind of data manipulation, such as aggregation, summarization, or cleansing, while populating an accelerator-only table. At the same time you get the benefit from the Accelerator's query performance.

The tables that you reference in the **SELECT** statement must exist on the Accelerator already. You can combine multiple table types (accelerator-shadow, accelerator-only, and accelerator-archived) in one **SELECT** statement.

An **INSERT** into an accelerator-only table statement is always routed to the Accelerator if the **QUERY ACCELERATION** setting is set correctly. Otherwise, it will fail with `SQLCODE=-4742` and reason code 2, as you can see in Example 2-4.

Example 2-4 Query Accelerator setting not set correctly

```
THE STATEMENT CANNOT BE run BY DB2 OR IN THE ACCELERATOR (REASON 2).  
SQLCODE=-4742, SQLSTATE=560D5, DRIVER=4.16.53
```

INSERT using VALUES form

The **INSERT** using **VALUES** form is used to insert a single row into the table using the values provided or referenced. With the **INSERT** statement shown in Example 2-5, three rows are inserted in MYAOT1. At the time of writing, multi-row insert is not supported.

Example 2-5 INSERT using VALUES example

```
CREATE TABLE MYAOT1 (COL1 VARCHAR(10)) IN ACCELERATOR ACCEL1 IN DATABASE AOTDB;  
COMMIT;  
INSERT INTO MYAOT1 VALUES ('TEST');  
INSERT INTO MYAOT1 VALUES ('TEST1');  
INSERT INTO MYAOT1 VALUES ('TEST2');  
COMMIT;
```

INSERT using SELECT form

The **INSERT** using **SELECT** form is used to insert one or more rows into the table using values from other tables on the Accelerator.

An accelerator-only table can be used as the target table in an **INSERT** with **SELECT** statement if *all* tables referenced in the **SELECT** clause exist on the same accelerator. You can combine multiple table types (accelerator-shadow, accelerator-only, and accelerator-archived) in the **SELECT** clause. If the **SELECT** part references a non-accelerator DB2 table, an error occurs.

With the **INSERT** statement in Example 2-6 on page 21, all rows from the existing accelerator-shadow table ASHADOW1 are inserted in MYAOT1. The coded character set identifier (CCSID) of the accelerator-only table must correspond to the table CCSID from which an accelerator-only table is populated.

Example 2-6 INSERT using SELECT example

```
CREATE TABLE MYAOT1 (COL1 VARCHAR(10)) IN ACCELERATOR ACCEL1 IN DATABASE AOTDB;  
COMMIT;  
INSERT INTO MYAOT1 (SELECT * FROM ASHADOW1);  
COMMIT;
```

If the target table is an accelerator-only table, the complete **INSERT** statement is routed to the Accelerator. The **QUERY ACCELERATION** setting must be set to **ELIGIBLE**, **ENABLE**, or **ALL**. Setting the **ZPARM QUERY_ACCEL_OPTIONS** to option 2 is not required. It *is* required for **INSERT** from **SELECT** statements that target a non-accelerator DB2 table and run the **SELECT** part on the Accelerator.

Because you insert data from other tables on the Accelerator, for example accelerator-shadow tables into accelerator-only tables, remember that accelerator-shadow tables might not contain the current data that your DB2 tables do. How current the data is depends on your load or replication strategies. For more details, see also 2.1.2, “Accelerator-shadow table” on page 14.

Notes: The following list provides a summary about the **INSERT** using **SELECT** form:

- ▶ All tables referenced in the **INSERT** and **SELECT** part must exist in the same accelerator.
- ▶ The coded character set identifier (CCSID) of the accelerator-only table needs to correspond to the table CCSID from which an accelerator-only table is populated.
- ▶ If the target table of an **INSERT** statement is an accelerator-only table, the complete statement is routed to the Accelerator (if **QUERY ACCELERATION** settings allow).
- ▶ If the target table of an **INSERT** statement is a non-accelerator DB2 table, only the **SELECT** part of the statement is routed to the Accelerator (if the **ACCEL_QUERY_OPTIONS** setting allows).

INSERT data using the DB2 Analytics Accelerator Loader

Another possibility to load data into accelerator-only tables is described in Chapter 9, “Integrating more data sources and archiving for analytics” on page 165. In particular, data from other sources, or a huge amount of data, can be loaded in a convenient way.

2.3.3 Removing accelerator-only tables

To remove an accelerated-only table, use the **DROP TABLE** statement or the “Remove tables” dialog in Data Studio, which calls the **DROP TABLE** statement implicitly. DB2 knows the Accelerator to which the accelerator-only table belongs. The **DROP TABLE** statement removes the table from the Accelerator, and deletes the entry from the **SYSACCELERATEDTABLES** catalog table as well. Each **CREATE** or **DROP** statement must run in its own DB2-side transaction and, therefore, must be completed by a **COMMIT** or **ROLLBACK** statement, as shown in Example 2-7.

Example 2-7 Drop table / database statement

```
DROP TABLE MYAOT1;  
COMMIT;  
DROP DATABASE AOTDB;  
COMMIT;
```

The Accelerator should be known and started; otherwise, a SQL warning +4748 occurs. In this case, the **DROP TABLE** statement nevertheless completes, but only deletes the “proxy” table in DB2, and an orphaned table remains on the Accelerator. Orphaned tables can be removed by using Data Studio or the **ACCEL_REMOVE_TABLES** stored procedures.

In Example 2-2 on page 18, we use the **IN DATABASE** clause when creating accelerator-only tables. This simplifies the removal of accelerator-only tables and their container databases, because you explicitly know the database to drop after dropping its accelerator-only tables.

Without specifying a database name, DB2 creates an implicit database for every accelerator-only table. Removing these is a cumbersome process, because you must figure out the implicitly created database names first before you can drop them. DB2 does not drop the implicitly created database automatically when its accelerator-only table is dropped.

A **DROP DATABASE** statement does not remove a database and the tables therein if some of the tables are accelerator-only tables. To remove such a database, first remove all accelerator-only tables by using separate **DROP TABLE** statements, and then drop the database.

2.3.4 Using accelerator-only tables in queries or DML statements

The same restrictions apply for queries running against accelerator-only tables as for accelerator-shadow tables or accelerator-archived tables. The conditions for successful query routing to the Accelerator are described in the following IBM Knowledge Center:

https://www.ibm.com/support/knowledgecenter/SS4LQ8_4.1.0/com.ibm.datatools.aqt.doc/gui/concepts/c_idaa_query_offloading_criteria.html

Accelerator-only table can be referenced in accelerated queries if all tables referenced in the query exist on the same Accelerator. An accelerator-only table can be referenced in an **INSERT**, **UPDATE**, or **DELETE** statement if other tables referenced in the statement exist on the same accelerator. Statements referencing accelerator-only tables must be run in the Accelerator. The **QUERY ACCELERATION** setting must be set to **ENABLE**, **ELIGIBLE**, or **ALL**.

Note: The following list provides a summary about using accelerator-only tables in queries:

- ▶ Queries that reference accelerator-only tables must be run in the Accelerator and not in DB2.
- ▶ All tables must exist in the same accelerator.
- ▶ SQL statements that update (I/U/D) the content of accelerator-only tables must be run in the Accelerator and not in DB2.

2.4 Transactional considerations

DB2 used in conjunction with the Accelerator does not support two-phase commit. Only single-phase commit is supported. All uncommitted non-read-only operations must occur on one side, either on DB2 or on the Accelerator, but not on both. A single unit of work must not contain statements that modify more than one side. If it goes to the Accelerator, a single transaction must run in the same Accelerator, not in multiple.

If you run a statement that touches more than one side, it fails with message **DSNT408I** **SQLCODE = -817, ERROR: The SQL statement cannot be run because the statement will result in a prohibited data change operation.**

For example, the scenarios shown in Figure 2-8 are not valid.

Scenarios	Example
I/U/D touches 2 Accelerators in one unit of work	INSERT INTO T1_ACCELERATOR1_ONLY ...; INSERT INTO T2_ACCELERATOR2_ONLY ...; COMMIT;
I/U/D Non-accelerator DB2 table followed by I/U/D accelerator-only table	INSERT INTO NON_ACCELERATOR_DB2_ONLY_TABLE ...; INSERT INTO T1_ACCELERATOR1_ONLY ...; COMMIT;
I/U/D accelerator-only table followed by I/U/D Non-accelerator DB2 table	INSERT INTO T1_ACCELERATOR1_ONLY ...; INSERT INTO NON_ACCELERATOR_DB2_ONLY_TABLE ...; COMMIT;
Two CREATE accelerator only table in one unit of work	CREATE TABLE T1_ACCELERATOR1_ONLY ... IN ACCELERATOR; CREATE TABLE T2_ACCELERATOR1_ONLY ... IN ACCELERATOR; COMMIT;
CREATE accelerator only table followed by I/U/D statement where target table is accelerator- only table	CREATE TABLE T1_ACCELERATOR1_ONLY ... IN ACCELERATOR; INSERT INTO T1_ACCELERATOR1_ONLY ...; COMMIT;
CREATE accelerator-only table preceded by I/U/D statement where target table is accelerator-only table (same/different accelerator)	INSERT INTO T1_ACCELERATOR1_ONLY ...; CREATE TABLE T1_ACCELERATOR1_ONLY ... IN ACCELERATOR; COMMIT; INSERT INTO T1_ACCELERATOR1_ONLY ...; CREATE TABLE T1_ACCELERATOR2_ONLY ... IN ACCELERATOR; COMMIT;

Figure 2-8 Commit scope restrictions

Conceptually, the **CREATE TABLE ... IN ACCELERATOR** or the **DROP TABLE** statement for an accelerator-only table modifies both sides, because they create or remove a table on the Accelerator side and also update the DB2 catalog. Therefore, these statements must be run in a separate unit of work, and cannot be combined with a statement modifying the DB2 side or the Accelerator side. Example 2-8 shows how to code the statements in the correct sequence.

Example 2-8 Valid coding order of statements

```
CREATE TABLE T1_ACCELERATOR1_ONLY... IN ACCELERATOR A1;
COMMIT;
CREATE TABLE T2_ACCELERATOR1_ONLY... IN ACCELERATOR A1;
COMMIT;
INSERT INTO T1_ACCELERATOR1_ONLY... SELECT...;
INSERT INTO T2_ACCELERATOR1_ONLY... SELECT...;
SELECT FROM T1_ACCELERATOR1_ONLY, T2_ACCELERATOR1_ONLY,....;
COMMIT;
```

Notes: The following list provides a quick summary about SQL statements in Accelerator:

- ▶ A single unit of work must not contain statements that modify more than one side (Accelerator or DB2).
- ▶ A single unit of work can either modify the DB2 site or the accelerator side.
- ▶ A single unit of work cannot modify more than one Accelerator.

2.5 Lifecycle management

Use accelerator-only tables where currently large amounts of data are moved away from DB2 for the following activities:

- ▶ To transform it
- ▶ To load the transformed data back to DB2
- ▶ To query the transformed data later

Use accelerator-only tables for intermediate results where source data is on the Accelerator already, and where it is easy to regenerate the intermediate results.

Accelerator-only tables should be created in one or more separate DB2 databases depending on your application needs. This makes life easier with the lifecycle management of accelerator-only tables, because you explicitly know the database names and do not have to figure out the DB2 generated database name(s) for later deletion.

A **DROP DATABASE** statement does not remove a database and the tables therein if some of the tables are accelerator-only tables. To remove such a database, first remove all accelerator-only tables by using separate **DROP TABLE** statements, and then drop the database.

Example 2-9 shows what a possible sample flow could look like.

Example 2-9 Lifecycle management sample flow

```
SET CURRENT QUERY ACCELERATION = ENABLE;
CREATE TABLE T1_ACCELERATOR1_ONLY... IN ACCELERATOR A1 IN DATABASE ...;
COMMIT;
CREATE TABLE T2_ACCELERATOR1_ONLY... IN ACCELERATOR A1 IN DATABASE ...;
COMMIT;
INSERT INTO T1_ACCELERATOR1_ONLY... SELECT...;
INSERT INTO T2_ACCELERATOR1_ONLY... SELECT...;
SELECT FROM T1_ACCELERATOR1_ONLY, T2_ACCELERATOR1_ONLY,...;
COMMIT;
DROP TABLE T1_ACCELERATOR1_ONLY;
COMMIT;
DROP TABLE T2_ACCELERATOR1_ONLY;
COMMIT;
DROP DATABASE ...;
```

Rather than dropping the already defined accelerator-only tables in Example 2-9, we could also think of another scenario where we do not drop the already created accelerator-only tables. Perform a “Delete all data from accelerator-only table” action so that the data in the accelerator-only tables is deleted, and then run an **INSERT INTO** statement again into the accelerator-only tables.

Of course this depends on the usage scenario, but is at least a potential option avoiding **CREATE** and **DROP** table statements that are not necessary in certain cases.

Accelerator-only tables exist only in the Accelerator, therefore no DB2 backup and recovery support is available. For this reason, it makes sense to ensure that accelerator-only tables are recoverable, which means that the jobs and their original data that have been used for the creation of accelerator-only tables are saved for later recovery activities. You can also use **INSERT INTO SELECT** statements to insert data from an accelerator-only table into a DB2-only table to make use of DB2-supplied backup and recovery capabilities.

2.6 Limitations and restrictions

The following stored procedures are supported with accelerator-only tables:

- ▶ **ACCEL_GET_TABLES_INFO**
- ▶ **ACCEL_GET_TABLES_DETAILS**
- ▶ **ACCEL_REMOVE_TABLES**

To clean up orphans, **DROP TABLE** is the usual method.

- ▶ **ACCEL_ALTER_TABLES**

To change distribution key columns.

- ▶ **ACCEL_LOAD_TABLES**

With DB2 Analytics Accelerator Loader in append-only mode. Removing old data must be done using **DELETE** statement. Concurrent invocation of multiple stored procedure calls for the same table is not supported.

- ▶ **ACCEL_CONTROL_ACCELERATOR**

List or cancel tasks related to accelerator-only tables.

All other stored procedures will fail with a corresponding message.

Table 2-2 shows some popular stored procedures that are not supported with accelerator-only tables. It also shows the rationale behind it or the circumvention to use.

Table 2-2 Selected stored procedures and accelerator-only tables

Stored procedure name	Circumvention and rationale
SYSPROC.ACCEL_ADD_TABLES	Submit a CREATE TABLE ... IN ACCELERATOR statement instead.
SYSPROC.ACCEL_SET_TABLES_REPLICATION	Accelerator-only tables cannot be enabled for incremental updates.
SYSPROC.ACCEL_ARCHIVE_TABLES	Accelerator-only tables cannot be archived.
SYSPROC.ACCEL_RESTORE_ARCHIVE_TABLES	Accelerator-only tables cannot be restored as they cannot be archived.
SYSPROC.ACCEL_SET_TABLES_ACCELERATION	Accelerator-only tables exist on an Accelerator only and therefore are by themselves enabled for acceleration.

For the creation of accelerator-only tables, certain restrictions currently exist. The most important are shown in Figure 2-9.

Unsupported data type (LOB, XML ...)	Invalid CCSID - no support for ASCII mixed or ASCII graphic	PRIMARY KEY, UNIQUE KEY, FOREIGN KEY	CHECK constraint
DEFAULT	GENERATED	AS SECURITY LABEL	FIELDPROC
IMPLICITLY HIDDEN	Unicode column in EBCDIC table CCSID UNICODE	LIKE with IN ACCELERATOR keyword	LIKE referencing accelerator only table
AS fullselect with IN ACCELERATOR option	As fullselect (full select reference the accelerator only table	MQT definition with IN ACCELERATOR option	MQT definition references accelerator-only table
HASH	COMPRESS	APPEND	DSSIZE
PARTITION BY	BUFFERPOOL	MEMBER CLUSTER	TRACKMOD
VALIDPROC, EDITPORC	IN TABLESPACE	DATA CAPTURE	LOGGED,NOT LOGGED
AUDIT	VALATILE	IN ACCELERATOR A1, A2 (Multiple accelerators)	Accelerator must be active

Figure 2-9 Restrictions for the creation of accelerator-only tables

You can find a detailed list of restrictions in the following IBM Knowledge Center:

https://www.ibm.com/support/knowledgecenter/SS4LQ8_4.1.0/com.ibm.datatools.aqt.doc/gui/references/r_idaa_restrictions_aots.html

Incremental update and High Performance Storage Saver functionality is not supported for accelerator-only tables. They exist only in the Accelerator, therefore, no DB2 backup and recovery support is available. Utility statements, such as **LOAD**, **UNLOAD**, and **REORG**, are not allowed against accelerator-only tables.

2.7 Performance considerations

Specific performance measurements were performed to understand the behavior and differences in terms of Elapsed Time and CPU consumption of **INSERT** statements using the **INSERT** with **SELECT** form. The measurements compare **INSERT**s in partition-by-growth (PBG) tables and accelerator-only tables (AOTs). The setup for the performance measurements is as follows:

- ▶ EC12, 6 Central Processors (CP), 80 GB memory storage
- ▶ N3001-010 system with 7 Snippet Processing Units (SPU) and 254 active disks
- ▶ 10 GB Ethernet connection between DB2 and N3001 system (active - passive bonding)

The measurements cover the following items:

- ▶ **INSERT** with **SELECT** form in target tables (for example, PBG versus accelerator-only tables)
- ▶ Different number of rows for insert
- ▶ Different number of columns for insert
- ▶ Different data types for insert

It is assumed that the data is already loaded in the Accelerator. The results in Figure 2-10 on page 27 show that the higher the number of inserted rows is, the more efficient inserting into accelerator-only tables is compared to inserting into DB2 PGBs. The queries RI201 to RI208 show an advantage in Elapsed Time of up to 98%, especially when we **INSERT** 1 million rows up to 1 billion of rows. The CPU consumption on the Accelerator is not notably different.

CORR	#rows	Elapsed Sel_only	Elapsed PBG	Elapsed IDT	PBG: IDT	CPU PBG	CPU IDT
RI101	5	102.872	103.073	102.001	-1.04%	0.016	0.009
RI102	2	90.402	90.327	90.308	-0.02%	0.012	0.011
RI103	175	286.339	284.891	286.213	0.46%	0.010	0.008
RI201	340	2.300	3.572	1.827	-48.84%	1.930	0.004
RI202	3,400,000	21.911	29.850	11.574	-61.23%	16.638	0.003
RI203	1,200,000	6.275	9.040	3.585	-60.34%	6.232	0.004
RI204	12,000,000	62.790	76.125	13.209	-82.65%	53.957	0.003
RI205	1,000,000	4.825	9.545	1.579	-83.46%	5.871	0.002
RI206	10,000,000	44.334	63.162	3.484	-94.48%	48.265	0.002
RI207	100,000,000	441.504	673.987	27.159	-95.97%	567.315	0.002
RI208	1,000,000,000	canceled	14.863.803	290.583	-98.05%	13.879.277	0.002
RI301	1,000,000	4.769	9.574	1.562	-83.69%	5.927	0.002
RI302	1,000,000	5.826	12.905	2.339	-81.88%	7.177	0.002
RI303	1,000,000	7.169	15.379	2.762	-82.04%	8.670	0.002
RI401	1,000,000	4.312	7.976	0.880	-88.97%	5.289	0.002
RI402	1,000,000	4.380	9.236	1.220	-86.79%	5.413	0.002
RI403	1,000,000	4.048	8.413	0.903	-89.27%	5.959	0.001

Figure 2-10 Insert comparison different scenarios

The legend to Figure 2-10 is as follows:

- ▶ CORR. Query characteristics:
 - RI101 ~ RI103: Complex select, small number of rows for insert
 - RI201 ~ RI204: More rows for insert
 - RI301 ~ RI303: 16, 32, or 48 columns for select and insert
 - RI401 ~ RI403: Integer, char, date columns for select and insert
- ▶ #rows. Number of rows to be inserted.
- ▶ Elapsed Sel_only. Class 2 Elapsed Time Select part only including moving data back to DB2.
- ▶ Elapsed _PBG. Class 2 Elapsed Time Insert into PBG including moving data back to DB2.
- ▶ Elapsed _IDT. Class 2 Elapsed Time Insert into AOT. No moving of data back to DB2.
- ▶ PBG:IDT. Class 2 Elapsed Time comparison PBG versus AOT.
- ▶ CPU PBG. Class 2 CPU Time PBG.
- ▶ CPU IDT. Class 2 CPU Time AOT.

Figure 2-11 shows that the advantage of Class 2 Elapsed Time increases with the number of rows to be inserted in accelerator-only tables versus PBG tables.

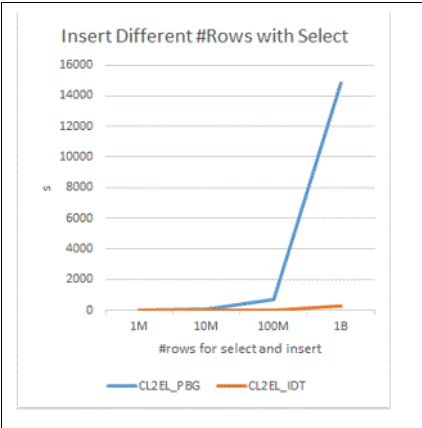


Figure 2-11 Insert different number of rows

Figure 2-12 shows that different number of columns has almost no effect on accelerator-only tables. The advantage of Class 2 Elapsed Time increases with the number of columns to be inserted.

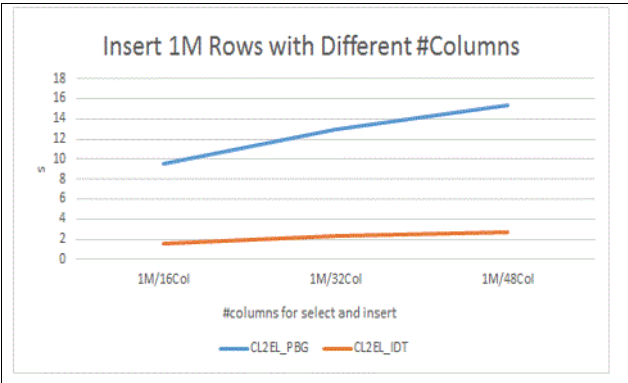


Figure 2-12 Insert 1 million rows with different number of columns

Figure 2-13 shows that different data types when inserting 1 million rows has almost no effect on accelerator-only tables.

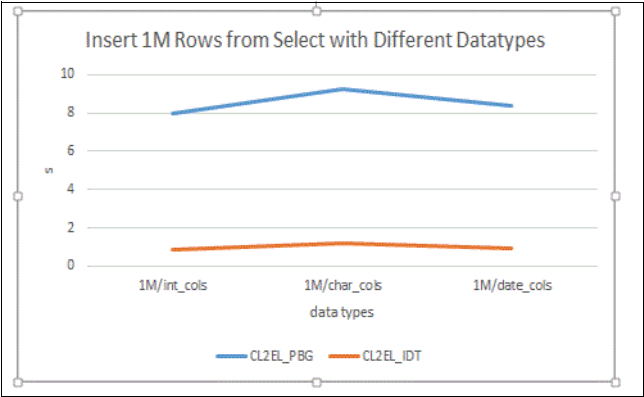


Figure 2-13 Insert 1 million rows with different data types

During the measurements, different performance patterns were observed. The most efficient performance patterns are those patterns where no redistribution is involved:

► **INSERT INTO AOT2 SELECT * FROM AOT1**

Most efficient because both tables are randomly distributed. No redistribution. **SELECT** and **INSERT** are pipelined on the same snippet processing unit (SPU).

► **INSERT INTO AOT1 SELECT * FROM LINEITEM**

Source table is key distributed. Target table is random distributed. There is data redistribution among the SPUs, but still efficient.

If several tables are joined and the result tables are grouped, or the number of rows is limited by a **FETCH FIRST** clause, redistribution among host and SPUs impacts the performance.

2.7.1 INSERT SELECT performance considerations

Insert-select statements are run massively parallel on the Accelerator, therefore copies of data can be created efficiently.

Example 2-10 shows the table layout and cardinality of a TPC-H SALES table, which we use for a set of performance experiments.

Example 2-10 Table layout of TPC-H SALES table for testing

Select	Column Name	Col	No	Col	Type	Length	Scale	Null	Def	FP	Col	Card
*				*		*		*	*	*		*
----->												
	SS_SOLD_DATE_SK	1			INTEGER	4		0	Y	Y	N	1856
	SS_SOLD_TIME_SK	2			INTEGER	4		0	Y	Y	N	44544
	SS_ITEM_SK	3			INTEGER	4		0	N	N	N	270336
	SS_CUSTOMER_SK	4			INTEGER	4		0	Y	Y	N	311296
	SS_CDEMO_SK	5			INTEGER	4		0	Y	Y	N	299008
	SS_HDEMO_SK	6			INTEGER	4		0	Y	Y	N	7552
	SS_ADDR_SK	7			INTEGER	4		0	Y	Y	N	328704
	SS_STORE_SK	8			INTEGER	4		0	Y	Y	N	388
	SS_PROMO_SK	9			INTEGER	4		0	Y	Y	N	1360
	SS_TICKET_NUMBER	10			INTEGER	4		0	N	N	N	335872
	SS_QUANTITY	11			INTEGER	4		0	Y	Y	N	104
	SS_WHOLESALE_COST	12			DECIMAL	7		2	Y	Y	N	9728
	SS_LIST_PRICE	13			DECIMAL	7		2	Y	Y	N	18688
	SS_SALES_PRICE	14			DECIMAL	7		2	Y	Y	N	17920
	SS_EXT_DISCOUNT_AM	15			DECIMAL	7		2	Y	Y	N	237568
	SS_EXT_SALES_PRICE	16			DECIMAL	7		2	Y	Y	N	425984
	SS_EXT_WHOLESALE_C	17			DECIMAL	7		2	Y	Y	N	352256
	SS_EXT_LIST_PRICE	18			DECIMAL	7		2	Y	Y	N	581632
	SS_EXT_TAX	19			DECIMAL	7		2	Y	Y	N	87040
	SS_COUPON_AMT	20			DECIMAL	7		2	Y	Y	N	237568
	SS_NET_PAID	21			DECIMAL	7		2	Y	Y	N	466944
	SS_NET_PAID_INC_TA	22			DECIMAL	7		2	Y	Y	N	655360
	SS_NET_PROFIT	23			DECIMAL	7		2	Y	Y	N	622592

We used a four million rows table with this layout and duplicated data inside the Accelerator ten times to get to higher row counts.

Then, we measured the time needed to run an **INSERT SELECT** statement without any further predicates or filtering.

Table 2-3 shows resulting times for different table sizes and key definitions in our environment.

Table 2-3 Elapsed times to copy data into an accelerator-only table

Table (Compressed size) in megabytes (MB) and gigabytes (GB)	Row count	Distribution key on source and target	Time for insert-select copy
Sales (210 MB)	4,000,000	Random	< 1 s
SalesD (210 MB)	4,000,000	SS_ITEM_SK	< 1 s
Sales10 (2 GB)	40,000,000	Random	5 s
Sales10D (2 GB)	40,000,000	SS_ITEM_SK	6 s
Sales100 (20 GB)	400,000,000	Random	49 s
Sales100D (20 GB)	400,000,000	SS_ITEM_SK	51 s
Sales1000 (200 GB)	4,000,000,000	Random	474 s
Sales1000D (200 GB)	4,000,000,000	SS_ITEM_SK	502 s

Because SS_ITEM_SK is not an ideal distribution key, data distribution results in a higher skew value (0.5). It is, however, interesting to observe that choosing the same distribution key for source and target table does not lead to faster copy times compared to random distribution on both source and target tables. The increase of the time needed to insert the different number of rows is linear to the increase of number of rows to be inserted.

Our example is not representative, and insert-select performance depends on table layout and other criteria as well.



Use cases that are enabled by accelerator-only tables and in-database analytics

Analytics has a direct effect on business performance, and is critical for improving decision-making, responding to business situations more quickly, and improving competitive advantage. The requirement to derive analytics from the most current, accurate data when the data enters an organization system is a top priority.

With the IBM DB2 Analytics Accelerator (Accelerator), clients whose data comes into their business through an IBM z Systems have the distinct advantage of a hybrid solution that enables them to run their analytics on the same platform as their transactional data. This hybrid, single-platform, integrated approach brings analytics closer to the transactional source data.

Having analytics closer minimizes the need to move data across platforms, reduces data latency, cost, and complexity, improves data governance, and reduces risk. Therefore, business users across the enterprise have access to more timely, accurate, and near real-time insight for improved decision making.

This chapter describes four major use cases for the DB2 Analytics Accelerator. We provide an overview of the four use cases, followed by a brief description about the new functions and features. This chapter also describes how the potential usage of the Accelerator is extended by the introduction of accelerator-only tables in Version 4.1 PTF 5, and by in-database analytics in Version 5.1. In each use case, we describe what the use case is before and after using these new features. This chapter covers the following topics:

- ▶ The four use cases of the DB2 Analytics Accelerator
- ▶ How accelerator-only tables and in-database analytics extend use cases
- ▶ Acceleration of existing business critical queries
- ▶ Derive business insight from z/OS transaction systems
- ▶ Reduce IT sprawl for analytics initiatives
- ▶ Improve access to historical data and lower storage costs
- ▶ Summary

3.1 The four use cases of the DB2 Analytics Accelerator

Looking at DB2 Analytics Accelerator usage scenarios is looking at where the transactional source data stored in DB2 for z/OS is analyzed today. There are four major use cases defined. Figure 3-1 shows a brief description of these four major use cases.

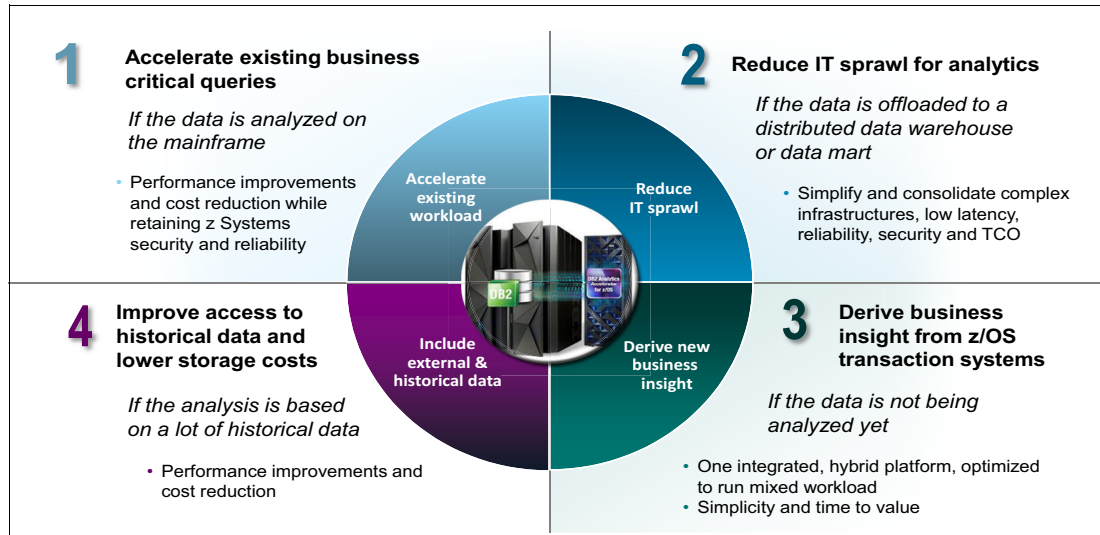


Figure 3-1 Current use cases for DB2 Analytics Accelerator

These four uses cases are described in more detail in the following list:

► Using Accelerator to *accelerate business-critical queries*

If the data is analyzed on IBM DB2 for z/OS, the Accelerator can be attached to that DB2 system. Thereby, the overall customer solution benefits from performance improvements and cost reduction, while retaining the security and reliability of IBM System z.

► Using Accelerator to *reduce information technology (IT) sprawl for analytics*

If the data is offloaded to a distributed data warehouse or data mart, the Accelerator could help to simplify and consolidate complex infrastructures and reduce time-consuming extract, transform, and load (ETL) processes. This process lowers data latency, enhances reliability, increases security, and decreases overall total cost of operation (TCO).

► Using Accelerator to *derive business insight from z/OS transaction systems*

Consider a case in which the data is not being analyzed yet, because queries are set aside due to performance or cost challenges, or because a new business intelligence (BI) applications is being considered. The Accelerator can be attached to the z/OS system to complement DB2 for z/OS, forming a hybrid infrastructure that best hosts this new workload.

► Using Accelerator to *improve access to historical data and lower storage costs*

If the analysis is based on a large amount of static historical data within DB2 for z/OS, the Accelerator can be used to improve performance and reduce costs for this analytical workload.

3.2 How accelerator-only tables and in-database analytics extend use cases

Version 5.1 of DB2 Analytics Accelerator opens a new dimension of analytical processing by introducing accelerator-only tables. Accelerator-only tables can enhance statistics and analytics tools that use temporary objects for reports. The tools are enhanced because the high velocity of the accelerator execution enables these tools to quickly gather all required data by using accelerator-only tables to implement their temporary objects.

Accelerator-only tables can also be used to accelerate data transformations that are implemented using Structured Query Language (SQL) statements. To take full advantage of the product's high-speed capabilities, it is now possible to store all interim results in accelerator-only tables. This enables subsequent queries or data transformations to process all relevant data on the accelerator, and as such to implement high-speed in-database transformations (IDT).

The accelerator-only tables and in-database analytics extend the value of DB2 Analytics Accelerator for the use cases described in 3.1, "The four use cases of the DB2 Analytics Accelerator" on page 32 as follows:

- ▶ Accelerating existing business critical queries

The accelerator-only tables can be used with IBM Campaign (Unica), IBM DB2 Query Management Facility (IBM QMF), or custom applications. They can also be used with other reporting tools, such as MicroStrategy. This feature broadens the use cases for accelerating critical queries of your existing business, and enables *a deeper insight into operational status through faster reporting*.

- ▶ Reduce IT sprawl for analytics

The accelerator-only tables can be used to *simplify data-transformation processes*. With the flexible data infrastructure of accelerator-only tables, you can eliminate data marts. In-database transformation within DB2 Analytics Accelerator is for reducing complexity, latency, and cost for transformations.

- ▶ Deriving business insight from z/OS transaction systems

Simply create a work area for a data scientist by defining a set of accelerator-only tables, to enable *deeper insight into customers and markets*.

Accelerate turnaround for predictive analytics applications by using in-database analytics. IBM SPSS and IBM Netezza Analytics data mining and in-database modeling can now be processed within the Accelerator. This enables *real-time modeling* to adapt your analytics models closer to reality.

- ▶ Improve access to historical data and lower storage costs

Simpler data integration, by using DB2 Analytics Accelerator Loader for z/OS to load non-DB2 for z/OS data.

3.3 Acceleration of existing business critical queries

Many organizations that have most of their operational data captured and housed on the IBM z platform have created their data warehouses on z Systems to take advantage of the platform's qualities of service. However, data-intensive queries that characterize data warehouse, business intelligence, and analytics workload are typically complex and long-running. Historically, running these queries in parallel with online transaction processing (OLTP) workload presented a challenge.

Due to their processor (CPU)-intensive profile, these queries might require extensive tuning efforts. In some cases, the queries cannot be run, and they became “forgotten queries”. Alternatively, they might need to be scheduled in batch processes overnight so that they do not affect corporate users during the day. However, such overnight schedules do not deliver information in a timely manner required in today's business critical environment.

Figure 3-2 shows an overview of the DB2 Analytics Accelerator usage scenario for rapid acceleration of business-critical queries.

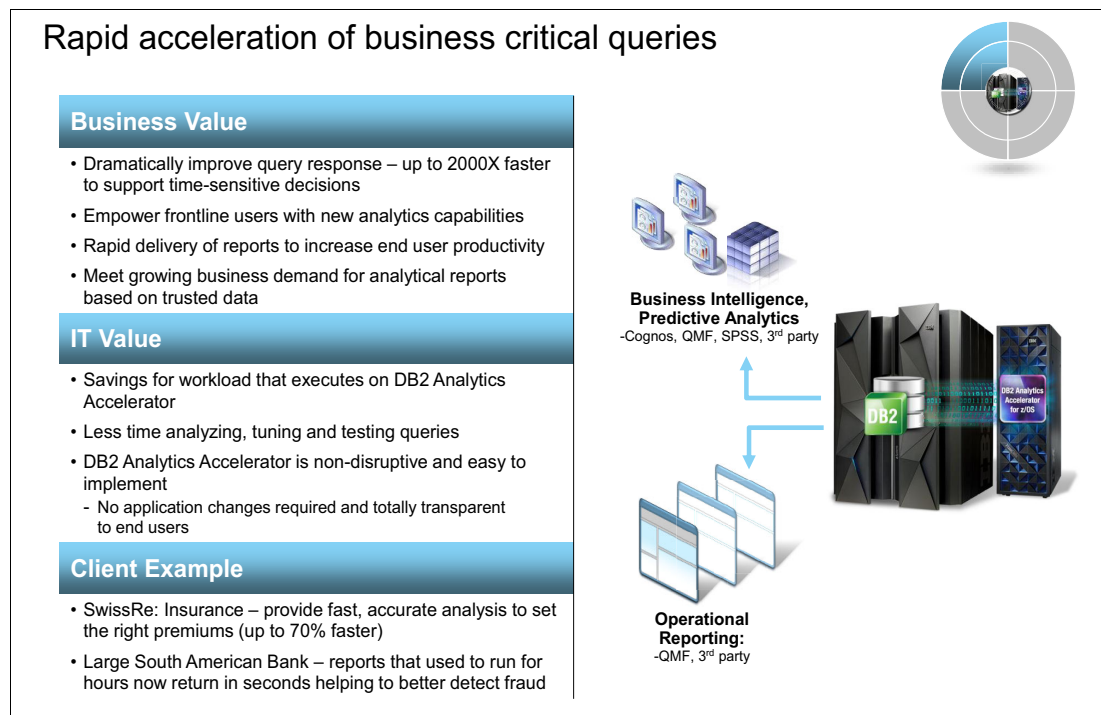


Figure 3-2 Overview of rapid acceleration of business-critical queries usage scenario

What the DB2 Analytics Accelerator already delivers

Extending DB2 for z/OS with an IBM DB2 Analytics Accelerator enables order-of-magnitude improvements to query response, to support time-sensitive decisions. Organizations can gain great advantage from this speed:

- ▶ Make better business decisions from the most up-to-date information
- ▶ Increase user productivity by delivering reports and information much faster
- ▶ Meet the growing business demand for analytical reports based on trusted data

This unique approach also removes the need to analyze, tune, and test analytic queries, and can reduce the time and effort required to manage DB2 for z/OS database applications in support of complex queries.

As a client example, consider a global insurance and reinsurance provider, SwissRe¹, who has claim data from numerous insurance companies that spread across multiple international locations. This company needs a way to obtain insight from all of their data to deliver insight to their business users quickly. This capability enables users to better understand risk, and to identify profitable segments. With DB2 Analytics Accelerator, this insurance company can now provide fast, accurate analysis from claim transactions in far less time than before.

The users of this company get their reports as much as 70 percent faster. Reports that took 10 hours are now available the same day, and user satisfaction has increased dramatically. Because those reports contain key analytics that guide pricing and decision-making across their lines of business, the solution has sharpened the company's competitive edge.

What is new with accelerator-only tables

DB2 Analytics Accelerator version 5.1 delivers new functionality to implement new usage scenarios for a deeper insight into operational status through faster reporting.

Information is power if you know how to extract value and insights out of it. The more that is known about a particular issue, situation, product, organization, or individual, the greater the likelihood of a better decision and business outcome.

In today's fast-paced business climate, companies must make quick decisions to win new business, maintain existing relationships, and build on the trust and goodwill earned over time. All larger companies rely on business intelligence reporting tools to gain a competitive edge. Reporting tools deliver ease of use for front-line workers, and for executives and managers, to give them deep insight into the status of their business.

This operational reporting is about operational detail (data) that is generated by operational processes, and mostly stored in operational databases (so-called online transaction processing (OLTP) systems). One OLTP system with large traction among the largest enterprises in the world is DB2 for z/OS.

Today, most business analytics are based on information that is stored in enterprise data warehouses, which in turn are fed mainly from transaction and operational systems. This data is rich in value, trusted, and understood. Used by most of the global banks and top US retailers, IBM System z holds a significant amount of business-critical information of the world. To be of benefit, operational reporting based on the enterprise data must provide current information to its users.

¹ See *IBM helps transform workload management and global system access* at http://www.ibm.com/common/ssi/cgi-bin/ssialias?subtype=AB&infotype=PM&appname=SWGE_ZS_ZS_USEN&htmlfid=ZSC03128USEN&attachment=ZSC03128USEN.PDF

Figure 3-3 shows an overview of the new usage scenario, delivering deeper insight into operational status through faster reporting, with DB2 Analytics Accelerator version 5.1.

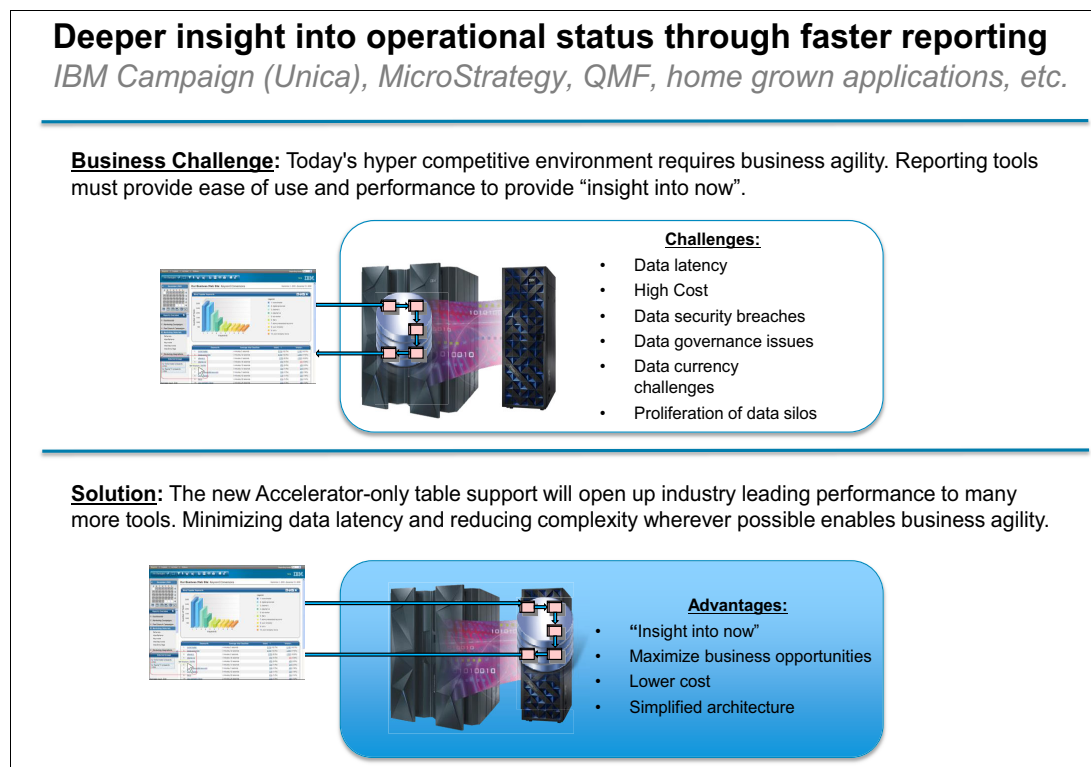


Figure 3-3 Usage scenario with DB2 Analytics Accelerator version 5.1

Companies who have IBM Campaign, MicroStrategy, QMF, and other similar tools can benefit from using these tools directly against operational data within DB2 for z/OS by means of the accelerator-only tables of the Accelerator. Frequently, the result from single-step queries might not be informative enough, and the performance might not be adequate. Running subsequent queries against the result from a previous query is required to have a more in-depth data analysis.

Currently, the common approach is to create local tables by using stored procedures, or a second data set stored with data offloaded from DB2 for z/OS. However, stored procedures are not flexible, and the ability to support ad hoc access is limited. Requiring an intermediary data source also implies more latency, more cost, data governance issues, and complexity.

Accelerator with the accelerator-only tables support enables these data analysis tools to create a local table on the Accelerator to store the intermediate results for subsequent queries. This support for accelerator-only tables extends query acceleration, and provides industry-leading performance to reporting tools such as MicroStrategy, QMF, custom applications, and quicker turnaround for iterative IBM Campaign Management processing. Minimizing data latency and reducing complexity wherever possible enables business agility.

3.4 Derive business insight from z/OS transaction systems

More and broader sets of users are now demanding access to analytics, from business users who need timely, accurate insight to do their jobs better, to customers who expect access to information any time, 24 x 7. These users have high expectations about the quality and time frame in which analytic insight is made available to them.

Figure 3-4 shows an overview of the *derive business insight* use case scenario with DB2 Analytics Accelerator.

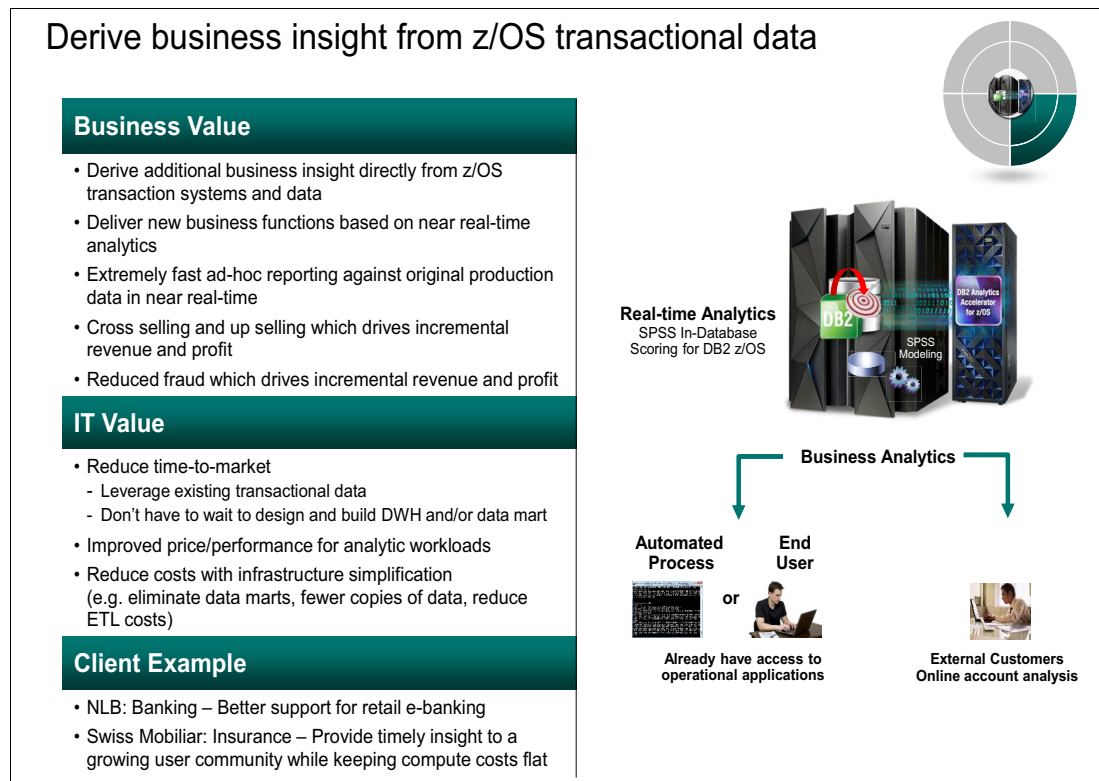


Figure 3-4 *Derive business insight* usage scenario

Because analytics have a direct effect on business performance, near real-time analysis is critical for improving decision making, responding to business situations faster, and improving competitive advantage. Therefore, the requirement to deliver near real-time analytics is crucial to business success.

Clients whose data comes into their business through an IBM z Systems mainframe have the distinct advantage of being able to run their analytics on the same platform as their transactional data. This hybrid, single-platform, and integrated approach means that business users get access to timely, accurate, and near real-time insight for improved business performance.

What the DB2 Analytics Accelerator already delivers

The introduction of DB2 Analytics Accelerator in conjunction with IBM z Systems brought a revolutionary change, resulting in a hybrid computing platform on z Systems. This hybrid infrastructure blends the best attributes of DB2 for z/OS, which is built for transactional workloads, with a cost-effective, high-speed query engine to run complex business analytics workloads.

DB2 Analytics Accelerator together with DB2 for z/OS forms a self-managing, hybrid database management system that transparently runs each workload in the most efficient way, so each transaction (OLTP, batch, or analytic query) is run in its optimal environment for greatest performance and cost efficiency. In this way, DB2 Analytics Accelerator turns DB2 for z/OS into a universal database management system, capable of handling both transactional and analytical data.

Organizations have brought in the Accelerator to help them gain additional business insight directly from their z/OS transaction systems and data. This enables them to deliver new business functions based on real-time and near real-time analytics. These analytics support applications, such as cross-sell and up-sell to drive sales, or fraud detection to reduce risk and drive profits.

What is new with accelerator-only tables and in-database analytics

DB2 Analytics Accelerator version 5.1 delivers new functionality called *accelerator-only tables*, which enables you to implement new scenarios to offer a deeper insight into customers and markets.

Today's competitive environment requires organizations to anticipate what their customers will do. Understanding customer behavior enables organizations to maximize revenue, reduce customer churn, and improve customer relationships. Data science is at the core of this new approach to business improvement.

The most challenging business problem of a company that a data scientist faces is that he needs to explore and examine data from multiple sources to eventually discover a previously hidden insight, which in turn can provide a competitive advantage or address a pressing business problem. Transaction and operational systems are important data sources. A data scientist eventually needs to handle the data stored in DB2 for z/OS.

Although the line of business (LOB) needs answers to their business problems as quickly as possible, the IT organization needs to provide facilities to enable ad hoc and easy access to data stored in DB2 for z/OS. At the same time, data governance, security compliance, cost control, and service level agreement (SLA) for the query production workload must be ensured.

Figure 3-5 shows an overview of the *deeper insight into customers and markets* use case scenario with DB2 Analytics Accelerator version 5.1.

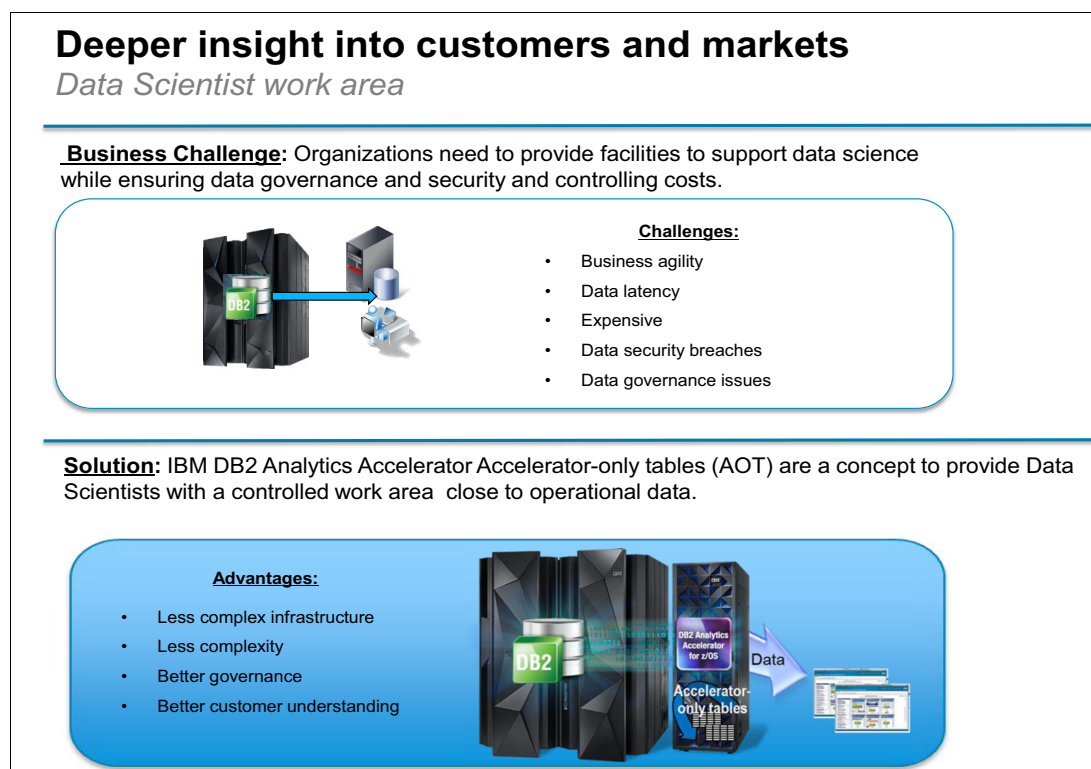


Figure 3-5 New usage scenario, which can be implemented with DB2 Analytics Accelerator version 5.1

IBM DB2 Analytics Accelerator provides significant capacity and performance to support data scientists' iterative processing without extra cost and, most importantly, without affecting production work and production response time. The new accelerator-only tables provide a possibility for a data scientist to work on DB2 for z/OS data, create transformations and aggregations on their own without requiring that the data be passed through DB2 for z/OS first and without a need to copy data throughout the organization.

Therefore the new accelerator-only tables provide a data scientist a faster and deeper insight into operational data with less infrastructure, less complexity, and better data governance.

Furthermore, DB2 Analytics Accelerator version 5.1 delivers new functionality called *in-database analytics*, which enables you to do modeling and scoring based on operational data much more efficiently.

Predictive analytics helps connect data to effective action by drawing reliable conclusions about current conditions and future events. It enables organizations to make predictions and then proactively act upon that insight to drive better business outcomes and achieve measurable competitive advantage.

With today's technology, a data scientist would typically access specifically prepared data in a data mart to perform data analysis and modeling. The data mart is generated from the enterprise data warehouse containing a subset of the data relevant to the analytical task.

This approach often limits the data analyst to a subset of the available data that was selected and decided in the past to be potentially relevant. Due to this approach, the data is not up-to-date or complete. It would be beneficial for the quality of the predictive models if the data analyst would have ad hoc access to all relevant operational data.

Another problem is that the resulting models rely on input data according to the data model in the data mart. This data model must be kept in sync with the operational model, to do adequate scoring (apply the model) in transaction processing.

A third problem is the freshness of the model used for scoring. If a fraud behavior or buying pattern changes, the sooner that change is propagated to the scoring model, the faster business benefits could be realized for the company.

Using the DB2 for z/OS scoring adapter for SPSS enables companies to optimizing decisions at the point of impact, where the transactions happen.

With in-database analytics processed within the Accelerator, the modeling process is as close as possible to operational/transactional data. It is there, where the data is created (through the operational process) and is there where the scoring model is needed to make scoring most efficient for both the using customer and the providing company.

3.5 Reduce IT sprawl for analytics initiatives

Today, analytics have a direct effect on business performance, and are critical for improved decision making, responding to business situations faster, and improving competitive advantage. As the need to support more data and growing numbers of analytics users, infrastructure sprawl can become an inhibitor as IT organizations strive to support new requirements for business critical analytics that demand the highest qualities of service.

Figure 3-6 shows an overview of the *reduced IT sprawl for analytics initiatives* use case scenario, which has been implemented with DB2 Analytics Accelerator.

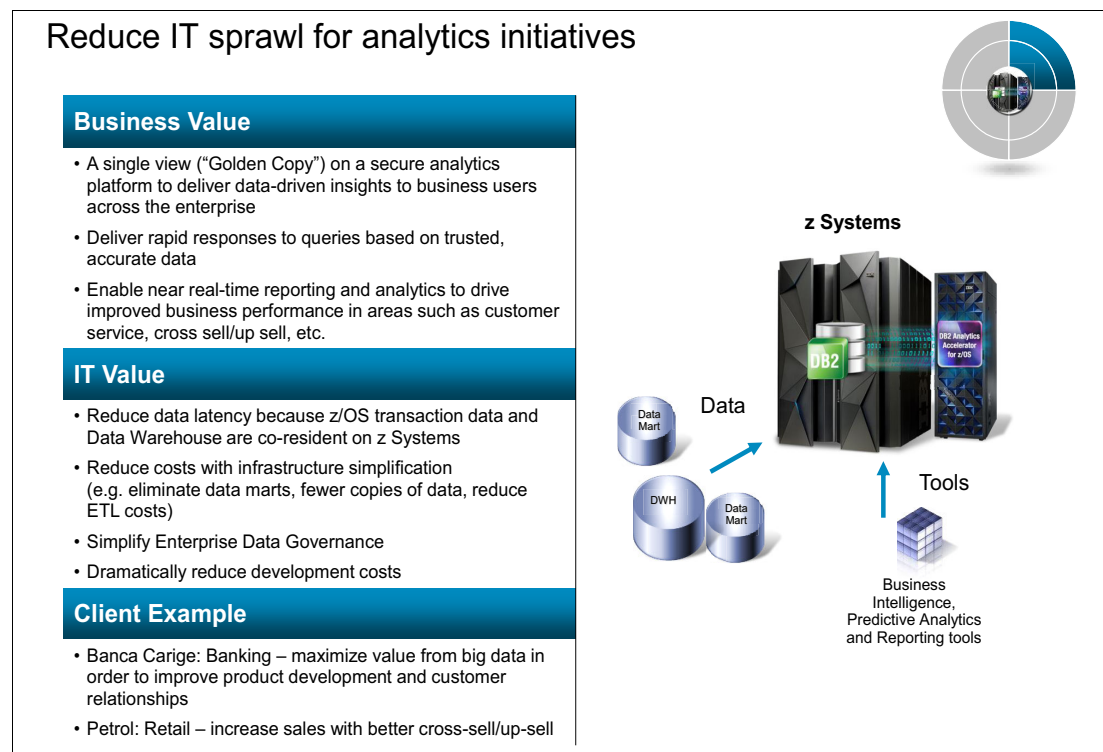


Figure 3-6 Reduce IT sprawl usage scenario

With traditional analytics approaches, organizations might duplicate and move data from System z to distributed departmental systems for analytic processing. Transferring data from one platform to another, however, can introduce data quality and security issues while adding data latency. Moreover, departmental analytics environments often struggle to deliver the availability, scalability, and performance levels required.

Organizations with most of their transactional data originating and housed on z Systems are rethinking their analytics approach, and are bringing together their transactional processing, data warehousing, and analytic tooling on System z. Implementing DB2 for z/OS and DB2 Analytics Accelerator in an IBM z Systems environment can help to meet and exceed business-critical analytics objectives and avoid the drawbacks of traditional analytics approaches.

What the DB2 Analytics Accelerator already delivers

DB2 for z/OS, DB2 Analytics Accelerator, and z Systems offer a hybrid solution that brings together high-volume business transactions, batch reporting, and complex analytic queries running concurrently in a mixed-workload environment. This new approach helps organizations to align their business operations and technology using a single, common platform to drive out complexity and cost while increasing flexibility to support business-critical analytics.

They can use their existing System z infrastructure, people, and processes to deliver their business-critical analytics across the enterprise with superior System z qualities of service: Reliability, availability, and scalability. This hybrid solution provides a single view on a secure analytics platform to deliver data-driven insights to business users across the enterprise, so decisions are made based on easy to find, trusted, and accurate data all driving improved business performance.

What is new with accelerator-only tables

In addition to the possibilities to reduce complexity, DB2 Analytics Accelerator Version 5.1 also provides an efficient, fast, and simple new way to implement data-transformation processes.

Many organizations use much valuable time transforming data to make it accessible for users. Processing that data more quickly means that it is available and accessible faster, helping organizations to more quickly close books, validate information, react to market changes, improve data governance, and build and maintain confidence in the underlying data.

Extract, transform, and load (ETL) aggregates and formats the data for highly efficient and fast user access. It integrates data from multiple sources, making the data usable by system users, and consistent across systems and time. Minimizing the time required to process data accelerates access to information by eliminating or minimizing data latency. Many times SQL is used to transform the data. ETL is essential with most of large enterprises, who have complex business models and more than one processing system.

Ideally, one would have a single environment from which all data would be perfect and drawn as needed, and with the performance required. Reality is somewhat different. Even within organizations where there is a single system, data might need to be transformed to make it consistent across the organization, and format it to be accessible by users. Therefore, ETL is a standard component of an information delivery architecture.

However, within some enterprises, data takes days to move from the operational platform, like DB2 for z/OS to their warehouse, mostly hosted on distributed platform, to be then accessible by the LOB users. The volumes are so large that they needed to transform terabytes of data.

ETL processing is often one of the major sources of latency and potential bottlenecks associated with moving data from operational processes to data warehouses and data marts. ETL is getting more complex, inefficient, and labor-intensive.

Figure 3-7 shows an overview of the data-transformation process usage scenario, which can be implemented with DB2 Analytics Accelerator Version 5.1.

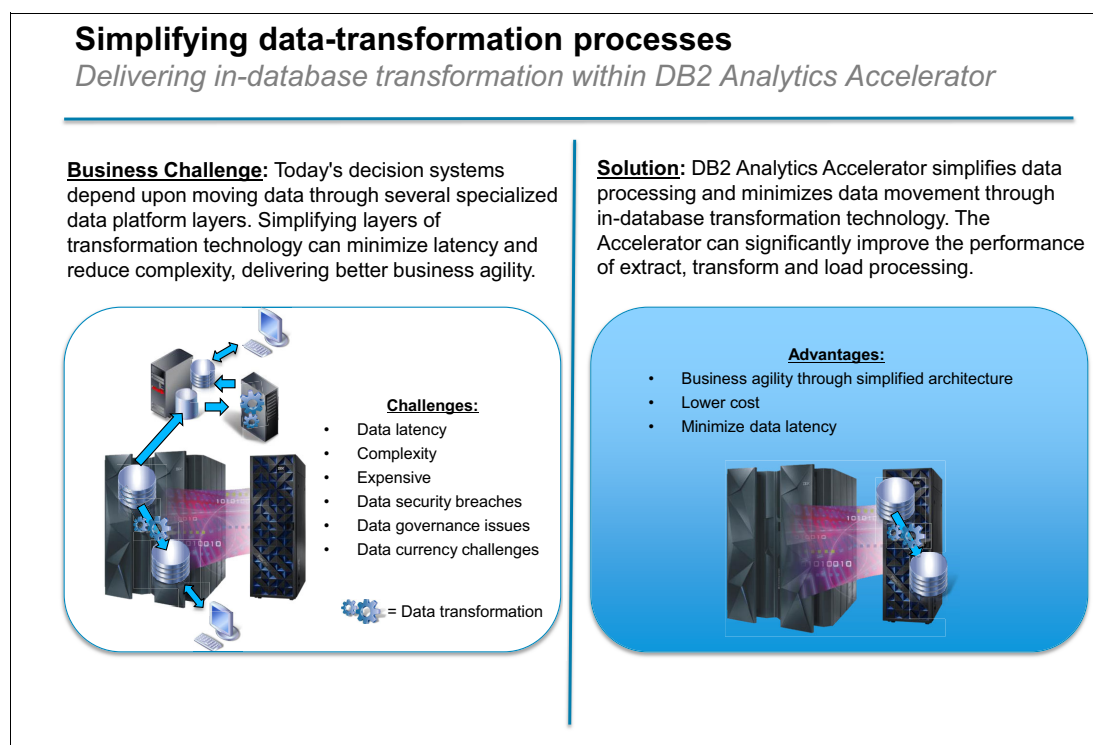


Figure 3-7 Simplifying data transformation processes with accelerator-only tables

DB2 Analytics Accelerator Version 5.1, enables IT organizations to simplify decision support architectures by reducing the capacity and processing steps needed for ETL, even potentially completely eliminating data layers, by a concept called *in-database transformation*. By moving ETL processes into the database engine, the Accelerator can dramatically improve processing and reduce costs.

Simplifying layers of ETL processing enables IT organizations to better meet SLA requirements, be more responsive to LOB requirements, while also reducing cost and resources needed. With this assistance, the IT organization helps the overall enterprise to increase business agility.

The concept of in-database transformation could be used by ETL tools like IBM InfoSphere DataStage, but also to accelerate custom ETL processes built on SQL functionality.

With DB2 Analytics Accelerator Version 5.1, it is possible to eliminate the need for data marts through the flexible data infrastructure of the accelerator-only table, and increase the responsiveness of the IT organization.

With traditional analytics approaches, organizations duplicate and move data from z Systems to distributed departmental systems for analytic processing. Transferring data from one platform to the next can introduce data quality and security issues while adding data latency. Moreover, departmental analytics environments often struggle to deliver the availability, scalability, and performance levels required.

Figure 3-8 shows an overview of a new usage scenario for data mart consolidation through flexible data infrastructure, which can be implemented with DB2 Analytics Accelerator V5.1.

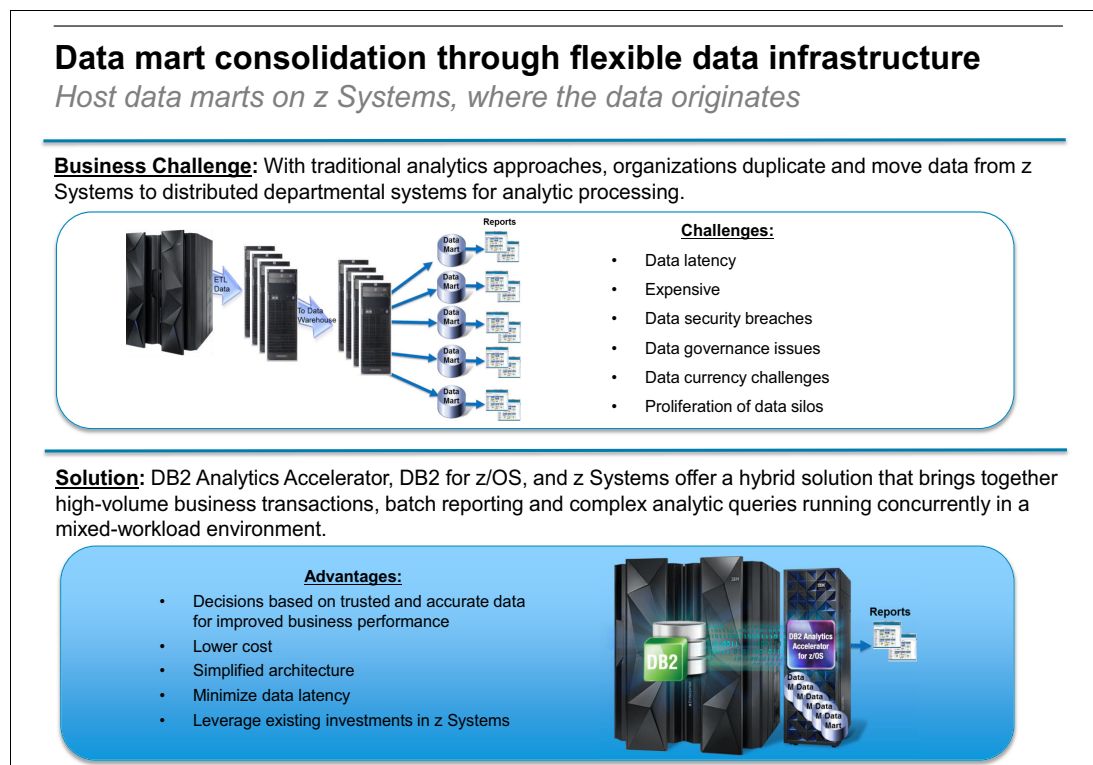


Figure 3-8 Data mart consolidation through flexible data infrastructure with DB2 Analytics Accelerator Version 5.1

The concept of accelerator-only tables is a powerful method to eliminate the need for such extra data marts. By just creating a set of new tables that exist on the Accelerator only, and sourcing the data from accelerator-shadow tables, it is possible for the IT organization to react quickly and efficiently to the changing data needs of the LOB users.

Therefore, it is now feasible to consolidate many of these data marts onto the Accelerator, close to where the data originates. This approach provides the cost-effectiveness and rapid deployment functionality that organizations can use for delivering insights in a timely manner to the correct people in the business.

3.6 Improve access to historical data and lower storage costs

As the use of analytics becomes more pervasive, organizations strive to make more data accessible to users to drive better decision-making, which can improve customer satisfaction, managerial oversight and support regulatory requirements. Organizations are challenged to balance the need for keeping large volumes of data available for analytic analysis with managing costs and keeping systems running optimally.

Figure 3-9 shows an overview of a usage scenario about improving access to historical data and lowering storage costs, which can be implemented with DB2 Analytics Accelerator.

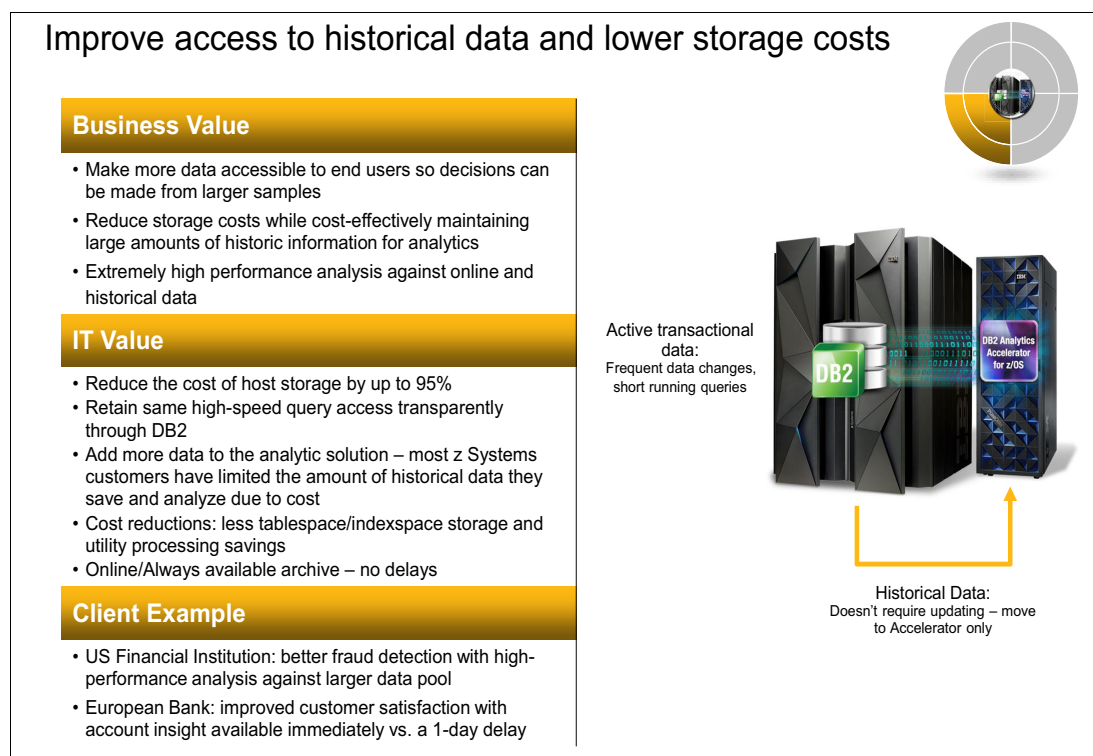


Figure 3-9 Improve access to historical data and lower storage costs with DB2 Analytics Accelerator

Most analytical systems are based on data that is 95% historical and therefore static. A retailer, for instance, might maintain seven years of past sales histories that contain every transaction for every product sold to every customer. Because this data is historical, it generally is not subject to revision or updates.

What the DB2 Analytics Accelerator already delivers

The High Performance Storage Saver (HPSS) feature of DB2 Analytics Accelerator helps reduce the cost of storing, managing, and processing this type of data. Organizations can move static data within tables or table partitions to the DB2 Analytics Accelerator, and remove the data from DB2 for z/OS. All of the data is still managed by and safeguarded in the DB2 directory, and all of the queries that target that data are now directed only to the DB2 Analytics Accelerator.

This process can dramatically reduce storage costs on z Systems, and enable organizations to substantially increase the amount of history maintained for each subject area so that decisions can be made from much larger data samples. Another great advantage of the HPSS feature is the capability of extremely high-performance analysis against both online and historical data, transparently to users and applications.

What is new with DB2 Analytics Accelerator version 5.1

With DB2 Analytics Accelerator version 5.1 and the DB2 Analytics Accelerator version 5.1 Loader product, it is possible to load a predefined set of non-DB2 for z/OS data, currently limited to IBM IMS data only, into the Accelerator.

Figure 3-10 shows an overview of data loading usage scenario, which can be implemented with DB2 Analytics Accelerator version 5.1.

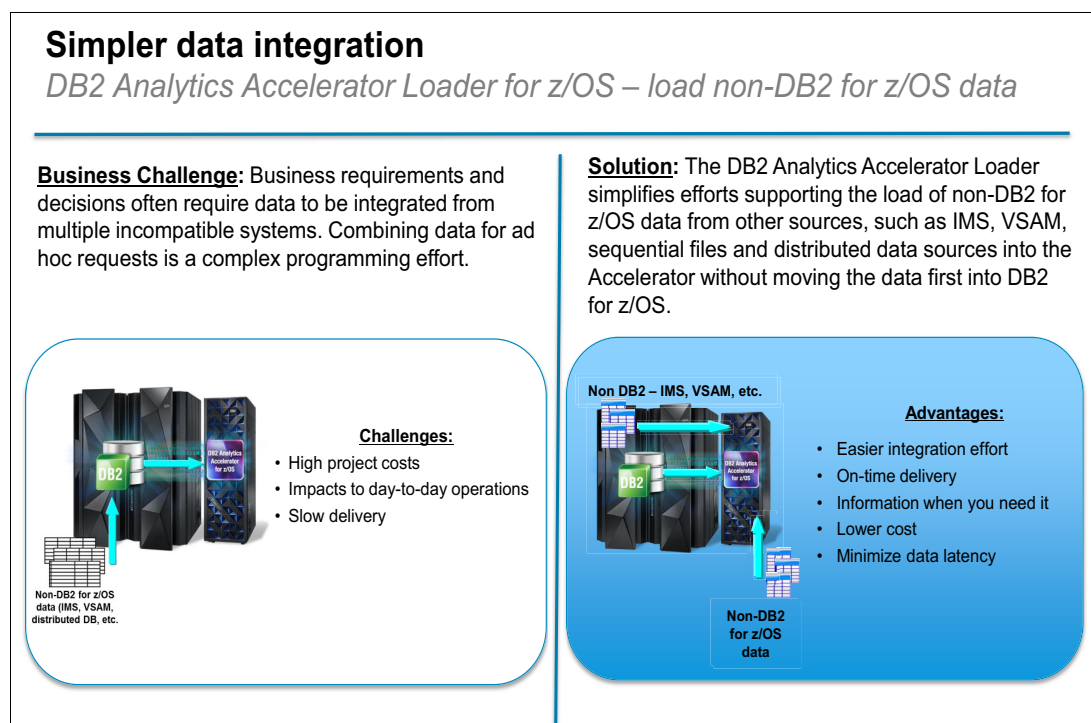


Figure 3-10 Overview of a data loading usage scenario with DB2 Analytics Accelerator Version 5.1

Large enterprises have many applications, systems, and sources of data, some of which are used to fulfill specific business functions. Often, it is necessary and beneficial to bring these “islands” of information together to reveal a complete and accurate picture, ultimately getting closer to the truth by considering multiple perspectives.

Business requirements and decisions often require data to be integrated from multiple incompatible systems. Ad hoc requests become a complex programming effort. This drives up project costs and negatively impacts day-to-day business operations. These efforts are typically slow to deliver, and often require several iterations. Changes require additional programming effort, which can take weeks or even months.

Many IBM z Systems customers have data in IMS, Virtual Storage Access Method (VSAM), and distributed databases. To make decisions and support regulatory requirements, they need to combine disparate data. Every day, such data is copied into warehouses for analytics. Directly loading this data into the DB2 Analytics Accelerator, using the DB2 Analytics Accelerator Loader, reduces the time and resources spent daily to move this data to other platforms. This reduces latency and improves security and data governance.

The DB2 Analytics Accelerator Loader simplifies efforts supporting the load of non-DB2 for z/OS data from other sources, such as IMS, VSAM, sequential files, and distributed data sources into the Accelerator without moving the data first into DB2 for z/OS.

3.7 Summary

The Accelerator supports the full lifecycle of a real-time analytics solution on a single, integrated system combining transactional data, historical data, and predictive analytics. The accelerator-only tables and the new DB2 Analytics Accelerator version 5.1 extends the four use case categories with important additional usage scenarios.

Figure 3-11 shows how the four major use cases are extended by additional usage scenarios with DB2 Analytics Accelerator Version 5.1.

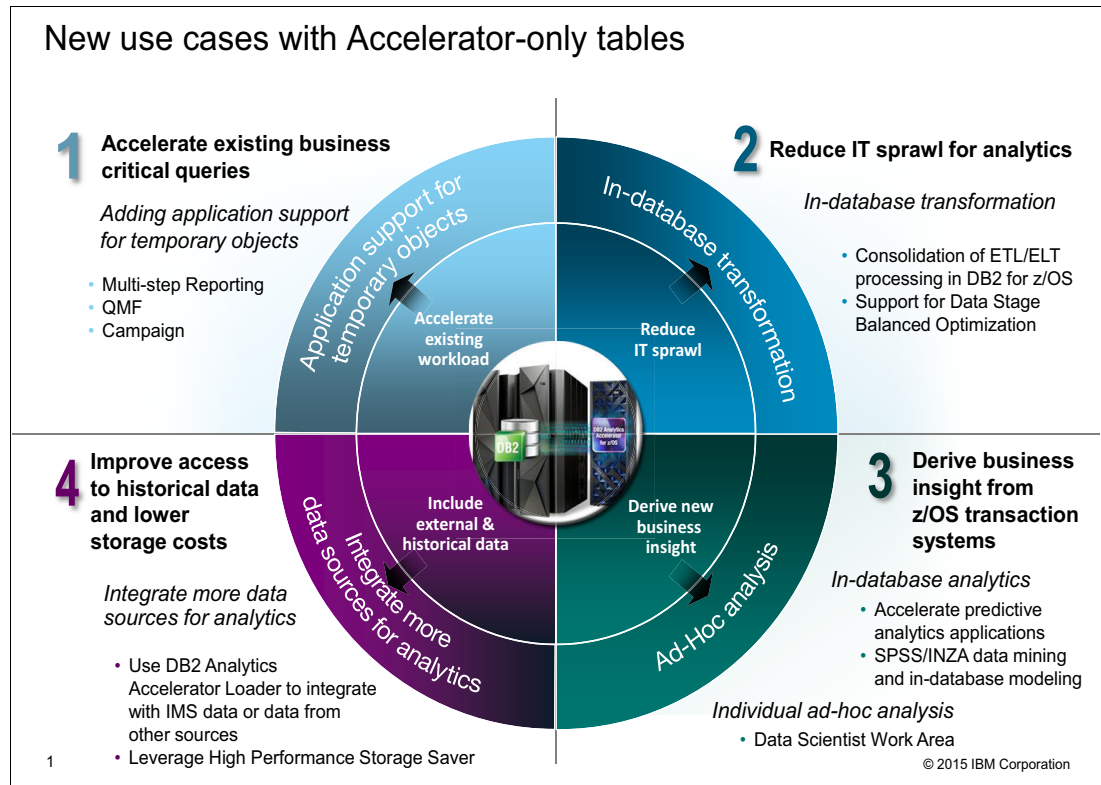


Figure 3-11 Four use cases with DB2 Analytics Accelerator Version 5.1



Multistep reporting

This chapter describes how DB2 Analytics Accelerator expands its scope from pure query acceleration to data manipulation on the accelerator. We describe how existing software products that perform multistep reporting can gain performance improvements by using accelerator-only tables.

Microstrategy Business Intelligent Reporting is one example of a product that performs multistep reporting and can use accelerator-only tables. Another example is IBM DB2 Query Management Facility (QMF) for z/OS. QMF for z/OS and its accelerator-only tables support are described in detail in Chapter 5, “Using IBM DB2 QMF to store query results and import tables” on page 53.

This chapter covers the following topic:

- Concepts of multistep reporting

4.1 Concepts of multistep reporting

There is a wide range of reporting and analytical tools available that enable users to access and query data in DB2 for z/OS. Setting values for special registers, such as CURRENT QUERY ACCELERATION, within available tools can be referred to as “business as usual”, so that most reporting and analytical tools can successfully send queries to an accelerator.

However, some of these tools run queries against DB2 for z/OS that are run on an accelerator, and they also follow a multistep reporting approach. A multistep reporting approach can be divided into several 1 to n steps, where first steps reduce initial large data volumes to smaller extracts of relevant data that can be handled more efficiently in following steps.

You can find an example of a multistep report using extra tables in Example 4-1.

Example 4-1 Multistep reporting flow

```
CREATE TABLE STEP1
(N_NATIONKEY INTEGER,
 WJXBFS1      DOUBLE)

INSERT INTO STEP1
(SELECT T3.C_NATIONKEY      N_NATIONKEY
      ,SUM(T1.L_EXTENDEDPRICE WJXBFS1
      FROM LINEITEM T1
      JOIN ORDERS  T2
      ON T1.L_ORDERKEY = T2.O_ORDERKEY
      JOIN CUSTOMER T3
      ON T2.O_CUSTKEY = T3.C_CUSTKEY
      GROUP BY T3.C_NATIONKEY)

CREATE TABLE STEP2
(N_NATIONKEY INTEGER,
 WJXBFS1      DOUBLE)

INSERT INTO STEP2
(SELECT A12.C_NATIONKEY      N_NATIONKEY,
      ,SUM(T1.O_TOTALPRICE WJXBFS1
      FROM ORDERS  T1
      JOIN CUSTOMER T2
      ON T1.O_CUSTKEY = T2.C_CUSTKEY
      GROUP BY T2.C_NATIONKEY)

SELECT DISTINCT T3.N_REGIONKEY R_REGIONKEY,
                T4.R_NAME      R_NAME,
                T1.N_NATIONKEY  N_NATIONKEY,
                T3.N_NAME      N_NAME,
                T1.WJXBFS1      WJXBFS1,
                T2.WJXBFS2      WJXBFS2
FROM STEP1 T1
JOIN STEP2 T2
ON T1.N_NATIONKEY = T2.N_NATIONKEY
JOIN NATION T3
ON T1.N_NATIONKEY = T3.N_NATIONKEY
```

```
JOIN REGION T4  
ON T3.N_REGIONKEY = T4.R_REGIONKEY
```

```
DROP TABLE STEP1
```

```
DROP TABLE STEP2
```

In Example 4-1 on page 48, we access tables within a star-schema. In this star-schema, LINEITEM represents the fact table, and other tables accessed are dimensional tables. Rather than running a more complex Structured Query Language (SQL) statement to join a huge fact table together with dimension tables, the task is split into multiple smaller statements. Technically, an **INSERT FROM SELECT** statement is run, selecting data from a source table and inserting a smaller result set into an intermediate target table.

Table types used for intermediate results can be exchanged: We have used standard tables; standard tables and declared global temporary tables (DGTTs) can be used interchangeably. A tool using multistep reporting can be configured to use either standard tables in DB2 for z/OS, declared temporary tables, or created temporary tables.

A prominent example of such tooling is MicroStrategy. A MicroStrategy report can consist of multiple steps, following the previously outlined algorithm to reduce data volumes as the report is run. By default, MicroStrategy uses DGTTs in DB2 for z/OS to store these intermediate results. After a report is completed, data is automatically removed from all DGTTs at the time that a reporting transaction completes.

Without accelerator-only tables, only the first **INSERT FROM SELECT** statement can be run on an accelerator, where data is selected from an accelerator and inserted into a table existing in DB2 for z/OS only.

The concept of accelerator-only tables introduces the capability for reporting software products, such as Microstrategy, to insert data on the accelerator, accelerate this process, and also accelerate follow-on processes that rely on data inserted into an accelerator-only table.

Figure 4-1 shows the architecture of a multistep reporting process that leverages accelerator-only tables. Intermediate results are saved on the accelerator in accelerator-only tables, and only the final result is returned back to DB2 for z/OS and the reporting application.

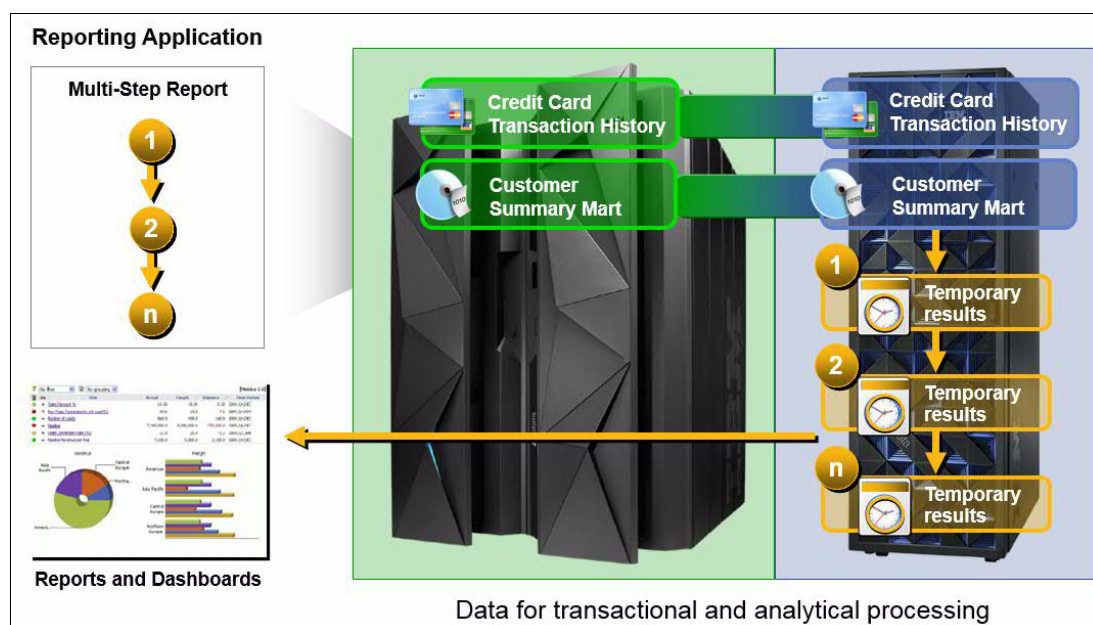


Figure 4-1 Multistep reporting with accelerator-only tables

Example 4-2 shows the multistep reporting flow of Example 4-1 on page 48, this time using accelerator-only tables.

Example 4-2 Multi-step reporting flow with accelerator-only tables

```
SET CURRENT QUERY ACCELERATION ELIGIBLE;
CREATE TABLE STEP1
  (N_NATIONKEY INTEGER,
   WJXBFS1      DOUBLE) IN ACCELERATOR A1 IN DATABASE D1;
COMMIT;

INSERT INTO STEP1
  (SELECT T3.C_NATIONKEY      N_NATIONKEY
    ,SUM(T1.L_EXTENDEDPRICE WJXBFS1
    FROM LINEITEM T1
    JOIN ORDERS   T2
      ON T1.L_ORDERKEY = T2.O_ORDERKEY
    JOIN CUSTOMER T3
      ON T2.O_CUSTKEY = T3.C_CUSTKEY
    GROUP BY T3.C_NATIONKEY);

COMMIT;

CREATE TABLE STEP2
  (N_NATIONKEY INTEGER,
   WJXBFS1      DOUBLE) IN ACCELERATOR A1 IN DATABASE D1;

COMMIT;

INSERT INTO STEP2
```

```


        (SELECT A12.C_NATIONKEY      N_NATIONKEY,
               ,SUM(T1.O_TOTALPRICE WJXBFS1
        FROM ORDERS  T1
        JOIN CUSTOMER T2
          ON T1.O_CUSTKEY = T2.C_CUSTKEY
        GROUP BY T2.C_NATIONKEY);

SELECT DISTINCT T3.N_REGIONKEY R_REGIONKEY,
               T4.R_NAME      R_NAME,
               T1.N_NATIONKEY  N_NATIONKEY,
               T3.N_NAME      N_NAME,
               T1.WJXBFS1      WJXBFS1,
               T2.WJXBFS2      WJXBFS2
FROM STEP1 T1
JOIN STEP2 T2
  ON T1.N_NATIONKEY = T2.N_NATIONKEY
JOIN NATION T3
  ON T1.N_NATIONKEY = T3.N_NATIONKEY
JOIN REGION T4
  ON T3.N_REGIONKEY = T4.R_REGIONKEY;

DROP TABLE STEP1;

DROP TABLE STEP2;

```



Using IBM DB2 QMF to store query results and import tables

This chapter introduces IBM DB2 Query Management Facility (QMF) for z/OS, and describes its support for DB2 Analytics Accelerator.

A new QMF for z/OS release, V11.2, is available since the 4th of September 2015. This new release provides enhancements to specific QMF commands to support accelerator-only tables.

This chapter describes multiple examples that use the new command enhancements to benefit from using accelerator-only tables and query acceleration. One example is to use the new command enhancements to perform multistep reporting on the accelerator.

This chapter covers the following topics:

- ▶ QMF and IBM z Systems
- ▶ QMF for z/OS and DB2 Analytics Accelerator
- ▶ Running queries and saving results using a QMF procedure
- ▶ Importing tables as accelerator-only tables
- ▶ Preferred practices

5.1 QMF and IBM z Systems

Historically, QMF was created to provide an easy, powerful query and reporting tool for DB2 for z/OS. Since its first release many years ago, QMF has been dramatically enhanced. It has evolved into a solution that offers industry-leading advantages in heterogeneous data access, presentation, and analytics across a wide variety of platforms, databases, browsers, and mobile devices.

QMF consists of the following clients:

- ▶ IBM Time Sharing Option (TSO) and IBM Customer Information Control System (IBM CICS®) client
Provides query, reporting, charting, procedure, and application development functions for users who work entirely from 3270 terminal emulators.
- ▶ Workstation client
Offers a rich desktop client development environment on distributed platforms for all QMF queries and reports. It can access all existing objects developed in the TSO and CICS environment. Runs on Microsoft Windows, Linux, and Mac OS.
- ▶ Web and mobile clients
Provides a feature set similar to that of workstation client, using a thin client, browser-based solution, and mobile device support. Runs on Windows, Linux, Solaris, IBM AIX®, HP-UX, iSeries, Linux for System z, and z/OS.

In this book, we do not cover the capabilities of the QMF clients in detail. To learn more about them, see *Complete Analytics with IBM DB2 Query Management Facility*, SG24-8012:

<http://www.redbooks.ibm.com/abstracts/sg248012.html>

QMF is available in the following ways:

- ▶ It is available as a priced optional monthly license charge (MLC) feature of the DB2 for z/OS product. There are two QMF features:
 - QMF Classic Edition 11
Offers QMF for the TSO and CICS environments. It is available for DB2 10 or 11 of DB2 for z/OS.
 - QMF Enterprise Edition 11
It includes QMF Classic Edition 11, in addition to the web, workstation, and mobile clients. It is available for DB2 10 or 11 of DB2 for z/OS.
- ▶ It is available as a stand-alone, separately orderable product called QMF for z/OS. This separate product became generally available October 2014. Because it is a product and not a feature of DB2 for z/OS, it is not tied to the DB2 for z/OS release cycle. As a result, new releases of the QMF for z/OS product can be made available on a more frequent basis.

The QMF for z/OS product includes all the capabilities included in the QMF 11 Classic and Enterprise Editions. Additionally, QMF for z/OS has a new release 11.2, which became available on September 4th 2015.

Further QMF descriptions and details in this book belong to the QMF for z/OS product only. QMF for z/OS v11.2 introduces new functions that are exclusive to the QMF for z/OS product and are not available with the QMF 11 Classic or Enterprise features of DB2 for z/OS.

QMF for z/OS provides a zero-footprint, mobile-enabled, highly secure business analytics solution. QMF for z/OS delivers modern analytics by using visual reports and dashboards, available across all current interfaces, including mobile devices, web browsers, workstations, and IBM TSO and IBM CICS clients.

QMF for z/OS uses the power, security, and robustness of IBM z Systems to create analytical solutions that directly access a large spectrum of data sources, such as relational and hierarchical databases, big data sources (Hadoop and IBM BigInsights), and website data. QMF for z/OS empowers users at all levels to use data to help them find answers, make decisions, and communicate those decisions.

QMF for z/OS provides visual dashboards, reports and analytics, and access to all existing QMF reports through various interfaces. No migration is required. Users can continue to use their existing QMF reporting infrastructure while quickly and easily transforming tabular reports into clear, visual reports that can be understood without difficulty and distributed rapidly.

5.2 QMF for z/OS and DB2 Analytics Accelerator

QMF for z/OS works seamlessly with DB2 Analytics Accelerator. DB2 routes QMF queries that qualify for acceleration to DB2 Analytics Accelerator if the **CURRENT QUERY ACCELERATOR** special register is set to a value other than NONE. For QMF queries, the precedence (lowest to highest) order for setting the value of the special register is as follows:

- ▶ The **QUERY_ACCELERATION** subsystem parameter
 - ▶ An explicit **SET CURRENT QUERY ACCELERATION** statement within a QMF query object
- This is supported in QMF Classic and Enterprise Edition since Version 10 and in QMF for z/OS.

The new release QMF for z/OS V11.2 extends the support for DB2 Analytics Accelerator by providing the capability to save query results, or to import tables directly into accelerator-only tables rather than into DB2 tables.

In general, saving a query result to a table is useful for the following purposes:

- ▶ To save intermediate results temporarily as part of a multistep process, for example, multistep reporting. In general, intermediate results are deleted after the process has completed.
- ▶ To persist a query result for later processing. This could be the result of a single query, or the final result of a multistep process.

Saving a query result into an accelerator-only table rather than saving it into a DB2 table has the following advantages:

- ▶ Enables a multistep procedure, for example a procedure that runs queries on results of prior queries, to run completely on DB2 Analytics Accelerator.
- ▶ Saves disk space and processor (CPU) resource costs on z Systems currently used for storing results in DB2 tables and reading them again.
- ▶ Enables acceleration of queries that run on the saved results.

However, saving a query result into an accelerator-only table rather than a DB2 table might also have disadvantages:

- ▶ At the time of writing, no backup concept exists for accelerator-only tables. If backup is important, then save the result into a DB2 table. You can load this table to DB2 Analytics Accelerator in a subsequent step to enable query acceleration on this table.
- ▶ If multiple accelerators are connected to a DB2 subsystem, an accelerator-only table can only exist on one of these accelerators. If you want to make a query result available on multiple accelerators for high availability and disaster recovery (HA/DR) or workload balancing purposes, save the result into a DB2 table and afterward load the table to the accelerators.

5.2.1 Accelerator-only table support in QMF for z/OS V11.2

QMF for z/OS V11.2 introduces two new *global variables* that must be set to enable the support to save data or import tables to accelerator-only tables:

- ▶ DSQEC_SAV_ALLOWED

This variable controls whether users save data to a new table in the database or an accelerator. Data can be saved to an accelerator using the **QMF Save Data**, **Run Query to table**, or **Import Table** commands. Except for option 0, this field does not influence the location of existing tables that are being replaced or appended to. Existing tables are replaced or appended to in the database or accelerator regardless of this option. Possible values are shown in the following list. Values 2, 3, or 4 must be used to save data to an accelerator:

- 0 - Disable Save Data

Specifies that users cannot issue the **QMF Save Data**, **Import Table**, or **Run Query** (with **TABLE** keyword) commands to save data to a table in the database or accelerator.

- 1 - Enable Save Data to database tables only

Specifies that users can save data to a table in the database using the **QMF Save Data**, **Import Table**, or **Run Query** (with **TABLE**) commands. Users cannot save data to accelerator-only tables. *This is the default.*

- 2 - Enable Save Data to accelerator-only tables only

Specifies that users can save data to an accelerator-only table using the **QMF Save Data**, **Import Table**, or **Run Query** (with **TABLE**) commands. Users cannot save data to database tables. The `DSQEC_SAV_ACCELNM` global variable contains the default name of the accelerator, but can be overridden with the **ACCELERATOR** keyword.

- 3 - Enable Save Data to either database or accelerator-only tables (database default)

Specifies that users can save data to either a table in the database or an accelerator-only table using the **QMF Save Data**, **Import Table**, or **Run Query** (with **TABLE**) commands. If no command keyword overrides, such as **SPACE** or **ACCELERATOR**, are present, tables are saved in the database.

- 4 - Enable Save Data to either database or accelerator-only tables (accelerator default)

Specifies that users can save data to either a table in the database or an accelerator-only table using the **QMF Save Data**, **Import Table**, or **Run Query** (with **TABLE**) commands. If no command keyword overrides, such as **SPACE** or **ACCELERATOR**, are present, tables are saved in the accelerator. When this option is chosen, `DSQEC_SAV_ACCELNM` must contain the name of the accelerator.

► DSQEC_SAV_ACCELNM

This variable contains the default name of the accelerator to be used when creating accelerator-only tables from QMF **Save Data**, **Import Table**, or **Run Query** (with **TABLE**) commands. This variable is only referenced if the **ACCELERATOR** keyword is not specified.

In addition to the new global variables, QMF for z/OS V11.2 introduces new syntax enhancements to the **SAVE DATA**, **RUN QUERY**, and **IMPORT TABLE** commands, as described in Table 5-1.

Table 5-1 Syntax enhancements to QMF commands

Command with syntax enhancement	Description
SAVE DATA AS <i>tablename</i> (ACCELERATOR <i>accelname</i>)	Saves data as accelerator-only table <i>tablename</i> in accelerator <i>accelname</i> .
SAVE DATA AS <i>tablename</i> (SPACE <i>name</i>)	Saves data as database table <i>tablename</i> in database and table space specified by <i>name</i> .
IMPORT TABLE <i>tablename</i> FROM <i>datasetOrFile</i> (ACCELERATOR <i>accelname</i>)	Imports table data into accelerator-only table <i>tablename</i> in accelerator <i>accelname</i> .
IMPORT TABLE <i>tablename</i> FROM <i>datasetOrFile</i> (SPACE <i>name</i>)	Imports table data to database table <i>tablename</i> in database and table space specified by <i>name</i> .
RUN QUERY <i>qname</i> (TABLE <i>tablename</i> ACCELERATOR <i>accelname</i>)	Runs a query and saves the result directly into accelerator-only table <i>tablename</i> in accelerator <i>accelname</i> .
RUN QUERY <i>qname</i> (TABLE <i>tablename</i> SPACE <i>name</i>)	Runs a query and saves the result directly into database table <i>tablename</i> in database and table space specified by <i>name</i> .

For all commands described in Table 5-1, the value of the global variable DSQEC_SAV_ALLOWED has the following effect:

- If DSQEC_SAV_ALLOWED=1 and the user enters a command and the **ACCELERATOR** keyword is specified, an error occurs.
- If DSQEC_SAV_ALLOWED=2 and the user enters a command and the **SPACE** keyword is specified, an error occurs.
- If DSQEC_SAV_ALLOWED=2 and the user does not enter the **ACCELERATOR** keyword, the value in the global variable DSQEC_SAV_ACCELNM is used. If both are blank, an error occurs.
- If DSQEC_SAV_ALLOWED=3 and the user does not enter the **SPACE** or **ACCELERATOR** keyword, the value of the **SPACE** from the QMF profile is used.
- If DSQEC_SAV_ALLOWED=4 and the user does not enter the **SPACE** or **ACCELERATOR** keyword, the value in the global variable DSQEC_SAV_ACCELNM is used. If both are blank, an error occurs. The user must specify the **SPACE** parameter to override to write to accelerator-only tables.

For more details about global variables, commands, or other information about QMF for z/OS V11.2, see the QMF IBM Knowledge Center:

http://www.ibm.com/support/knowledgecenter/SS9UMF_11.2.0/

5.3 Running queries and saving results using a QMF procedure

A procedure is a QMF object that enables you to issue multiple commands with a single **RUN** command. For example, you can issue commands to run a query, save a query result to a table or print a report.

In this chapter, we outline a sample scenario that uses a procedure to run queries on results of prior queries of the same procedure. The result of a query is saved to a table, which the next query uses to aggregate the data further and save it again in another table.

This scenario serves multiple purposes of saving a query result to a table. When the procedure runs, query results are considered as intermediate results because they are used as input to the next query. The queries create meaningful results that answer specific business questions are therefore persisted.

Multiple variants of the procedure are described that save query results, without or with using accelerator-only tables:

- ▶ Running the procedure on DB2 only
- ▶ Running the procedure on DB2 and the accelerator
- ▶ Running the procedure on the accelerator using the **SAVE DATA AS** command
- ▶ Running the procedure on the accelerator using the **RUN QUERY** command

We use the TSO client of QMF for z/OS to run the scenario.

5.3.1 Queries used in the sample scenario

We use the TPC-H benchmark tables as base tables for the scenario.

In QMF for z/OS, we create three query objects, each one contains one SQL query.

The first query, **LINEPR**, as shown in Example 5-1, runs on multiple base tables. The query is a product line profit measure query, and it determines how much profit is made on a given line of parts, broken out by supplier, nation, and year. The query finds the profit for each part of a specified product line, each nation, and each year.

In our example, we have parts that belong either to a *green* or a *red* product line. Therefore, the query contains the QMF variable &COLOUR that is substituted during run time to get the results either for the green or red product line.

The **LINEPR** query is a slightly adapted version of query **Q9** of the published TPC-H queries:

<http://www.tpc.org/tpch/spec/tpch2.1.0.pdf>

Example 5-1 Query LINEPR: Product line parts profit measure per part, nation, and year

```
select
  parts,
  nation,
  year,
  sum(amount) as sum_profit
from (
  select
    p_name as parts,
    n_name as nation,
    extract(year from o_orderdate) as year,
    l_extendedprice * (1 - l_discount) - ps_supplycost * l_quantity as amount
```

```

from
    TPC19.part, TPC19.supplier, TPC19.lineitem, TPC19.partsupp,
    TPC19.orders, TPC19.nation
where
    s_suppkey = l_suppkey
    and ps_suppkey = l_suppkey
    and ps_partkey = l_partkey
    and p_partkey = l_partkey
    and o_orderkey = l_orderkey
    and s_nationkey = n_nationkey
    and p_name like &COLOUR
) as profit
group by
    parts,
    nation,
    year
order by
    nation, year desc;

```

Figure 5-1 shows an extract of the output of query **LINEPR** when the variable &COLOUR was set to %green%.

PARTS	NATION	YEAR	SUM PROFIT
beige navy dim green forest	ALGERIA	1992	61475.5592
plum lime forest linen green	ALGERIA	1992	1305.8591
sky deep coral green tomato	ALGERIA	1992	8270.1344
floral seashell burlywood green chiffon	ALGERIA	1992	18121.9007
dodger green puff thistle light	ALGERIA	1992	69928.8907
green peach olive deep royal	ARGENTINA	1998	3302.2320
antique red moccasin green burlywood	ARGENTINA	1998	21537.1580
lawn beige deep green dim	ARGENTINA	1998	15970.8340
tomato green antique magenta navajo	ARGENTINA	1998	18233.4900
light seashell firebrick cornflower green	ARGENTINA	1998	5618.1720
cornsilk green seashell goldenrod metallic	ARGENTINA	1998	19126.9728
aquamarine forest powder peru green	ARGENTINA	1998	39686.4083
dark green antique puff wheat	ARGENTINA	1998	12983.0800
cornsilk pink royal frosted green	ARGENTINA	1997	34849.5444
antique red moccasin green burlywood	ARGENTINA	1997	129703.7080

Figure 5-1 Extract of the output of query **LINEPR** to measure profit per part, nation, and year

The second query, **SUMPR**, as shown in Example 5-2, uses the saved output of query **LINEPR** as input and aggregates the data further. It summarizes for each nation and each year the profit for all parts of that specific product line to analyze which product line makes how much profit in which year and in which nation. The query contains the variable &TABNAME that is substituted during run time with the name of the saved table that contains the output of query **LINEPR** for either product line green or red.

Example 5-2 Query **SUMPR**: Product line profit measure query per nation and year

```

Select nation, year, sum(sum_profit) as sum_profit from &TABNAME
group by nation, year
order by nation, year

```

Figure 5-2 shows an extract of the output of query **SUMPR** when the variable &TABNAME was set to the table name that contained the output of query **LINEPR** for the green product line.

NATION	YEAR	SUM PROFIT
ALGERIA	1992	285188.4503
ALGERIA	1993	448371.4336
ALGERIA	1994	426573.4415
ALGERIA	1995	486706.0631
ALGERIA	1996	156938.7971
ALGERIA	1997	401601.1258
ALGERIA	1998	386617.8283
ARGENTINA	1992	312412.4307
ARGENTINA	1993	351669.5325
ARGENTINA	1994	374046.7888
ARGENTINA	1995	399583.4009
ARGENTINA	1996	325019.6263
ARGENTINA	1997	423877.6754
ARGENTINA	1998	136458.3471
BRAZIL	1992	280563.4508
BRAZIL	1993	457273.7693

Figure 5-2 Extract of the output of query SUMPR to measure profit per nation and year for the 'green' product line

The third query, **TOTALPR**, as shown in Example 5-3, uses the saved outputs of query **SUMPR** for product line green and red as input, and aggregates the data further. It summarizes the profits over both product lines per nation and year to figure out which nations made the most profit in which year. The query contains the QMF variables &T1 and &T2 that are substituted during run time with the saved output tables of query **SUMPR** for both product lines.

Example 5-3 Query TOTALPR: Total profit per nation and year

```

Select nation, year, sum(profit) as total_profit from (
  Select nation, year, sum_profit as profit from &T1
  UNION ALL
  Select nation, year, sum_profit as profit from &T2
) as partprofit
group by nation, year order by year, total_profit desc;

```

Figure 5-3 shows an extract of the output of query **TOTALPR**.

NATION	YEAR	TOTAL PROFIT
GERMANY	1992	804610.5229
UNITED KINGDOM	1992	742633.9921
PERU	1992	634127.2893
ALGERIA	1992	630267.6509
CANADA	1992	519014.3775
ETHIOPIA	1992	495061.0930
BRAZIL	1992	450016.2639
IRAQ	1992	396454.6120
IRAN	1992	318802.4042
MOROCCO	1992	298416.3769
FRANCE	1992	225676.1413
JORDAN	1992	166872.9499
SAUDI ARABIA	1992	64241.8543
MOZAMBIQUE	1993	2016409.4107
UNITED STATES	1993	1339769.3373
ROMANIA	1993	1197446.4356

Figure 5-3 Extract of the output of query **TOTALPR** to measure the total profit over both product lines per nation and year

5.3.2 Running the procedure in DB2 for z/OS

In QMF for z/OS, we implemented a procedure that runs all three queries for both product lines and saves the results to DB2 tables.

The procedure is shown in Example 5-4. The queries **LINEPR** and **SUMPR** (listed in Example 5-1 on page 58 and Example 5-2 on page 59) are run for both product lines, and therefore run twice. Variables within the queries are substituted by the specified values, for example the **&COLOUR** variable is substituted with the value **%green%** using the syntax **(&&COLOUR='%green%')**. The **SAVE DATA AS** command saves the result of the run query into the specified DB2 table. QMF for z/OS creates the table if it does not yet exist.

Example 5-4 QMF procedure to run queries in DB2 and save results to DB2 tables

```

RUN QUERY BMB.LINEPR (&&COLOUR='%green%'
SAVE DATA AS PRODLIN_PROFIT_GREEN

RUN QUERY BMB.LINEPR (&&COLOUR='%red%'
SAVE DATA AS PRODLIN_PROFIT_RED

RUN QUERY BMB.SUMPR (&&TABNAME=PRODLIN_PROFIT_GREEN
SAVE DATA AS SUMS_PROFIT_GREEN

RUN QUERY BMB.SUMPR (&&TABNAME=PRODLIN_PROFIT_RED
SAVE DATA AS SUMS_PROFIT_RED

RUN QUERY BMB.TOTALPR (&&T1=SUMS_PROFIT_GREEN &&T2=SUMS_PROFIT_RED
SAVE DATA AS TOTAL_PROFITS

```

5.3.3 Running the procedure in DB2 for z/OS and DB2 Analytics Accelerator

If you use DB2 Analytics Accelerator and QMF for z/OS V11.1, the procedure outlined in Example 5-4 can partially benefit from query acceleration. The query **LINEPR** (listed in Example 5-1 on page 58) runs on the TPC-H base tables. If these base tables are loaded to the accelerator as accelerator-shadow tables, and the **CURRENT QUERY ACCELERATION** special register is set to a value other than NONE (for example to the value ELIGIBLE), the **LINEPR** query is run on the accelerator.

Because QMF for z/OS V11.1 does not support saving a result to an accelerator-only-table yet, the **SAVE DATA AS** command saves the query result to a DB2 table. Therefore, the subsequent queries **SUMPR** and **TOTALPR** (listed in Example 5-2 on page 59 and Example 5-3 on page 60) that operate on the saved DB2 tables run in DB2.

QMF supports setting the **CURRENT QUERY ACCELERATION** as part of a QMF query. To accelerate the **LINEPR** query, edit the QMF query object and enter the **SET** statement in front of the query, as shown in Example 5-5.

Example 5-5 Setting a special register in a QMF query object

SET CURRENT QUERY ACCELERATION ELIGIBLE;

```
select
  parts,
  nation,
  year,
  sum(amount) as sum_profit
from (<subselect>) as profit
group by
  parts,
  nation,
  year
order by
  nation, year desc;
```

Note: In QMF for z/OS, the DSQEC_RUN_MQ global variable controls whether queries with multiple SQL statements are allowed. To run a query with multiple statements, such as the one in Example 5-5, ensure that each statement is separated by a Semicolon (;), then set the DSQEC_RUN_MQ global variable to 1 and run the query. When the variable is set to 0, all statements after the first semicolon are ignored.

Another option would be to create a dedicated QMF query object that only contains the **SET CURRENT QUERY ACCELERATION ELIGIBLE** statement. Adapt your QMF procedures and run this query as the first query in your procedures. The **CURRENT QUERY ACCELERATION** special register is then set for all queries in your procedure.

Tip: As an alternative to setting the **CURRENT QUERY ACCELERATION** with a QMF query, you can set the **QUERY_ACCELERATION DSNZPARM** to a value other than NONE to enable acceleration on the base tables. Or, if you use DB2 11, you can set the **CURRENT QUERY ACCELERATION** register through a profile. This way you do not have to adapt any QMF queries or procedures to benefit from query acceleration.

5.3.4 Running the procedure in DB2 Analytics Accelerator using accelerator-only tables created by the **SAVE DATA AS** command

If you use DB2 Analytics Accelerator and QMF or z/OS V11.2, the procedure described in Example 5-4 on page 61 can benefit from query acceleration and using accelerator-only tables.

In this chapter, we describe multiple possibilities about how to run the procedure using accelerator-only tables that are created using the **SAVE DATA AS** command. How you run it depends on the option that you choose for the QMF global variable `DSQEC_SAV_ALLOWED`. This variable is described in detail in chapter 5.2.1, “Accelerator-only table support in QMF for z/OS V11.2” on page 56.

If you do not want to change the procedure at all, you can set the QMF global variable `DSQEC_SAV_ALLOWED` to option 2 (Enable Save Data to accelerator-only tables only) or option 4 (Enable Save Data to either database or accelerator-only tables; default to accelerator-only tables). You can additionally set the QMF global variable `DSQEC_SAV_ACCELNM` to the name of your accelerator, then run the procedure as is. All **SAVE DATA** commands save the query results to accelerator-only tables on the accelerator specified in `DSQEC_SAV_ACCELNM`.

Using option 4 or using option 3 (Enable Save Data to either database tables or accelerator-only tables; default to database tables) enables you to save query results to either database tables or accelerator-only tables, within the same procedure or among multiple procedures.

For a multistep procedure, such as that we use in our sample scenario, it might be preferable to save the final result into a database table rather than an accelerator-only table. You might want to do this because you want to back up this table, or you want to make this table available to multiple accelerators for workload balancing or HA/DR purposes.

If you use option 4 for variable `DSQEC_SAV_ALLOWED`, the procedure must be changed, as shown in Example 5-6. You change the procedure to write the results of the first four queries to accelerator-only tables on the accelerator specified in `DSQEC_SAV_ACCELNM`, and to write the final result into a database table in table space `DSQDBDEF.DSQTSDDEF`.

Example 5-6 QMF procedure with `DSQEC_SAV_ALLOWED` set to option 4

```
RUN QUERY BMB.LINEPR (&&COLOUR='%green%'
SAVE DATA AS PRODLINE_PROFIT_GREEN_AOT

RUN QUERY BMB.LINEPR (&&COLOUR='%red%'
SAVE DATA AS PRODLINE_PROFIT_RED_AOT

RUN QUERY BMB.SUMPR (&&TABNAME=PRODLINE_PROFIT_GREEN_AOT
SAVE DATA AS SUMS_PROFIT_GREEN_AOT

RUN QUERY BMB.SUMPR (&&TABNAME=PRODLINE_PROFIT_RED_AOT
SAVE DATA AS SUMS_PROFIT_RED_AOT

RUN QUERY BMB.TOTALPR (&&T1=SUMS_PROFIT_GREEN_AOT &&T2=SUMS_PROFIT_RED_AOT
SAVE DATA AS TOTAL_PROFITS (SPACE DSQDBDEF.DSQTSDDEF
```

If you use option 3 for variable DSQEC_SAV_ALLOWED, the procedure must be changed, as shown in Example 5-7. You change it to write the results of the first four queries to accelerator-only tables on accelerator STRIPER, and write the final result into a database table.

Example 5-7 QMF procedure with DSQEC_SAV_ALLOWED set to option 3

```

RUN QUERY BMB.LINEPR (&&COLOUR='%green%'
SAVE DATA AS PRODLINE_PROFIT_GREEN_AOT (ACCELERATOR STRIPER

RUN QUERY BMB.LINEPR (&&COLOUR='%red%'
SAVE DATA AS PRODLINE_PROFIT_RED_AOT (ACCELERATOR STRIPER

RUN QUERY BMB.SUMPR (&&TABNAME=PRODLINE_PROFIT_GREEN_AOT
SAVE DATA AS SUMS_PROFIT_GREEN_AOT (ACCELERATOR STRIPER

RUN QUERY BMB.SUMPR (&&TABNAME=PRODLINE_PROFIT_RED_AOT
SAVE DATA AS SUMS_PROFIT_RED_AOT (ACCELERATOR STRIPER

RUN QUERY BMB.TOTALPR (&&T1=SUMS_PROFIT_GREEN_AOT &&T2=SUMS_PROFIT_RED_AOT
SAVE DATA AS TOTAL_PROFITS

```

Whether you choose option 3 or option 4 depends on how many existing procedures in your environment you need to change. The change is to specify explicitly either which data to keep writing to database tables or which data to write to accelerator-only tables. Figure 5-4 shows the last statements that ran on the accelerator when the procedure was run. Because the TOTAL_PROFITS table is saved to DB2, the last statement is the **SELECT** statement of query **TOTALPR**. The **INSERT** statements result from the **SAVE DATA AS SUMS_PROFIT_RED_AOT** command.

Because, at the time of writing, the DB2 Analytics Accelerator does not support multi-row inserts into an accelerator-only table, each single row is inserted using a single **INSERT** statement. This way of saving data to an accelerator-only table takes longer than saving data to a database table. Although queries can run faster on the accelerator than on DB2, saving the result might run faster on DB2 than on the accelerator. Whether a procedure completely runs faster on the accelerator than on DB2 depends on the types of queries used, and on the number of rows that are saved in the result tables.

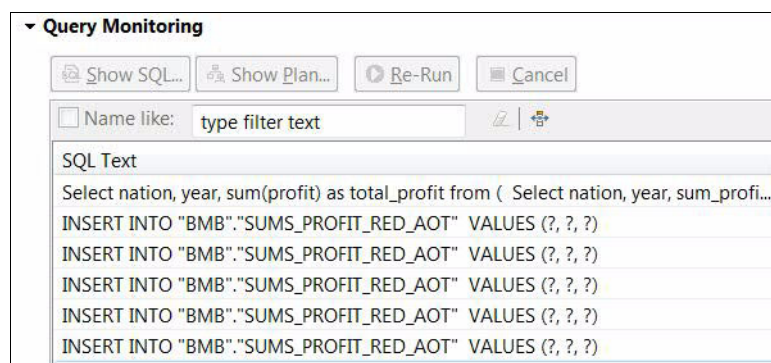


Figure 5-4 Statements of a QMF procedure using SAVE DATA AS run on the accelerator

Important: Because all QMF queries of the procedure run on the accelerator (either on the base tables or on the created accelerator-only tables), the **CURRENT QUERY ACCELERATION** register must be set appropriately. Set it either using **QUERY_ACCELERATION DSNZPARM**, through profile (DB2 11 only), or within a QMF query object.

5.3.5 Running the procedure in DB2 Analytics Accelerator using accelerator-only tables created by the RUN QUERY command

If you use DB2 Analytics Accelerator and QMF for z/OS V11.2, the procedure described in Example 5-4 on page 61 can benefit more from query acceleration and using accelerator-only tables in terms of performance than described in section 5.3.4, “Running the procedure in DB2 Analytics Accelerator using accelerator-only tables created by the SAVE DATA AS command” on page 63, if you adapt the procedure further. The **RUN QUERY** command can save the result into a table directly, without presenting the result to the user.

The **RUN QUERY** command saves the query result faster into an accelerator-only table than the **SAVE DATA AS** command, because the **RUN QUERY** command runs the query and saves the result into a table using a single **INSERT INTO SELECT FROM SQL** statement. If you change your existing procedures to use the **RUN QUERY** command to save query results into accelerator-only tables rather than using the **SAVE DATA AS** command, then **INSERT** performance increases.

In this section, we describe how to run the procedure using accelerator-only tables that are created using the **RUN QUERY** command, depending on the option that you choose for the QMF global variable **DSQEC_SAV_ALLOWED**. This variable is described in detail in 5.2.1, “Accelerator-only table support in QMF for z/OS V11.2” on page 56.

Using option 4 or using option 3 for variable **DSQEC_SAV_ALLOWED** enables you to save query results to either database tables or accelerator-only tables, within the same procedure or among multiple procedures.

If you use option 4 for variable **DSQEC_SAV_ALLOWED**, the procedure must be changed as shown in Example 5-8. The change is to write the results of the first four queries to accelerator-only tables on accelerator specified in **DSQEC_SAV_ACCELNM**, and to write the final result into a database table in table space **DSQDBDEF.DSQTSDEF**.

We run the examples using the TSO client of QMF for z/OS. In the TSO client, the Plus sign (+) as the first character in the line determines that the command is continued in this line.

Example 5-8 Procedure using RUN QUERY commands with DSQEC_SAV_ALLOWED set to option 4

```
RUN QUERY BMB.LINEPR (&&COLOUR='%green%' TABLE PRODLINE_PROFIT_GREEN_AOT
RUN QUERY BMB.LINEPR (&&COLOUR='%red%' TABLE PRODLINE_PROFIT_RED_AOT

RUN QUERY BMB.SUMPR (&&T=PRODLINE_PROFIT_GREEN_AOT TABLE SUMS_PR_GREEN_AOT
RUN QUERY BMB.SUMPR (&&T=PRODLINE_PROFIT_RED_AOT TABLE SUMS_PR_RED_AOT

RUN QUERY BMB.TOTALPR (&&T1=SUMS_PR_GREEN_AOT &&T2=SUMS_PR_RED_AOT
+ TABLE TOTAL_PROFITS SPACE DSQDBDEF.DSQTSDEF
```

If you use option 3 for variable **DSQEC_SAV_ALLOWED**, the procedure must be changed as shown in Example 5-9. The change is to write the results of the first four queries to accelerator-only tables on accelerator **STRIPER**, and to write the final result into a database table.

Example 5-9 Procedure using RUN QUERY commands with DSQEC_SAV_ALLOWED set to option 3

```
RUN QUERY BMB.LINEPR (&&COLOUR='%green%' TABLE PRODLINE_PROFIT_GREEN_AOT
+ ACCELERATOR STRIPER
RUN QUERY BMB.LINEPR (&&COLOUR='%red%' TABLE PRODLINE_PROFIT_RED_AOT
+ ACCELERATOR STRIPER

RUN QUERY BMB.SUMPR (&&T=PRODLINE_PROFIT_GREEN_AOT TABLE SUMS_PR_GREEN_AOT
+ ACCELERATOR STRIPER
```

```

RUN QUERY BMB.SUMPR (&T=PRODLIN_PROFIT_RED_AOT TABLE SUMS_PR_RED_AOT
+ ACCELERATOR STRIPER

RUN QUERY BMB.TOTALPR (&T1=SUMS_PR_GREEN_AOT &T2=SUMS_PR_RED_AOT
+ TABLE TOTAL_PROFITS

```

Figure 5-5 shows the list of all statements of the procedure that run on the accelerator. All **RUN QUERY** commands generate an **INSERT INTO SELECT FROM** statement that is run on the accelerator. Although the **TOTAL_PROFITS** table is created on DB2 and the **INSERT** itself is therefore run on DB2, the whole SQL statement is sent to the accelerator. However, only the **SELECT** part is run on the accelerator. The result set is returned to DB2 and inserted there into the **TOTAL_PROFITS** table.

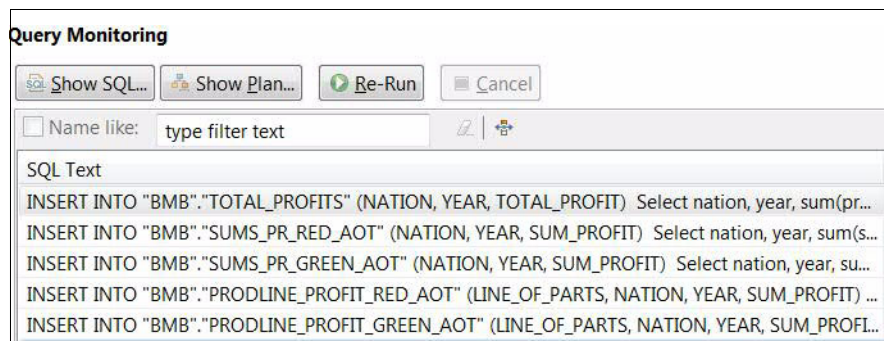


Figure 5-5 Statements of a QMF procedure using **RUN QUERY** run on the accelerator

Figure 5-6 proves that the **TOTAL_PROFITS** table is not created on the accelerator, although the **INSERT INTO TOTAL.PROFITS** statement is listed in the list of statements run on the accelerator. The figure shows, in alphabetical order, the list of tables available on the accelerator for user **BMB** after the procedure was run. The created accelerator-only tables of the first four **RUN QUERY** commands are contained in the list, but the **TOTAL_PROFITS** table is not, as expected.

Tables (61 of 103 loaded / 101 of 103 enabled for acceleration)

Name	Size	Acceleration	Last Load
BGDTPEN	-	Enabled	3/25/15 2:11 PM
DECORG	0 MB	Enabled	7/13/15 11:36 AM
PEDT042	-	Enabled	3/25/15 2:11 PM
PRODLIN_PROFIT_GREEN_AOT	15 MB	Enabled	Operational
PRODLIN_PROFIT_RED_AOT	15 MB	Enabled	Operational
STAFF	3 MB	Enabled	7/15/15 3:29 PM
SUMS_PR_GREEN_AOT	12 MB	Enabled	Operational
SUMS_PR_RED_AOT	11 MB	Enabled	Operational
TEST	-	Enabled	7/30/15 5:24 PM
[FILTERED] BUWD	12 ...	1 of 1	0 of 1 tables

Figure 5-6 List of tables on the accelerator including accelerator-only tables

You can also display tables and table types from the QMF TSO client. Use the **LIST TABLES** command to display the list of tables. Then use the **DESCRIBE** key next to a table to get the table description. If the table is an accelerator-only table, the subtype field shows **Accelerator Only Table**, as shown in Figure 5-7 on page 67.

Table List	
Name	Table Description
PLAN TABLE	
PRODLIN PROFIT G>	
PRODLIN PROFIT R>	
SPIEST1	
STAFF	
STAFFOUT	
STAFF1	
STAFF2	
STAFF3	
SUMS PR GREEN AOT	
SUMS PR RED AOT	BMB
SUMS PRODLIN PRO>	BMB
SUMS PRODLIN PRO>	BMB
TEST	BMB
TEST1	BMB
TOTAL PROFITS	BMB

Figure 5-7 List of tables and table description in QMF TSO client

Figure 5-8, Figure 5-9, and Figure 5-10 show the complete **SQL INSERT** statements generated from the **RUN QUERY** command. Only the **SELECT** part is coded in our QMF query objects, as described in 5.3.1, “Queries used in the sample scenario” on page 58. A **CREATE TABLE** statement and the **INSERT** part are generated during execution of the **RUN QUERY** command.

```
INSERT INTO BMB.PRODLIN_PROFIT_GREEN_AOT (LINE_OF_PARTS, NATION,
YEAR, SUM_PROFIT)
SELECT LINE_OF_PARTS, NATION, YEAR, SUM(AMOUNT) AS SUM_PROFIT
FROM
  (SELECT P_NAME AS LINE_OF_PARTS, N_NAME AS NATION, EXTRACT(year from o_orderdate) AS YEAR,
    L_EXTENDEDPRICE * (1 - L_DISCOUNT) - PS_SUPPLYCOST * L_QUANTITY AS AMOUNT
  FROM TPCH19.PART, TPCH19.SUPPLIER, TPCH19.LINEITEM,
    TPCH19.PARTSUPP, TPCH19.ORDERS, TPCH19.NATION
  WHERE S_SUPPKEY = L_SUPPKEY AND PS_SUPPKEY = L_SUPPKEY AND PS_PARTKEY = L_PARTKEY
    AND P_PARTKEY = L_PARTKEY AND O_ORDERKEY = L_ORDERKEY
    AND S_NATIONKEY = N_NATIONKEY AND P_NAME LIKE '%green%') AS PROFIT
GROUP BY LINE_OF_PARTS, NATION, YEAR
ORDER BY NATION, YEAR DESC
```

Figure 5-8 Generated INSERT statement using LINEPR query

```
INSERT INTO BMB.SUMS_PR_GREEN_AOT (NATION, YEAR, SUM_PROFIT)
SELECT NATION, YEAR, SUM(SUM_PROFIT) AS SUM_PROFIT
FROM PRODLIN_PROFIT_GREEN_AOT
GROUP BY NATION, YEAR
ORDER BY NATION, YEAR
```

Figure 5-9 Generated INSERT statement using SUMPR query

```
INSERT INTO BMB.TOTAL_PROFITS (NATION, YEAR, TOTAL_PROFIT)
SELECT NATION, YEAR, SUM(PROFIT) AS TOTAL_PROFIT
FROM (SELECT NATION, YEAR, SUM_PROFIT AS PROFIT
  FROM SUMS_PR_GREEN_AOT
  UNION ALL
  SELECT NATION, YEAR, SUM_PROFIT AS PROFIT
  FROM SUMS_PR_RED_AOT) AS PARTPROFIT
GROUP BY NATION, YEAR
ORDER BY YEAR, TOTAL_PROFIT DESC
```

Figure 5-10 Generated INSERT statement using TOTALPR query

Note: According to our tests, setting the **CURRENT QUERY ACCELERATION** register within the same QMF query object does not work if we use the **RUN QUERY** command to run the query and create a table from the result. An error occurs indicating that the query contains multiple statements. Only single statements are allowed in a QMF query object if the result should be saved to a table using the **RUN QUERY** command.

Creating accelerator-only tables from query results using QMF V11.1

Even if you still use QMF for z/OS V11.1, you can benefit from using accelerator-only tables and query acceleration. Adapt your procedures to use **RUN QUERY** commands only, and do not use **SAVE DATA AS** commands for results that you want to save as accelerator-only tables. Additionally, change your QMF query objects to create an accelerator-only table, followed by an **INSERT INTO SELECT FROM** statement.

Example 5-10 shows our adapted QMF procedure. We use the **SAVE DATA AS** command only for the last query, because we want to save the result in a DB2 table.

Example 5-10 QMF procedure for QMF V11.1

```
RUN QUERY BMB.LINEPR (&&COLOUR='%green%' &&T=PRODLIN_PROFIT_GREEN_AOT
RUN QUERY BMB.LINEPR (&&COLOUR='%red%' &&T=PRODLIN_PROFIT_RED_AOT

RUN QUERY BMB.SUMPR (&&T=SUMS_PR_GREEN_AOT &&QT=PRODLIN_PROFIT_GREEN_AOT
RUN QUERY BMB.SUMPR (&&T=SUMS_PR_RED_AOT &&QT=PRODLIN_PROFIT_RED_AOT

RUN QUERY BMB.TOTALPR (&&T1=SUMS_PR_GREEN_AOT &&T2=SUMS_PR_RED_AOT
SAVE DATA AS TOTAL_PROFITS
```

For the first four queries, the save part is hidden in the QMF query object. Each query object creates an accelerator-only table and inserts a query result into that table. Example 5-11 shows the statements of the **SUMPR** query object. The table name variables &T and &QT are substituted during run time by the values specified in the procedure.

Example 5-11 QMF V11.1 query object using accelerator-only tables

```
CREATE TABLE &T (NATION CHAR(25),
                  YEAR INT,
                  SUM_PROFIT DECIMAL(15,2))
IN ACCELERATOR STRIPER IN DATABASE BMBTEST;
COMMIT;

SET CURRENT QUERY ACCELERATION ELIGIBLE;

INSERT INTO &T (
Select nation, year, sum(sum_profit) as sum_profit from &QT group by
nation, year
order by nation, year );
```

5.4 Importing tables as accelerator-only tables

QMF for z/OS supports importing a table into the DB2 subsystem from a TSO data set or a UNIX file. With QMF for z/OS V11.2, this support has been extended to import the table to the accelerator directly as an accelerator-only table. This new support enables you to make tables from other DB2 subsystems, or other database management systems (DBMS), available on the accelerator for reporting or analysis purposes, without importing the data to the connected DB2 subsystem first.

We used the QMF **EXPORT** command to export a table from a DB2 subsystem into a TSO data set using the syntax **EXPORT TABLE BMB.STAFF TO 'BMB.QMF.STAFF'**. The default data format of the exported table is QMF, but you can also export tables in IXF, CSV, or XML format. These are also the data formats that the **IMPORT** command can use.

The syntax that you use for the **IMPORT** command might look different depending on the option the DSQEC_SAV_ALLOWED global variable is set to, and the accelerator name that is specified in the DSQEC_SAV_ACCELNM global variable. The global variables are described in detail in 5.2.1, “Accelerator-only table support in QMF for z/OS V11.2” on page 56.

If the QMF global variable DSQEC_SAV_ALLOWED is set to option 2 or 4, the following **IMPORT** command imports the table to the accelerator that is specified in the DSQEC_SAV_ACCELNM global variable:

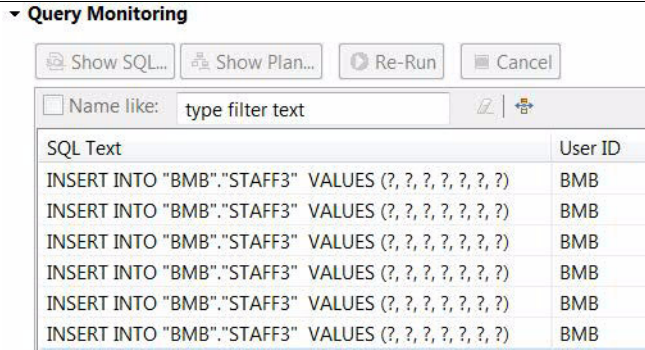
```
IMPORT TABLE BMB.STAFF2 from 'BMB.QMF.STAFF'
```

If you want to import the table to another accelerator than that specified in the DSQEC_SAV_ACCELNM variable, then extend the **IMPORT** command by specifying a dedicated accelerator name for the ACCELERATOR keyword.

If the QMF global variable DSQEC_SAV_ALLOWED is set to option 3, you must specify the ACCELERATOR keyword in the **IMPORT** command to import the table into the accelerator. Otherwise, the table is imported into the DB2 subsystem. For example, to import the table into the accelerator named STRIPER, use the following **IMPORT** command:

```
IMPORT TABLE BMB.STAFF3 from 'BMB.QMF.STAFF' (ACCELERATOR STRIPER
```

Because at the time of writing DB2 Analytics Accelerator does not support multi-row inserts into an accelerator-only table, during import each row is inserted using a single **INSERT** command. Therefore, importing tables as accelerator-only tables might take longer than importing tables as DB2 tables. However, after the import step has completed, queries that run on the imported accelerator-only tables can benefit from acceleration, and run faster than on DB2. Figure 5-11 shows the **INSERT** statements that are run on the accelerator while the **IMPORT** command runs.



SQL Text	User ID
INSERT INTO "BMB"."STAFF3" VALUES (?, ?, ?, ?, ?, ?)	BMB
INSERT INTO "BMB"."STAFF3" VALUES (?, ?, ?, ?, ?, ?)	BMB
INSERT INTO "BMB"."STAFF3" VALUES (?, ?, ?, ?, ?, ?)	BMB
INSERT INTO "BMB"."STAFF3" VALUES (?, ?, ?, ?, ?, ?)	BMB
INSERT INTO "BMB"."STAFF3" VALUES (?, ?, ?, ?, ?, ?)	BMB
INSERT INTO "BMB"."STAFF3" VALUES (?, ?, ?, ?, ?, ?)	BMB

Figure 5-11 Statements on accelerator during **IMPORT**

Remember: The **CURRENT QUERY ACCELERATION** register must be set appropriately to import data to an accelerator-only table. Set it either in advance using **QUERY_ACCELERATION DSNZPARM**, through profile (DB2 11 only), or within a dedicated QMF query object that you run before running the **IMPORT** command.

5.5 Preferred practices

In the previous sections, we described multiple examples of how to use the accelerator-only support in QMF.

The setting of the QMF global variable `DSQEC_SAV_ALLOWED` specifies the default behavior of whether data is written to database tables or accelerator-only tables if no location is specified in the QMF commands explicitly. If you start using QMF V11.2 and want to benefit from using accelerator-only tables, one of your first decisions is to which option to set this variable. These preferred practices might help you:

- ▶ Option 2 for this variable means that data can be saved only to accelerator-only tables, and not to database tables. This option might be appropriate in the following cases:
 - You want to save data to one accelerator-only table, even if you have multiple accelerators for workload balancing or HA/DR purposes.
 - No backup is needed for the saved data, because the data can always be re-created from tables in DB2.
 - You want to save data always in accelerator-only tables. There is no need to save any query result or import tables to database tables in DB2.
- If the performance of the procedures is important, use the **RUN QUERY** command rather than the **SAVE DATA AS** command to save query results. This ensures that the performance of the insert step into accelerator-only tables is optimized.
- You already have multiple existing QMF procedures in your environment that save data to accelerator tables and you do not want to change them at all.
- ▶ Option 3 or option 4 gives you more flexibility than option 2, because they enable you to save data to *either* accelerator-only tables or database tables. Option 3 saves the data per default to a database table, where option 4 saves it per default to an accelerator-only table. One of these options is appropriate in the following cases:
 - You want to use accelerator-only tables for storing temporary data only (for example, intermediate results of multistep processes), and you want to persist final data in database tables for backup purposes. These tables can be loaded to the accelerator as a subsequent step.
 - You have multiple accelerators for workload balancing or HA/DR purposes, and want to make some data available on all accelerators. This data must be saved to database tables and loaded to the accelerators afterward.
 - Changing existing QMF procedures is not an issue.

If you already have many existing procedures that save data to database tables, set `DSQEC_SAV_ALLOWED` to option 3. Afterward, your procedures run as before and save the data per default to database tables. This gives you the possibility to adapt your procedures step-by-step by adding the **ACCELERATOR** keyword to dedicated **SAVE** commands.

- If you start with QMF for z/OS, and you therefore do not have existing procedures yet, it depends on your preferred default behavior whether you choose option 3 or 4 for DSQEC_SAV_ALLOWED.
- If the performance of the procedures is important, use the **RUN QUERY** command rather than the **SAVE DATA AS** command to save query results. This ensures that the performance of the insert step into accelerator-only tables is optimized.



Accelerating IBM Campaign processing

IBM Campaign (formerly Unica) is another example of a software solution that can take advantage of accelerator-only tables and IBM DB2 Analytics Accelerator (Accelerator).

For efficient campaign processing, IBM Campaign creates database objects temporarily, which you can turn into accelerator-only tables by just changing configuration parameters of the corresponding data source. This makes processing large results faster and less processing (CPU)-intensive with IBM DB2 for z/OS on IBM z Systems.

This chapter explains IBM Campaign concepts and integration with DB2 for z/OS and DB2 Analytics Accelerator.

This chapter covers the following topics:

- ▶ What is IBM Campaign
- ▶ Components and architecture
- ▶ Our IBM Campaign environment
- ▶ Campaign example scenario used in this chapter

6.1 What is IBM Campaign

IBM Campaign is a campaign management software used to design, run, measure, and analyze outbound direct marketing campaigns. IBM Campaign supports all types of large-scale, multi-wave, and cross-channel campaigns. It also enables you to create a central repository for reusable offers, segments, and contact and response history, so marketers do not need to “reinvent the wheel” for each campaign.

IBM Campaign uses flowcharts to define campaign logic. Each flowchart in a campaign performs a sequence of actions on data that is stored in customer databases or in flat files.

IBM Campaign terminology

Campaign management and IBM Campaign use a set of common terms and concepts, which we describe in Table 6-1.

Table 6-1 Description of IBM Campaign terms

IBM Campaign term	Description
Flowchart	A flowchart is a collection (or set) of one or more tasks (or processes) that IBM Campaign runs to specify and select the wanted targets of a marketing campaign, and, optionally, assign offers and track responses.
Process	Processes are actions taken in the course of execution of a campaign, such as selecting data, creating segments, and assigning offers.
Cell	A cell is a list of IDs, such as customer IDs, that serve as input to, and output from, processes in a flowchart.
Offer	An offer is a single marketing communication or message that you send to targeted groups.
Campaign	A campaign is a container for one or more flowcharts. A campaign is created with a Summary, a Target Cell, and an Analysis tab.
Campaign code	A campaign code is a unique identifier, but campaign <i>names</i> do not have to be unique.
Table catalog	A table catalog is a collection of mappings, or connections, to data sources (tables and flat files) that contain data that you use in IBM Campaign.
Select process	A select process defines the criteria to locate a set or group, such as a list of customers, from marketing data. The select process outputs a cell containing a list of IDs, such as customer IDs, which can be modified and refined by other processes. Each select process in a flowchart takes one or more cells as input, transforms the data, and produces one or more cells as output.
Merge process	A merge process accepts input cells and outputs one combined cell.
Segment process	You can use Segment process to configure a segment by query. You can use a Segment process only within the flowchart where you created it. A segment process must have at least one input. If you input multiple cells to a Segment process, they all receive the same segmentation.
Sample process	With a Sample process, you can divide contacts into groups. In a typical use of a Sample process, you would set up target and control groups with which you can measure your marketing campaign.

IBM Campaign term	Description
System tables	System tables contain IBM Campaign application data. They are configured during installation.
User tables	User tables contain data about your company's customers, prospects, or products, for use in marketing campaigns. You must map user tables or files to make the data accessible for use in flowcharts.

System tables and user tables

IBM Campaign connects to different types of databases to access system and user data. *System* tables are used to store metadata for campaign processing, such as the layout and the status of flowcharts. In our context, *user* tables are more interesting: This is where potentially large customer and transactional data, which is used as input for campaigns, exists. We describe the advantages of the Accelerator if user tables are on DB2 for z/OS.

6.2 Components and architecture

Figure 6-1 shows logical components and architecture of an IBM Campaign system. Users access a web application server with a browser over Hypertext Transfer Protocol (HTTP). You can implement the back end as a single node, or a cluster, which runs an IBM Campaign listener service (`unica_acsvr`). The nodes connect to the IBM Campaign system tables through a native Open Database Connectivity (ODBC) connection.

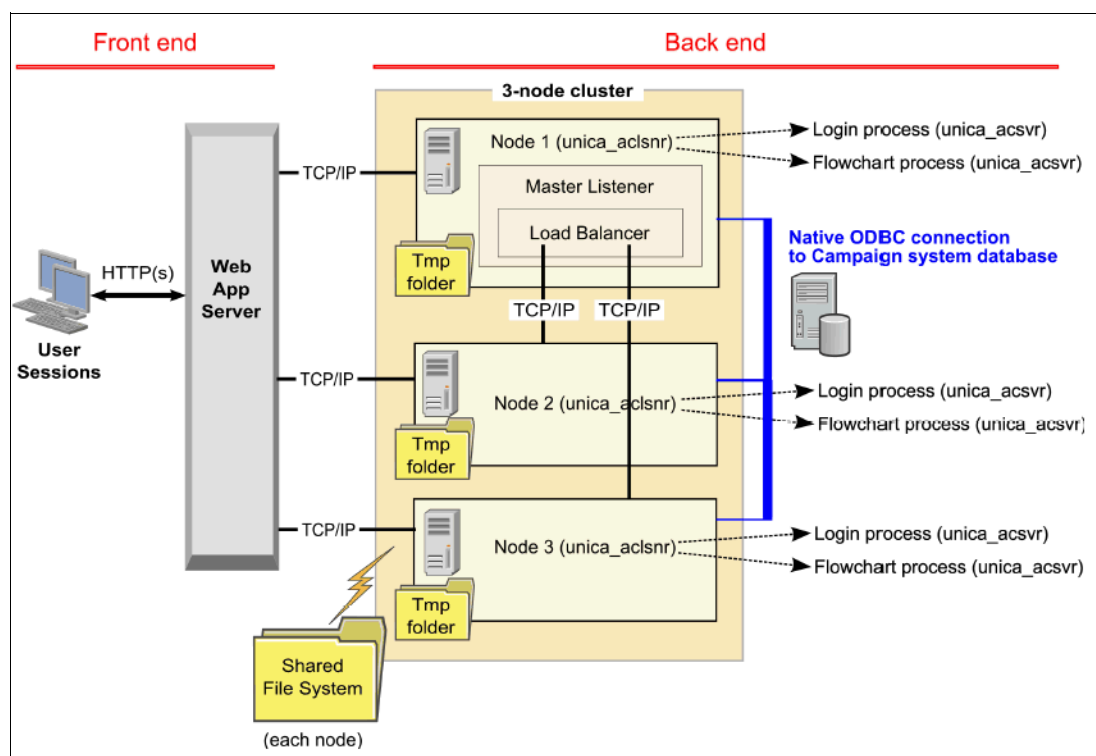


Figure 6-1 IBM Campaign components

IBM Campaign supports the following web application servers:

- ▶ IBM WebSphere Application Server
- ▶ Oracle WebLogic Server

IBM Campaign supports the following operating systems:

- ▶ Microsoft Windows Server
- ▶ IBM AIX
- ▶ Solaris
- ▶ Red Hat Enterprise Linux (RHEL)
- ▶ SUSE Linux Enterprise Server

Note: The back end for IBM Campaign cannot run on Linux on z Systems.

You can store system tables on the following database management systems:

- ▶ DB2 for Linux, UNIX, and Windows
- ▶ Oracle
- ▶ SQL Server

Note: With the current version of IBM Campaign, you cannot store system tables in DB2 for z/OS.

A full, detailed list of supported software requirements is available on the following website:

<http://www.ibm.com/support/docview.wss?uid=swg27044265&aid=1>

6.2.1 IBM Campaign and DB2 for z/OS

IBM Campaign version 9.1.1 supports user tables in DB2 for z/OS so that you can directly use and access transactional data originating from the z systems for campaign processing.

IBM Campaign supports DB2 for z/OS 10.1 and 11.0 with recommended service upgrade (RSU1205) and program update tape (PUT1205) in new function mode (NFM).

Based on configuration properties, the system detects the database management system and issues the appropriate Structured Query Language (SQL) statement.

When temporary tables are created, the data source property for the database must have parameter **DB2NotLoggedInitially** set to FALSE.

6.2.2 IBM Campaign performance considerations and usage of temp tables

When dealing with large volumes of data and multiple concurrent users, campaign processing performance becomes critical, because it impacts an organization's capability to quickly drive time-critical (for example, seasonal) campaigns.

A campaign defined in a multi-node flowchart can therefore use temporary tables and in-database optimization to run more efficiently.

Creating temporary tables helps to store intermediate results. The server uses rows directly from these temp tables for the processing of the subsequent flowchart elements. This can dramatically improve the flowchart execution performance.

After you enable temporary table usage, every time you add a box in an IBM Campaign flowchart, it creates a temporary database table with just a list of IDs. A simple select statement will run and save a table. When you create a subsequent select box and link the previous one to it, it uses the first temporary table as its starting point. This makes the queries more efficient by starting with a smaller base at each new level.

To explain the concept and benefits, let us assume a campaign selecting customers based on income (> USD 300,000) and state (living in the US state of Massachusetts, 'MA').

Figure 6-2 shows a section from the campaign flowchart containing two boxes, each of them represents a select process with a select from the customer table with 10 million rows. The first select process queries for customers whose income is > USD 300,000 or more, and the second select process queries for customers from State = 'MA'. The first box has 600,000 customers, the second one returns 1,500,000 customers. The flowchart section result consists of the intersection of these two groups.

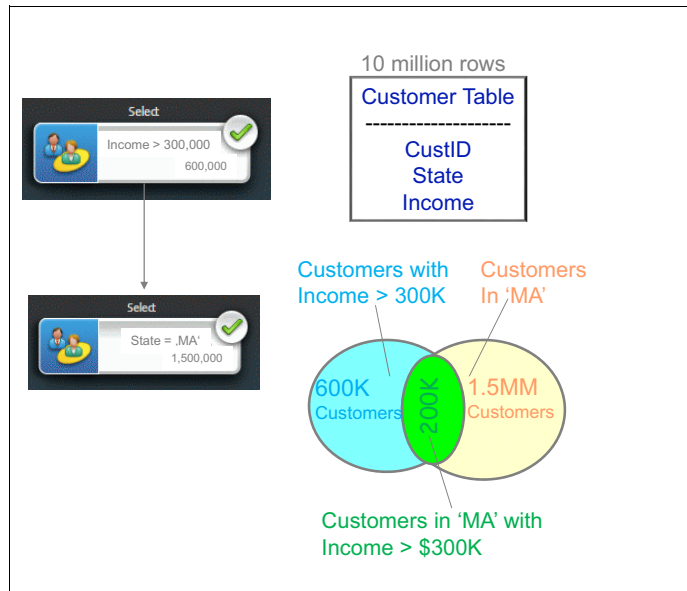


Figure 6-2 Section from a Campaign Flowchart with selects from the Customer table

Figure 6-3 describes what exactly happens when you do not use temp tables to run those select boxes.

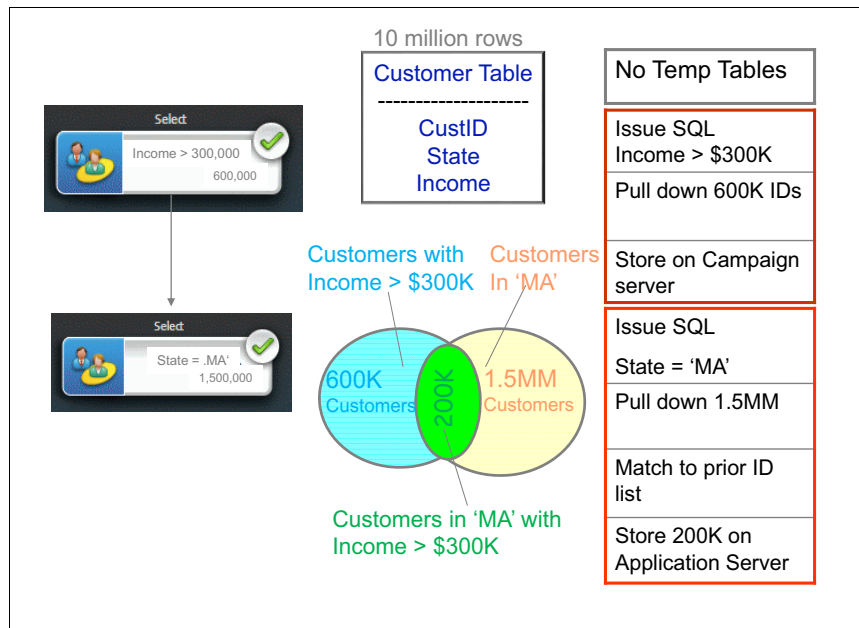


Figure 6-3 Execution without temp table

You can optimize the execution of the select boxes through the usage of temp tables. Figure 6-4 illustrates what happens when you use temp tables.

The temp table ID List stores the 600,000 ID results of the first query, enabling the subsequent query to join against it as a subset to apply the additional criterion (in this case Customers from 'MA' State). This is more optimal than having to run the second query against the entire table a second time.

Therefore, the query is quick because it uses the indexes in the second table and must query against a reduced number of records (600,000 rather than 10 million). The result of the two queries can be a new temporary table of only 200,000 records.

Nearly every time you run a select query it only stores the results of that query as a list of IDs (IDs are typically customer IDs or account IDs).

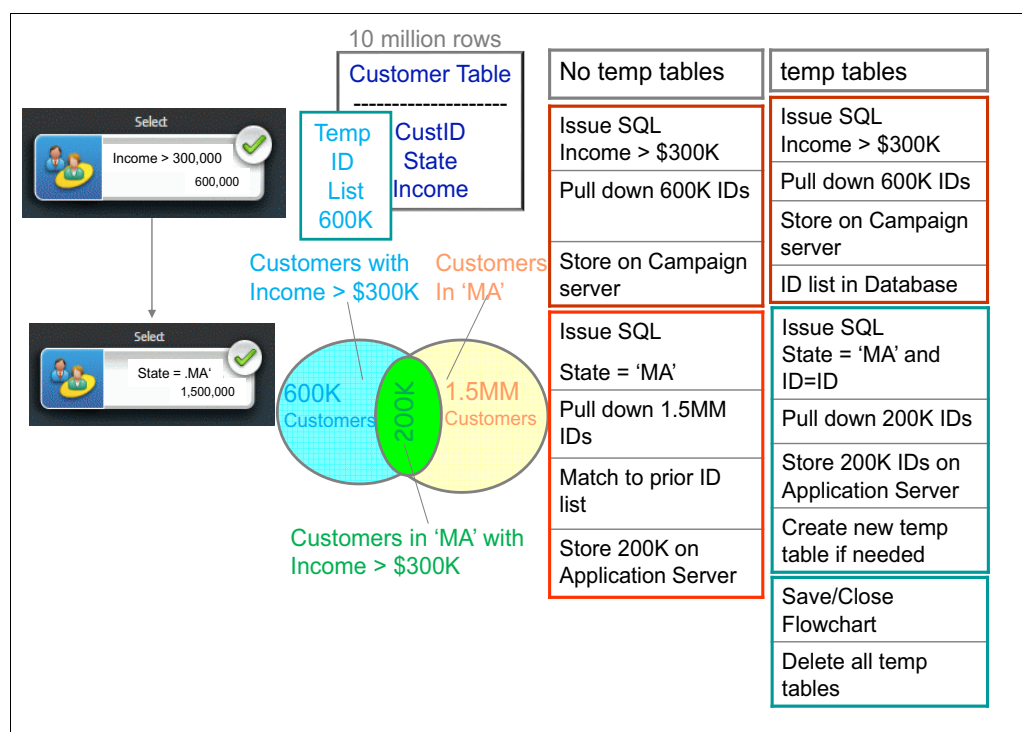


Figure 6-4 Optimizing flowchart execution using Temp table

6.2.3 Defining temporary tables

To initiate and enable the use of temporary tables, set the **AllowTempTables** parameter in the IBM Campaign Configuration settings. You must set this parameter for each data source (Campaign | partitions | partition[n] | dataSources | <datasourcename> | AllowTempTables).

Note: Although the term suggests otherwise, these are not DB2 global temporary tables. IBM Campaign just creates temporary common DB2 for z/OS base tables, in a database and a table space.

The following list provides several temp tables-related parameters:

- ▶ **AllowTempTables.** Use this parameter to enable or disable the use of temp tables. Set the parameter value to TRUE or FALSE.
- ▶ **SuffixOnTempTableCreation.** Use this parameter to force temp tables into a particular table space. Set the parameter value to the SQL statement that forces table into specified space for your database.
- ▶ **TempTablePrefix.** Use this parameter to define a schema, set a prefix for temp tables, and also add user ID tokens to temp table names. Set the parameter value to schema.prefix.<USER>. Example 6-1 shows the effect of setting **TempTablePrefix**.

Example 6-1 TempTablePrefix

CMP_TMP.UAC_<USER> results in UAC_kandrewsB.NMS02 for user kandrews with table in schema CMP_TMP

- ▶ **TempTablePostExecuteSQL.** Use this parameter to issue an SQL statement after the temp table has been created. Use an SQL statement as a parameter value.

Creating temporary tables on DB2 for z/OS means the management of **CREATE TABLE** statements having DB2 for z/OS rules, which requires specific settings and parameters:

- ▶ DB2 for z/OS requires you to specify an Owner Prefix in the table name.
- ▶ The Create statement on DB2 for z/OS requires you to specify IN DATABASE \$DBname or IN \$TSname.\$DBname.

6.3 Our IBM Campaign environment

For this book, we installed IBM Campaign in our test environment as depicted in Figure 6-5.

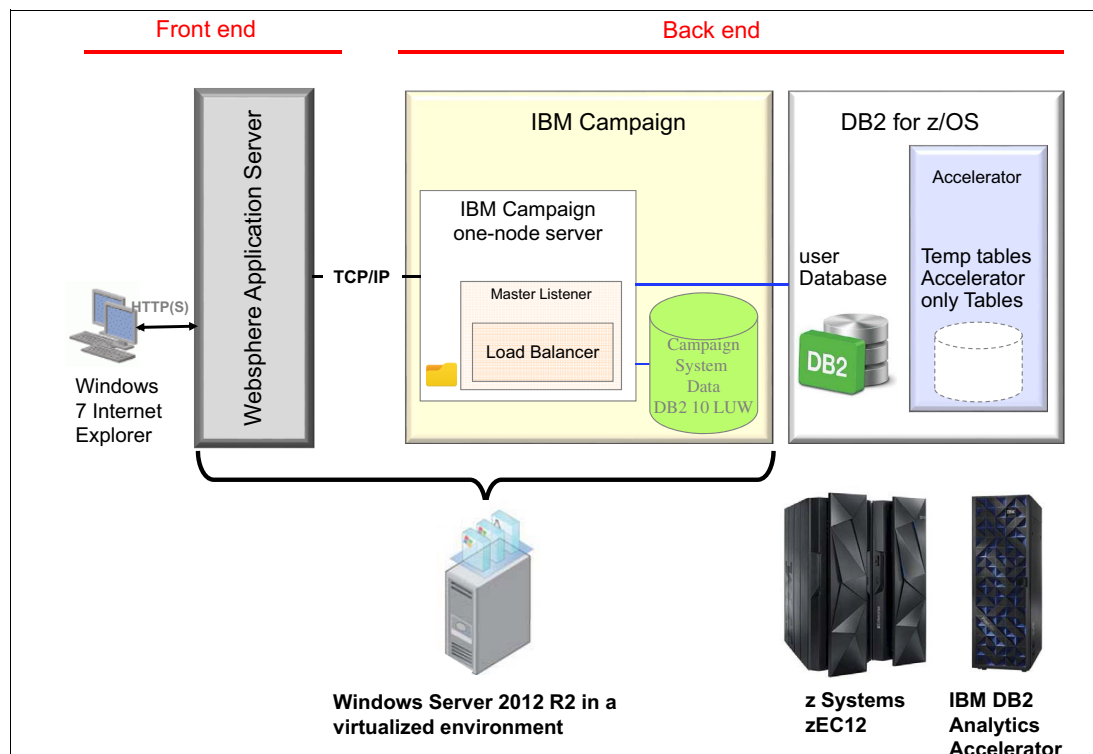


Figure 6-5 IBM Campaign test environment

We run WebSphere Application Server V8.5.5.2 and the IBM Campaign Listener on Microsoft Windows Server 2012 R2 in a virtualized environment.

We store system tables for IBM Campaign in a local DB2 for Linux, UNIX, and Windows V10.1 database. Figure 6-6 shows the well-defined Java Naming and Directory Interface (JNDI) name campaignPartition1DS from the data source definition pointing to the campaign system database.

General Properties

- Scope: cells:IDAA-ELTNode01Cell:nodes:IDAA-ELTNode01:servers:server1
- Provider: DB2 Using IBM JCC Driver
- Name: Campaign System Database
- JNDI name: campaignPartition1DS

Figure 6-6 Properties of system table data source definition in WebSphere Application Server

In our test environment, we use IBM zEnterprise EC12 (zEC12) to host DB2 for z/OS customer data (see Appendix A, “Description of IBM z Systems environment used for this publication” on page 183 for a detailed description of our mainframe environment).

We defined access to DB2 for z/OS in two places. First, a WebSphere Application Server data source establishes a JNDI name, which binds to a Java Database Connectivity (JDBC) type 4 connection. In our case, we use TMCC025 as the JDNDI name, with properties as shown in Figure 6-7.

Common and required data source properties

Name	Value
Driver type	4
Database name	CC825
Server name	tmcc-123-25.boeblingen.de.ibm.co
Port number	446

Figure 6-7 Properties of user table data source definition in WebSphere Application Server

In the browser-based configuration for IBM Campaign (**Settings** → **Configuration**), we create two user data sources for our tests: CC825 and CCC825-Accel (see Figure 6-8).

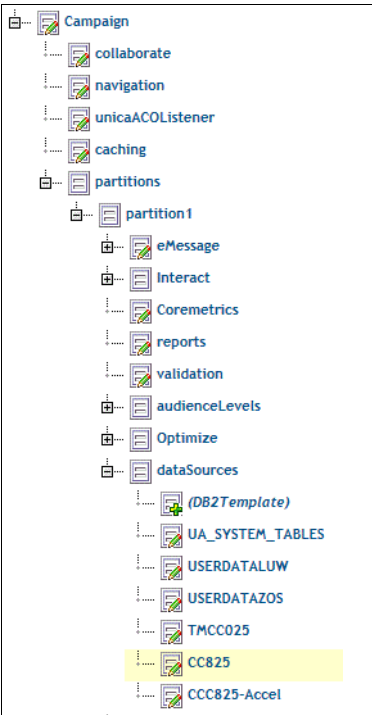


Figure 6-8 Data sources defined in IBM Campaign configuration

Both have the following properties:

- ▶ Type = DB2
- ▶ DB2NotLoggedInitially = FALSE
- ▶ JndiName = TMCC025
- ▶ AllowTempTables = TRUE
- ▶ TempTablePrefix = UAC

For our tests, we use a subset of the TPC-H tables. Because flowchart cells typically represent a list of customer IDs, we mainly use the CUSTOMER table for our campaign flowchart.

Example 6-2 shows the layout of the CUSTOMER table. The version we use has 150,000 customer entries and represents the list of customers from which we have a selection for a campaign example based on specific criteria.

Example 6-2 Table layout of CUSTOMER table

Select	Column Name	Col No	Col Type	Length	Scale	Null	Def	FP	Col	Card
*		*	*	*	*	*	*	*		*
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
	C_CUSTKEY	1	INTEGER	4	0	N	N	N		150000
	C_NAME	2	VARCHAR	25	0	Y	Y	N		137216
	C_ADDRESS	3	VARCHAR	40	0	Y	Y	N		150000
	C_NATIONKEY	4	INTEGER	4	0	Y	Y	N		25
	C_PHONE	5	CHAR	15	0	Y	Y	N		150000
	C_ACCTBAL	6	DECIMAL	10	2	Y	Y	N		150000
	C_MKTSEGMENT	7	CHAR	10	0	Y	Y	N		5
	C_COMMENT	8	VARCHAR	117	0	Y	Y	N		118784
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

6.4 Campaign example scenario used in this chapter

We create a campaign that an insurance company can use at the end of the year to make car insurance offers for customers from specific countries and cities who already own at least one car. One of the most important topics for insurance companies during this period of the year is customer retention and growth, which means that they must compete with each other in a limited time to gain new customers while retaining their existing customers.

Therefore, each insurance company works hard to launch an efficient car insurance campaign. Because time is quite critical in this case, we show in our example how companies can profit from temporary table usage and accelerator-only tables to optimize their campaigns.

For our tests, we select customers from Germany and France with cars, and create a segment process in IBM Campaign for further processing.

6.4.1 Flowchart for car insurance campaign

We start by creating a campaign named Car insurance campaign. In that campaign, we create an empty flowchart for Car holders in Germany and France using the CUSTOMER table with about 150,000 customer entries. Figure 6-9 shows the flowchart that we create. We identify in the first Select box COUNTRY customers from France (C_NATIONKEY=6) and Germany (C_NATIONKEY=7). The second box selects customers holding at least one car. The merged box result should represent the intersection of both Select box results. The merged box result should represent the intersection of both Select box results.

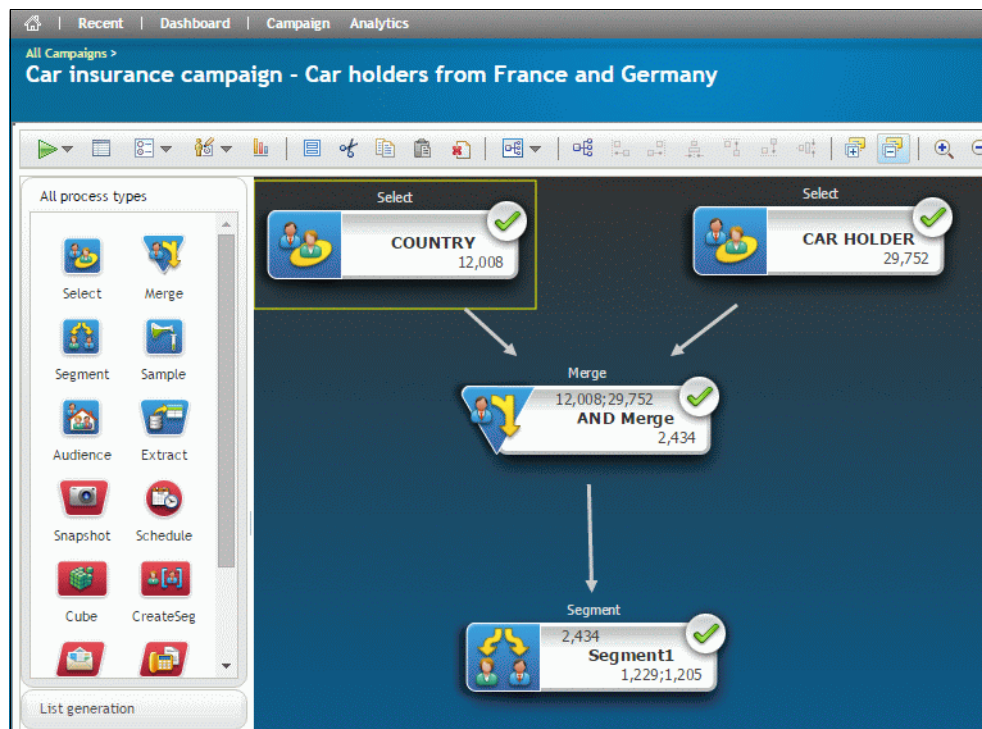


Figure 6-9 Flowchart in car insurance campaign

The last segment process consists of defining a segment each for customers from France and Germany. When we run this process, it submits a set of SQL statements to the referenced user table. The log of the process looks like that shown in Example 6-3 on page 83.

Example 6-3 Extract of an information log for flowchart without temp table usage

```
09/03/2015 12:24:50.787(4532)[I] [DB QUERY][COUNTRY]CC825 (thread
00000000000003B0): SELECT C_CUSTKEY FROM TPCH0.CUSTOMER WHERE
((TPCH0.CUSTOMER.C_NATIONKEY < 8) AND (TPCH0.CUSTOMER.C_NATIONKEY > 5)) ORDER BY
C_CUSTKEY [sdbtacc:2700]
...
09/03/2015 12:24:50.803(4532)[I] [DB QUERY][CAR HOLDER]CC825 (thread
0000000000000D60): SELECT C_CUSTKEY FROM TPCH0.CUSTOMER WHERE
(TPCH0.CUSTOMER.C_MKTSEGMENT = 'AUTOMOBILE') ORDER BY C_CUSTKEY [sdbtacc:2700]
...
09/03/2015 12:24:51.834(4532)[I] [TABLE ACC][COUNTRY]CC825 (thread
00000000000003B0): Data retrieval completed; 12008 records retrieved and returned
to caller. [sdbtacc:429]
...
09/03/2015 12:24:51.865(4532)[I] [TABLE ACC][CAR HOLDER]CC825 (thread
0000000000000D60): Data retrieval completed; 29752 records retrieved and returned
to caller. [sdbtacc:429]
...
09/03/2015 12:24:51.865(4532)[I] [CELL ACC][AND Merge]CellAccess: Data is ready.
09/03/2015 12:24:51.881(4532)[I] [CELL ACC][AND Merge]CellAccess: Data is ready.
09/03/2015 12:24:51.897(4532)[I] [PROCESS][AND Merge]Merge: N_RECORDS 2434
```

Example 6-3 and Figure 6-9 on page 82 show the results. Among our 150,000 customers, we have 29,752 car holders as a result of the CAR HOLDER box execution, and 12,008 customers coming from France and Germany as a COUNTRY box result. The cardinalities are shown at the lower right of each box after execution. Our campaign results in 2,434 qualifying customers with the two segments for Germany and France with 1229 and 1205 customers.

Note that this is just an example scenario. In real life, a campaign normally deals with a much larger number of customers.

6.4.2 Using temp tables for car insurance campaign

To enable temporary table usage, we select **Use In-DB Optimization during Flowchart Run** for our flowchart (see Figure 6-10).

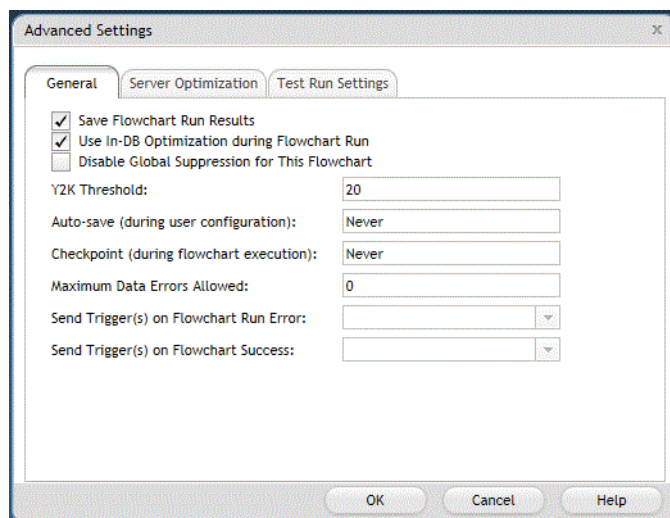


Figure 6-10 Enabling for In-DB optimization in Flowchart “Advanced Settings”

We also define a DB2 database, named UNICATMP, to keep the temporary tables. We set the data source property SuffixOnTempTableCreation for CC825 to IN DATABASE UNICATMP.

With these settings, the same flowchart shown in 6.4.1, “Flowchart for car insurance campaign” on page 82 now runs the SQL statements for the user data table, as shown in Example 6-4.

Note that three temporary tables are created: UAC_1_0, UAC_1_1 and UAC_1_2. They store the temporary selections of the nodes, and the final merge result.

Example 6-4 Log output for flowchart with temp tables and in-database optimization enabled

```
09/03/2015 12:48:59.643(5488)[I] [DB QUERY][COUNTRY]CC825 (thread
000000000000156C): CREATE TABLE UAC_1_0 (C_CUSTKEY INTEGER) IN DATABASE UNICATMP
[sdbtacc:2700]
09/03/2015 12:48:59.659(5488)[I] [DB QUERY][CAR HOLDER]CC825 (thread
00000000000015DC): CREATE TABLE UAC_1_1 (C_CUSTKEY INTEGER) IN DATABASE UNICATMP
[sdbtacc:2700]
09/03/2015 12:49:00.659(5488)[I] [DB QUERY][COUNTRY]CC825 (thread
0000000000000DD0): INSERT INTO UAC_1_0 SELECT C_CUSTKEY FROM TPCHO.CUSTOMER WHERE
((TPCHO.CUSTOMER.C_NATIONKEY < 8) AND (TPCHO.CUSTOMER.C_NATIONKEY > 5))
[sdbtacc:2700]
09/03/2015 12:49:00.674(5488)[I] [DB QUERY][CAR HOLDER]CC825 (thread
0000000000000E20): INSERT INTO UAC_1_1 SELECT C_CUSTKEY FROM TPCHO.CUSTOMER WHERE
(TPCHO.CUSTOMER.C_MKTSEGMENT = 'AUTOMOBILE') [sdbtacc:2700]
...
09/03/2015 12:49:01.706(5488)[I] [DB QUERY][AND Merge]CC825 (thread
0000000000000B98): CREATE TABLE UAC_1_2 (C_CUSTKEY INTEGER) IN DATABASE UNICATMP
[sdbtacc:2700]
09/03/2015 12:49:02.706(5488)[I] [DB QUERY][AND Merge]CC825 (thread
00000000000009C0): INSERT INTO UAC_1_2 SELECT A.C_CUSTKEY FROM (SELECT
UAC_1_0.C_CUSTKEY FROM UAC_1_0, UAC_1_1 WHERE UAC_1_0.C_CUSTKEY =
UAC_1_1.C_CUSTKEY) A [sdbtacc:2700]
09/03/2015 12:49:02.752(5488)[I] [TABLE ACC][AND Merge]CC825 (thread
00000000000009C0): Statement successful; 2434 records affected. [sdbtacc:2794]
```

Remember that we use a pretty trivial and small example for this book. For much larger user table data, this in-database processing is much more efficient than merging inside the application. In the next section, we see how to even further improve processing with the help of accelerator-only tables.

6.4.3 Using and enabling accelerator-only tables for car insurance campaign

Accelerator-only tables provide an alternative to DB2 for z/OS temporary tables, particularly for large intermediate results and if you already store source data on the Accelerator. You can extend Accelerator use for IBM Campaign, save execution time, and also avoid stressing the z/OS system during prime times.

To enable accelerator-only table usage, and also to support query routing to the accelerator, we use another data source (CCC825-Accl in our case), pointing to the same DB2 for z/OS database, but with the following changes:

- ▶ SQLOnConnect = "SET CURRENT QUERY ACCELERATION = ELIGIBLE"
- ▶ SuffixOnTempTableCreation = "IN ACCELERATOR STRIPER IN DATABASE UNICAAOT"

We use the **SQLOnConnect** parameter to make sure that queries are directed to the Accelerator if syntactically possible. This also required us to change accelerator-only table content during flowchart processing.

We added the setting for **SuffixOnTempTableCreation** to the **create table** statement, it follows the syntax for accelerator-only table creation, which we described earlier in this book. The Accelerator name we use in our environment is STRIPER.

In our environment, we create a DB2 database (UNICAA0T) that keeps the accelerator-only tables for IBM Campaign. We suggest using a dedicated database to ease management and control access privileges.

New runs of the flowchart process initially try to drop and then re-create temporary tables. The same is true if you use accelerator-only tables. If you should handle the lifecycle of these accelerator-only tables differently, a scheduled job could regularly check for accelerator-only tables in the defined database and clean them up, if desired.

For access rights, the user running campaigns might not have the DB2 privilege to create tables in all databases. By defining a dedicated database, database administrators (DBAs) can provide “create table” access rights to the user group running campaigns, and thereby enable them to use accelerator-only tables with this configuration.

Figure 6-11 shows the data source definition for CCC825-Acce1, which we use for this run of the flowchart.

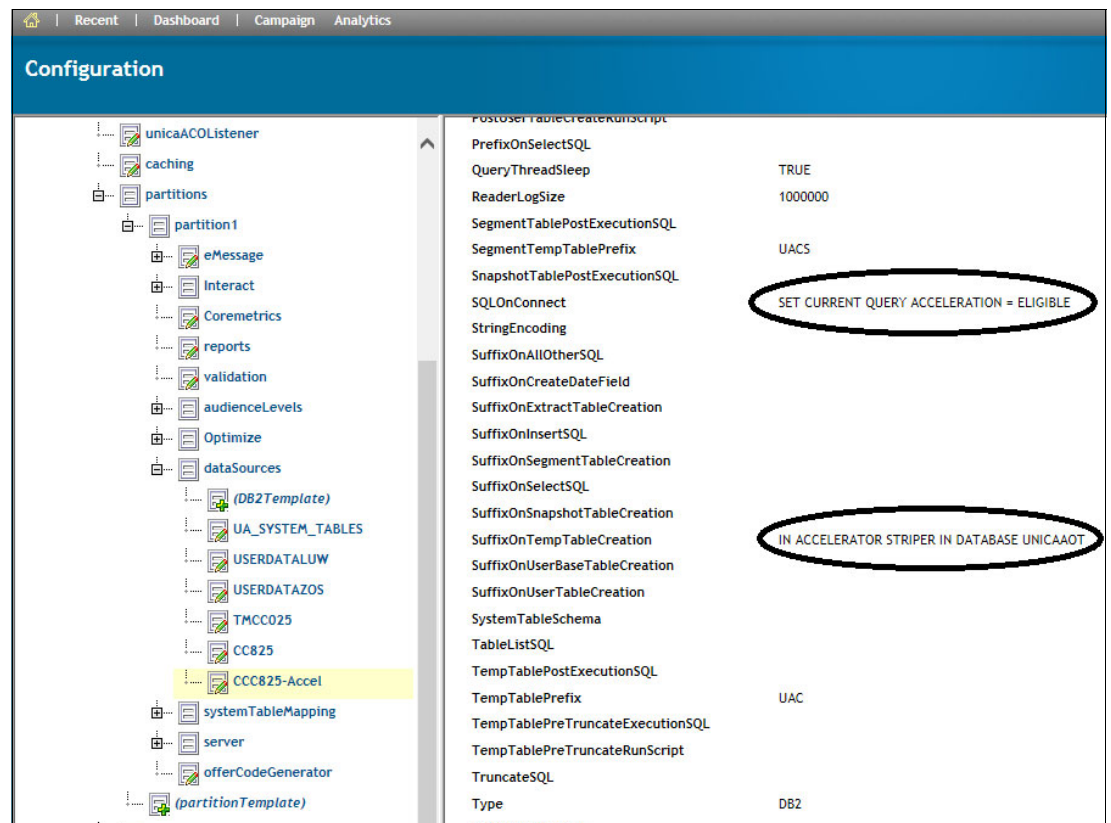


Figure 6-11 Parameters for query routing and accelerator-only table usage

With these settings, we show the SQL statements in Example 6-5 running for the user data table when we process the flowchart. The sequence is essentially the same as in 6.4.3, “Using and enabling accelerator-only tables for car insurance campaign” on page 84, except that IBM Campaign used a different syntax to create table objects, which leads to the creation of accelerator-only tables.

Example 6-5 Log output for flowchart with accelerator-only tables

```
09/03/2015 13:19:39.049(3780)[I] [DB QUERY][COUNTRY]CCC825-Accel: sql_on_connect
("SET CURRENT QUERY ACCELERATION = ELIGIBLE") completed successfully.
09/03/2015 13:19:39.049(3780)[I] [DB QUERY][COUNTRY]CCC825-Accel (thread
000000000000147C): CREATE TABLE UAC_3_1v (C_CUSTKEY INTEGER) IN ACCELERATOR
STRIPER IN DATABASE UNICAAOT [sdbtacc:2700]
09/03/2015 13:19:39.080(3780)[I] [DB QUERY][Automobile]CCC825-Accel (thread
000000000000087C): CREATE TABLE UAC_3_1u (C_CUSTKEY INTEGER) IN ACCELERATOR
STRIPER IN DATABASE UNICAAOT [sdbtacc:2700]
09/03/2015 13:19:40.096(3780)[I] [DB QUERY][COUNTRY]CCC825-Accel (thread
00000000000012F0): INSERT INTO UAC_3_1v SELECT C_CUSTKEY FROM TPCH0.CUSTOMER WHERE
((TPCH0.CUSTOMER.C_NATIONKEY > 5) AND (TPCH0.CUSTOMER.C_NATIONKEY < 8))
[sdbtacc:2700] 6.4.3, “Using and enabling accelerator-only tables for car
insurance campaign” on page 84
09/03/2015 13:19:41.096(3780)[I] [DB QUERY][Automobile]CCC825-Accel (thread
0000000000001298): INSERT INTO UAC_3_1u SELECT C_CUSTKEY FROM TPCH0.CUSTOMER WHERE
(TPCH0.CUSTOMER.C_MKTSEGMENT = 'AUTOMOBILE') [sdbtacc:2700]
09/03/2015 13:19:42.143(3780)[I] [DB QUERY][AND Merge]CCC825-Accel (thread
00000000000017F4): CREATE TABLE UAC_3_1w (C_CUSTKEY INTEGER) IN ACCELERATOR
STRIPER IN DATABASE UNICAAOT [sdbtacc:2700]
09/03/2015 13:19:43.143(3780)[I] [DB QUERY][AND Merge]CCC825-Accel (thread
0000000000001200): INSERT INTO UAC_3_1w SELECT A.C_CUSTKEY FROM (SELECT
UAC_3_1v.C_CUSTKEY FROM UAC_3_1v, UAC_3_1u WHERE UAC_3_1v.C_CUSTKEY =
UAC_3_1u.C_CUSTKEY) A [sdbtacc:2700]
...
```

To confirm that these statements actually run on the Accelerator (and use accelerator-only tables), you can use IBM Data Studio. Figure 6-12 shows the Query Monitoring section, and the statements issued by IBM Campaign and run on the Accelerator.

SQL Text	User ID	Start Time	State	Wait Time
INSERT INTO UAC_3_1y SELECT TPCH0.CUSTOMER.C_CUSTKEY FROM (...)	FNEUMAN	03.09.15 13:26:31	Successful	0 seconds
INSERT INTO UAC_3_1x SELECT TPCH0.CUSTOMER.C_CUSTKEY FROM (...)	FNEUMAN	03.09.15 13:26:30	Successful	0 seconds
INSERT INTO UAC_3_1w SELECT A.C_CUSTKEY FROM (SELECT UAC_3_1...	FNEUMAN	03.09.15 13:26:28	Successful	0 seconds
INSERT INTO UAC_3_1u SELECT C_CUSTKEY FROM TPCH0.CUSTOMER ...	FNEUMAN	03.09.15 13:26:26	Successful	0 seconds
INSERT INTO UAC_3_1v SELECT C_CUSTKEY FROM TPCH0.CUSTOMER ...	FNEUMAN	03.09.15 13:26:25	Successful	0 seconds

Figure 6-12 IIBM Campaign SQL Statements running on the Accelerator

For lifecycle management, for example cleanup of accelerator-only tables used in previous campaigns, the same rules apply as for temporary table creation. Whenever a new campaign is started, a drop SQL statement is run to delete an old table before it is reused again. Owner prefix for table names must be used to avoid name clashes for temporary database objects.



In-database transformations

This chapter describes the concept of *in-database transformations*. It also portrays custom implementations, such as self-written applications, which transform data from an operational model to a data model that is optimized from a performance perspective for reporting and analytical purposes.

In addition, this chapter provides information about how existing homegrown transformations can be migrated to Structured Query Language (SQL) to use accelerator-only tables. Finally, this chapter describes the use of IBM DB2 Analytics Accelerator for z/OS with IBM InfoSphere Information Server, focusing on IBM InfoSphere DataStage and Balanced Optimization.

This chapter covers the following topics:

- ▶ In-database transformations
- ▶ Custom transformation and extract, transform, and load processes
- ▶ Accelerator and accelerator-only table usage in IBM InfoSphere DataStage:
 - Configuration considerations when using InfoSphere Information Server with DB2 Analytics Accelerator
 - General usage of accelerator for queries in DataStage
 - Use of accelerator-only tables in DataStage
 - DataStage job optimization considerations
 - Accelerator maintenance using DataStage jobs
 - Pitfalls when using the accelerator with DataStage
 - Running other extract, transform, and load (ETL) tools in DataStage

7.1 In-database transformations

Most customer installations use two types of data stored within existing information technology (IT) systems. The most important one is stored *operational data*, which reflects the current information about an entity. A typical example of an entity is storing information about an existing customer with all of its attributes. This information must be legally accurate, and is stored in a system that holds a legal database.

All changes to an entity (a customer in our example) are entered into this system, reflecting relevant changes in real time to follow online read processes. Such a system is also called a *System of Record*. One example of a data change is a new address assigned to an existing customer that is represented in a front-end system, or an account statement immediately after an update was made.

The second kind of stored data is concerning *reporting and analytical requirements*. For regulative requirements, such as risk management or legally required financial reporting, existing data is analyzed by various means. Historically, this analysis was performed on a different system, either a relational database management system (RDBMS) or non-RDBMS.

Reasons for this approach are also of historical nature, because for decades it was not possible to perform efficient reporting over operational data structures. Reporting or analytical workloads running concurrently with operational workloads could easily impair business operations, and result in longer elapsed times or potential concurrency issues for mission-critical workload.

For this purpose, separate warehousing models, data marts, and operational data stores were created, depending on individual requirements. The process to move data to a different system can be implemented using various mechanisms, such as using SQL to move and transform data in a single step, or in multiple steps.

Though transforming data into different structures using SQL might not be the most efficient method in many cases, DB2 Analytics Accelerator enables DB2 for z/OS to perform these data transformations in an efficient way by using accelerator-only tables. Using accelerator-only tables, you can now perform SQL data transformations completely on the accelerator. You can also benefit from acceleration capabilities while calling the same application programming interface that you use for other SQL with DB2 for z/OS.

One trivial example of a simple transformation from operational into reporting data structures is to aggregate all account balances of a given financial customer and store a new entry with such an aggregated balance in a new table. In return, requests to list customers by their maximum total account balance are faster, because only a single row needs to be accessed. In contrast, without data transformation, all customer records must be read for every incoming request, performing the same calculations over and over again.

Processing steps required to transform data from operational to reporting-optimized structures are referred to as extract, transform, and load (ETL) and extract, load, and transform (ELT) processes. The difference between ETL and ELT is about the system actually performing these transformations. Within ETL processes, the transformation is performed with specific transformation tools at a system that can either be the source system or a specialized transformation system.

The ELT processes transform data into target structures within the target system. Inserting transformed data into a DB2 for z/OS table while selecting data from another system is considered to be the transformation step of an ELT process where the transformation is applied at the target system. Transforming data within a RDBMS is referred to as *in-database transformation*.

However, ELT processing is only one use case among many for in-database transformation. Example 7-1 shows a trivial example of an in-database transformation using SQL that can be part of an ELT process.

Example 7-1 ELT using INSERT from SELECT to aggregate customer account balances

```
INSERT INTO CUSTOMER_ACCOUNTS_DWH
(SELECT C_CUSTKEY, SUM(C_ACCTBAL)
 FROM CUSTOMER
 GROUP BY C_CUSTKEY);

UPDATE CUSTOMER_DWH
  SET C_RATING = 'A'
 WHERE C_ACCTBAL > 10000000;
```

This example inserts a single row into table CUSTOMER_ACCOUNTS_DWH that contains aggregated account balances of all accounts that a customer holds from table CUSTOMER. In a second step, the individual customer rating is set to a value of A if the aggregated account balance over all accounts exceeds 10,000,000.

Transformations in DB2 for z/OS

You might ask now: “So what is the big deal about accelerator-only tables? That is what can already be done in DB2 for z/OS.”

If you run similar transformation flows using a single **SQL INSERT FROM SELECT** statement within the same DB2 for z/OS subsystem or the same data sharing group (that is, inserting data from an online transaction processing (OLTP) member to a dedicated data warehouse (DWH) member), you cannot control **COMMIT** frequencies in a unit of recovery. This is especially important in DB2 for z/OS data sharing environments. If you do not commit, the following situations may occur:

- ▶ Do not free locked resources
- ▶ Prevent others from accessing your manipulated data
- ▶ Can cause timeouts for concurrently running applications
- ▶ Can prevent utilities from completing
- ▶ Might run into painful rollback scenarios

This has been a major inhibitor to use such an architecture for complex transformation processing involving multiple tables, most likely in the number of hundreds or thousands of tables for a complex ETL or ELT flow.

Another example of in-database transformation is based on the concept of *triggers*. Defining triggers on your operational tables provides you the ability to enable your OLTP transactions to update your data warehouse data incrementally whenever your transactional data changes. However, this method should only be used if data in your data warehouse can efficiently be accessed using index or unique index without significantly increasing the response time for transactions.

By implementing triggers for this purpose, you can also achieve in-database transformation before data is applied to your data warehouse. For more complex data transformations, calling a stored procedure from a trigger can also be an option. But due to the fact that each transaction increases in elapsed time if additional processing logic is added, either inside the application code or inside the database management system (DBMS), this approach is not suggested for time-critical applications.

Practically, all data manipulation using SQL on tables within the target system can be considered as in-database transformation. These are essentially for the following situations:

- ▶ Inserting data from another table (**INSERT FROM SELECT**)
- ▶ Updates on a target table
- ▶ Deletion of data from a target table

With accelerator-only tables, you can now run in-database transformations on an accelerator.

Enabling in-database transformation for DB2 Analytics Accelerator

Similar to other workloads running on DB2 Analytics Accelerator, running queries on accelerator-only tables must be enabled from a user perspective by one of the following settings:

- ▶ Through special register settings for dynamic SQL
- ▶ A **BIND** parameter for static SQL
- ▶ Using **DSNZPARM** for a system-wide default
- ▶ Through Java Database Connectivity (JDBC) or Open Database Connectivity (ODBC) driver properties
- ▶ DB2 profiles for remote applications

For more information about these settings, see *Enabling query acceleration with IBM DB2 Analytics Accelerator for ODBC and JDBC applications without modifying the applications* at the following web addresses:

<http://www.ibm.com/support/docview.wss?uid=swg27038078>

http://www.ibm.com/support/knowledgecenter/SSEPEK_11.0.0/com.ibm.db2z11.doc.sqlref/src/tpc/db2z_currentqueryacceleration.dita?cp=SSEPEK_11.0.0%2F6-1-4-6&lang=en

The intention of setting special register **CURRENT QUERY ACCELERATION** explicitly is to make sure that the requesting application can tolerate operating on potentially older data as it is stored in DB2 for z/OS when data is inserted into an accelerator-only table. When data stored in an accelerator-only table is modified, the only option for DB2 for z/OS is to send such a query to an accelerator for processing.

After an accelerator-only table was created in an accelerator, queries can access these tables if the special register **CURRENT QUERY ACCELERATION** was set to **ENABLE**, **ENABLEWITHFAILBACK**, **ELIGIBLE**, or **ALL** by one of the previously provided mechanisms. For more details about creating accelerator-only tables, see 2.3.1, “Creating accelerator-only tables” on page 18.

After an accelerator-only table was created in an accelerator, you can start populating the table and modifying the table content. See Example 7-2 for an illustration of how an accelerator-only table can be initially populated with data that is accessible on the accelerator.

Example 7-2 Initial population of an accelerator-only table

```
SET CURRENT QUERY ACCELERATION = ELIGIBLE;  
  
INSERT INTO schema.table  
(SELECT * FROM otherschema.othertable);
```

Note that you can only insert data into an accelerator-only table by SQL statements from one of the following sources:

- ▶ accelerator-shadow tables
- ▶ accelerator-archive tables
- ▶ accelerator-only tables

All tables accessed on an accelerator *must originate* from the same DB2 for z/OS subsystem or data sharing group in relation to your current connection. You cannot access data on an accelerator that was copied or moved to an accelerator from another DB2 for z/OS subsystem or data sharing group.

Note that you cannot update data on an accelerator in a transaction that also updates data in DB2 for z/OS or another accelerator.

Important: Updating data stored in accelerator-only tables is only allowed if no other data has been updated in DB2 for z/OS or another accelerator within the same unit of recovery. All data sources accessed in SQL statements inserting data into an accelerator-only table must be available in the accelerator.

7.2 Custom transformation and extract, transform, and load processes

Although transformation processes are largely deployed using ETL or ELT tools, such as Data Stage, many existing applications were created before ETL or ELT tools were widely available. In the past, SQL transformations were not necessarily a method of choice to move data from an operational data model to a different data model that is optimized for reporting. This was mainly because of performance penalties and the lack of **COMMIT** control (no regular **COMMIT** frequencies could be implemented if data was inserted into a target table using a single **INSERT FROM SELECT** statement).

Most commonly, the following mechanisms were used for performance reasons to transform data from an operational data model to a warehousing model, such as a star or snowflake schema, if no ETL or ELT tooling was available:

1. Unload a DB2 table into a sequential file.
2. Run custom application code to transform data into a target structure, including or excluding other information sources.
3. Load data into a new DB2 table.

You can run transformation processes, such as those pointed out in step 2, on z Systems or on other platforms. If another platform is involved, network transfer time does also contribute to the overall elapsed time. Depending on data volumes moved, networking time can account for a significant portion of the total elapsed time.

To keep network resource demands as low as possible, we suggest reducing network trips, and avoiding “data tourism” (sending data for further processing to other platforms). In addition, transform data (whenever possible) inside the database engine by using accelerator-only tables.

Further benefits of using accelerator-only tables include avoidance of unloading data from source structures and loading data into the final target structures. A side effect of this reduced effort is the reduced need for temporary storage, because no temporal data needs to be stored on a direct access storage device (DASD).

Especially for existing application environments, transformation processes can become tremendously complex, involve long chains of batch jobs, and take a long time to complete. Such complex processes to create efficient reporting or analytical structures are usually performed once every couple of days or weeks, but they rarely happen once or multiple times a day.

For our considerations, it is irrelevant if transformations are performed on or off the platform, because if you plan to migrate existing algorithms to accelerator-only tables, existing applications are likely to change.

We can summarize disadvantages of existing, complex transformations as follows:

- ▶ Potentially long elapsed times, depending on transformation complexity
- ▶ Extra network time due to data tourism, sending data back and forth between multiple platforms, including administration of these environments
- ▶ Storage costs to hold unloaded data

To benefit from accelerator-only tables, it is essential to understand the application logic that runs in transformation processes. It is key to understand if existing application logic can be transformed into SQL language components. Most transformation logic can be expressed using SQL statements, too.

However, this might not be the case for all algorithms. Nevertheless, if application and transformation logic can be expressed using SQL, and transformed statements can use available acceleration potential, they are likely good candidates for benefitting from accelerator-only tables.

Example 7-3 provides a pseudo source code snippet illustrating how to sum up account balances for all accounts of all customers from table CUSTOMER_ACCOUNTS and insert one resulting row per customer in table CUSTOMER_ACCOUNTS_DWH.

Example 7-3 Pseudo source code for data transformation

```
DECLARE CURSOR
  SELECT C_CUSTKEY, C_ACCTBAL FROM CUSTOMER_ACCOUNTS
  ORDER BY C_CUSTKEY

OPEN CURSOR

FETCH CURSOR
  INTO :C-CUSTKEY ,:C-ACCTBAL

C-CUSTKEY-PREV := C-CUSTKEY
C-ACCTBAL-SUM  := 0

PERFORM UNTIL SQLCODE = 100
  IF C-CUSTKEY = C-CUSTKEY-PREV THEN
    C-ACCTBAL-SUM := C-ACCTBAL-SUM + C-ACCTBAL

  ELSE
    INSERT INTO CUSTOMER_ACCOUNTS_DWH
      VALUES (:C-CUSTKEY-PREV ,:C-ACCTBAL-SUM)

    C-CUSTKEY-PREV := C-CUSTKEY
    C-ACCTBAL-SUM  := C-ACCTBAL
  END-IF

  FETCH CURSOR
    INTO :C-CUSTKEY ,:C-ACCTBAL

END-PERFORM

CLOSE CURSOR
```

From an application perspective, Example 7-3 fetches attributes for customer number (C_CUSTKEY) and account balance (C_ACCTBAL) from a source table, and adds up account balances per customer. After a new value for C_CUSTKEY is read, a row is inserted into target table CUSTOMER_ACCOUNTS_DWH. The corresponding SQL code representing the same algorithm is shown in Example 7-4.

Example 7-4 Corresponding SQL logic for data transformation

```
INSERT INTO CUSTOMER_ACCOUNTS_DWH
(SELECT C_CUSTKEY, SUM(C_ACCTBAL)
 FROM CUSTOMER_ACCOUNTS
 GROUP BY C_CUSTKEY);
```

If you compare both Example 7-3 on page 92 and Example 7-4, the application logic can easily be pushed into the database engine, reducing the time needed for round-trips between the application load module and DB2 for z/OS. As a rule, the invocation of the SQL interface (which is just issuing EXEC SQL from an application) without performing any additional SQL activities results in around 3,000 machine instructions.

As you can calculate, for 1,000,000 entries in a source table, the application algorithm runs (1,000,001 **FETCH** operations + **DECLARE CURSOR** + **OPEN CURSOR** + **CLOSE CURSOR**) x 3,000 = 300,012,000 machine instructions. If instead the logic is pushed into the database engine, only a single call of the SQL interface would be required to achieve the same goal, dramatically reducing the number of required round trips between an application and DB2 for z/OS.

If the transformation shown in Example 7-3 on page 92 were run on another system, you could further simplify your processes by eliminating data transfer to this platform while pushing the logic into the database engine. Although the migrated SQL statement can also be run within DB2 for z/OS, it is likely that the same SQL statement will run much quicker inside an accelerator, taking advantage of its massive parallel processing capabilities.

7.2.1 Consolidation and optimization

Although most data transformation processes can potentially be rewritten to use SQL and accelerator-only tables, it might not always be the most efficient approach to use SQL for all cases. We suggest investigating if SQL on accelerator-only tables could be a valid option in one or more of the following cases:

- ▶ Source data is stored in DB2 for z/OS.
- ▶ Networking time significantly contributes to existing elapsed times to perform transformations.
- ▶ Existing logic can be simplified by migrating it to SQL and old logic can be removed to reduce complexity.
- ▶ Migrated SQL is supported by the accelerator.
- ▶ No additional complexity is introduced by leveraging accelerator-only tables.
- ▶ Unloaded data causes storage costs.

It is important to note that current accelerator capabilities do not include backup and recovery of accelerator-only tables. To restore the content of accelerator-only tables in case it is no longer available, you must undertake one of the following actions:

- ▶ Re-create the content (in case it isn't available anymore).
- ▶ Propagate the content again using SQL Data Manipulation Language (DML) statements.

If re-creating data stored in an accelerator-only table can only be accomplished by rerunning complex transformations that (even if performed inside an accelerator) take some time to complete, you could implement your own backup and recovery steps by unloading and reloading data into accelerator-only tables. For further information about loading data into accelerator-only tables using IBM DB2 Analytics Loader for z/OS, see 9.3, “Support for accelerator-only tables” on page 169.

Note: Accelerator-only tables cannot be backed up and recovered by traditional utilities or existing Accelerator stored procedures.

If source data is stored in DB2 for z/OS and target data is stored on another platform, DB2 Analytics Accelerator provides an ultimate consolidation opportunity by transforming source data into target structures in an accelerator in a first step. In such scenarios, final data structures can be efficiently used to satisfy incoming query requests without further need to transfer results to another system. For more information about consolidation opportunities, see 3.5, “Reduce IT sprawl for analytics initiatives” on page 40.

You can also take it one step further and think about entirely avoiding the effort of building target data structures, within or outside of an accelerator. These data structures are not limited to an entirely complex reporting or analytical environment, but they can also exist in terms of data marts. Data marts are typically built for performance, and to answer specific business questions.

Maintaining data marts with the latest data always poses a challenge regarding a “single version of the truth” within any given environment. With DB2 Analytics Accelerator in place, there is a valid opportunity to consolidate those data marts. In many cases, so-called *virtual marts* can be defined on the original source data to maintain schema layouts without changing SQL statements accessing a mart.

In Table 7-1, we show potential examples for data mart consolidation using DB2 Analytics Accelerator.

Table 7-1 Examples for data mart consolidations

Scenario without DB2 Analytics Accelerator	Scenario including DB2 Analytics Accelerator
Multiple customer summary marts	Merge marts with the same topic but a different selection.
Payments in last three months	Removed. Rather, access total payment transaction history.
Transactions grouped by account type	Replaced with a virtual mart on source data with unchanged data schema.

If you are in the process of migrating existing ETL or ELT processes to benefit from accelerator-only tables, we highly advise also validating if reporting or analytical queries on the target structures can also be run on source data without transforming it. DB2 Analytics Accelerator can provide processing capabilities to efficiently run most variations of long-running workloads in an efficient way.

Tip: If you analyze existing ETL or ELT flows to migrate them to SQL syntax, check if final reports can be run on original source data that is copied to an accelerator without actually transforming it.

However, there might be use cases in which accelerator-only tables would not provide an efficient method to create final data structures. Sample cases could be application logics that require multiple sources and perform operations on data that you cannot simply transform to SQL language elements. Examples of those elements are:

- ▶ Application of specialized algorithms that are not available for SQL
- ▶ Unsupported functionality of required SQL inside an accelerator (see *IBM DB2 Analytics Accelerator for z/OS V4.1.0 User's Guide*, SH12-7040 for further details about existing SQL limitations)

7.2.2 Performance

Figure 7-1 shows class 2 elapsed times for inserting 1, 10, and 100 million rows into declared global temporary tables (CL2EL_DGTT) and partition-by-growth (CL2EL_PBG) tables within DB2 for z/OS and accelerator-only tables within an accelerator (CL2EL_IDT). Additionally, you can find the class 2 elapsed time for running the **SELECT** portion of the statement only, shown as CL2EL_Sel_only in Figure 7-1.

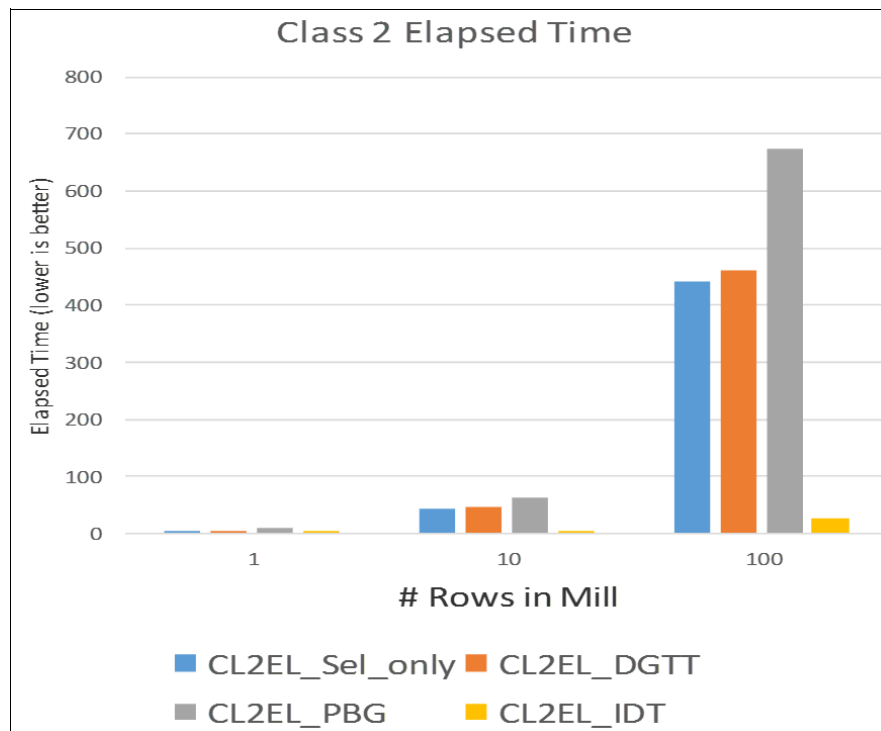


Figure 7-1 Class 2 elapsed time for inserting rows into different table structures

Figure 7-2 shows class 2 CPU times for inserting 1, 10, and 100 million rows into the same table structures that we used in Figure 7-1 on page 95. Note that we did not measure CPU time for running the **SELECT** portion of **INSERT FROM SELECT**. As a bottom line, we can tell that running **INSERT FROM SELECT** is both faster and nearly transparent in terms of class 2 CPU time on an accelerator compared to running the same statement in DB2 for z/OS, inserting data into common table structures within DB2 for z/OS.

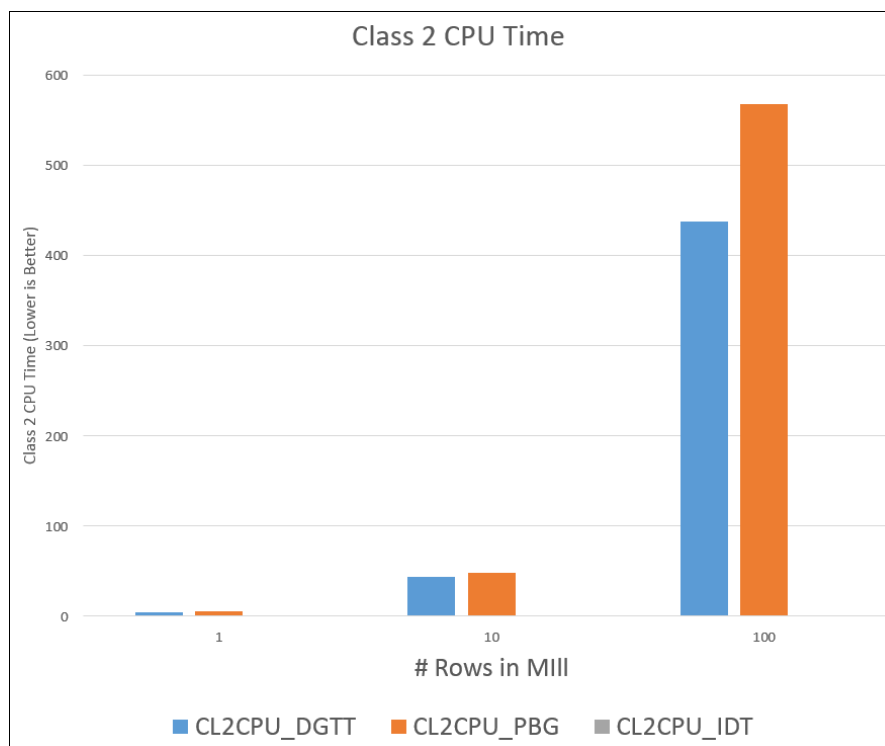


Figure 7-2 Class 2 CPU time for inserting rows into different table structures

It is these elapsed time improvements and CPU reduction that lead to the suggestion to run in-database transformations on an accelerator whenever it makes sense to run them there.

7.3 Accelerator and accelerator-only table usage in IBM InfoSphere DataStage

This chapter describes the usage of DB2 Analytics Accelerator for DB2 for z/OS (accelerator) with IBM InfoSphere Information Server, in detail DataStage and Balanced Optimization.

7.3.1 IBM InfoSphere Information Server

The following section provides an overview of IBM InfoSphere Information Server, its components and introduces Balanced Optimization.

Architecture and components

The IBM InfoSphere Information Server consists of Common services, Unified parallel processing, and Unified metadata as the core of the server architecture. Figure 7-3 on page 97 gives an overview of the server core components with various user interfaces (UIs) as a service-oriented architecture (SOA) with unified functions.

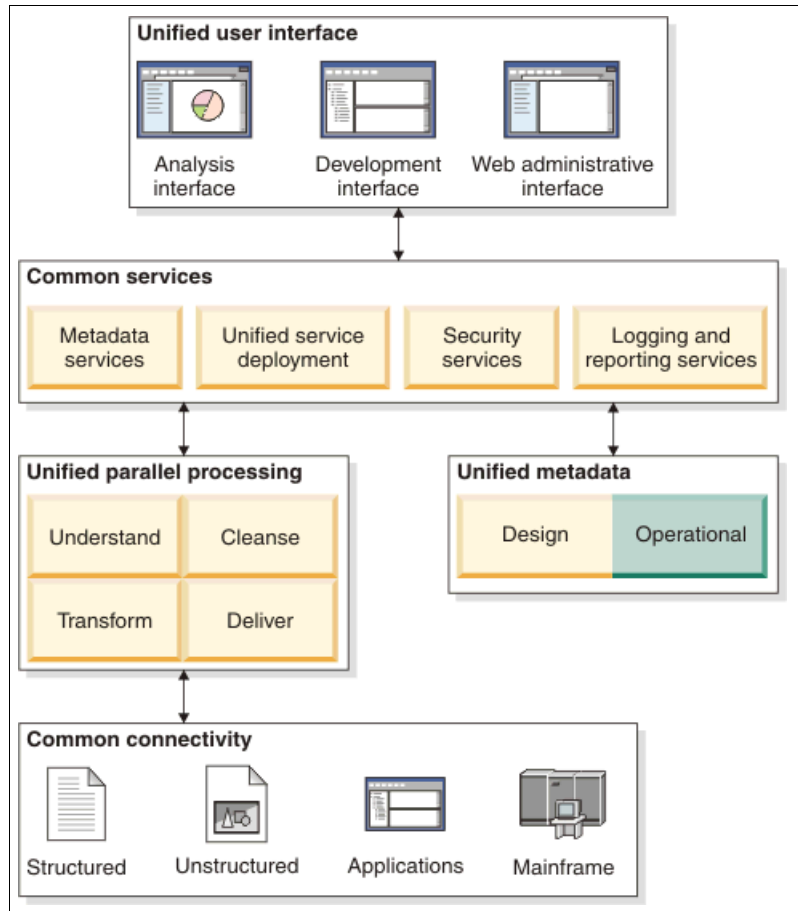


Figure 7-3 InfoSphere Information Server architecture

Unified parallel processing

The Unified parallel processing maps to parallel processing engines used by IBM InfoSphere Information Analyzer, IBM InfoSphere QualityStage, or IBM InfoSphere DataStage. Multiple engines can be used to parallelize the data pipelining process (staging of data while it is processed).

How the data is assigned to the individual engines is determined by the data partitioning approach (hashing, round robin, random, and others). If no resource constraints or other data skew issues exist, data partitioning can provide linear increases in application performance.

The process engines can run on different operation systems. On Microsoft Windows, IBM AIX, and Linux, each processing engine maps to a single computer if registered to the same Information Server service tier. In addition, AIX and Linux support multiple engines being run on the same computer, but must be registered to different service tiers.

Unified metadata

The unified metadata infrastructure enables shared understanding between business and technical domains, because it stores all InfoSphere Information Server suite components. All of the products depend on the repository to navigate, query, and update metadata, and support an easier collaboration of different roles and functions using the products. For example, profiling information created by InfoSphere Information Analyzer is instantly available to users of InfoSphere DataStage.

The two kinds of metadata being stored in the repository:

- ▶ Dynamic (including design-time information)
- ▶ Operational (including performance monitoring, audit data, log data, and much more)

From a technical perspective, the repository is a Java Platform, Enterprise Edition (Java EE) application that uses a standard relational database, such as IBM DB2, Oracle, or SQL Server for persistence (DB2 is provided with InfoSphere Information Server).

Common services

Those shared services include administrative tasks such as security, user administration, logging, and reporting and are managed and controlled in one place, regardless of which suite component is being used. It also includes the metadata services to provide access and analysis of metadata across. Three general categories of service exist:

- ▶ Design (to help developers create function-specific services)
- ▶ Execution (logging, scheduling, monitoring, reporting, security, and web framework)
- ▶ Metadata (metadata shared across tools)

From a technical perspective, the common services tier is deployed on Java EE-compliant application servers such as IBM WebSphere Application Server, which is included with InfoSphere Information Server.

Topology design

Because InfoSphere Information Server has unified components and tiers, various configurations are possible. The term *computer* is used to refer to a physical computer, logical partition (LPAR), or virtual environment.

A basic configuration, which is used to produce the samples listed in this chapter, can have all tiers (engine, service, and metadata repository) on one computer. This is shown on Figure 7-4.

The advantage of this setup is a minimal requirement on hardware and limited maintenance effort, although it lacks any high availability (HA) capabilities. For testing and demonstration purposes, this setup was used in this book.

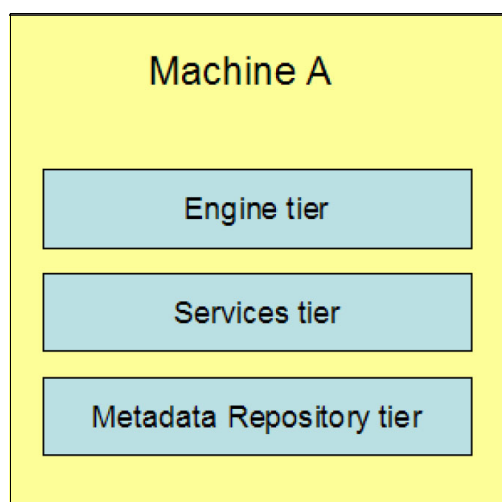


Figure 7-4 All tiers located on one computer

A more sophisticated topology (shown on Figure 7-5) uses an engine tier that is configured for parallel or grid processing. Either by using multiple processes of a symmetric multiprocessor (SMP) system or two or more computers for massively parallel processing (MPP) or cluster configurations.

This maximizes the performance of the engine tier. In combination with two computers in an active-standby configuration hosting the service tier and metadata repository tier, this topology provides some level of high availability where downtime can be tolerated.

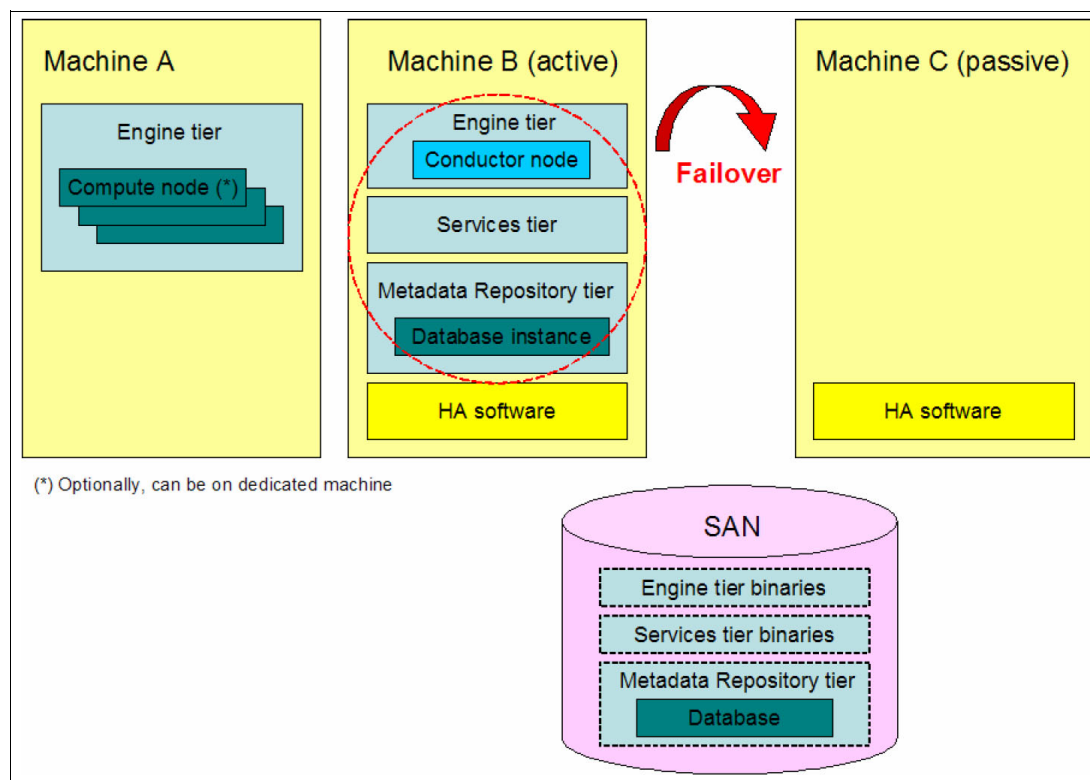


Figure 7-5 One computer or cluster for engine tier, one active and one passive computer for service tier and one SAN

A complete overview of the various topology designs, with a description of the advantages and disadvantages of each, is available in the white paper *IBM InfoSphere Information Server, Best Practices Topology*:

https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/Wc7a4f8c1c8ac_4813_9e13_1f2836f1287d/page/Designing+a+topology+for+InfoSphere+Information+Server

DataStage and System z

InfoSphere Information Server extends its capabilities to the mainframe with InfoSphere Information Server for System z and InfoSphere Classic. These solutions enable organizations to deliver trusted information for key business decisions and provide the most current information to people, processes, and applications.

InfoSphere Information Server for System z is a fully integrated software platform that profiles, cleanses, transforms, and delivers information from both mainframe and distributed data sources to drive greater insight for your business, without added IBM z/OS operational costs. It uniquely balances the reliability, scalability, and security of the System z platform with the low-cost processing environment of the Integrated Facility for the Linux specialty engine.

Hosting all of the fundamental information components of the solution on the System z server, including the operational data store, the data warehouse, and any data marts, addresses strategic business needs, centered around the demand for real-time or near real-time access to information to support key decision-making or business process requirements.

Combining InfoSphere Information Server with the DB2 Analytics Accelerator, and using Balanced Optimization with accelerator-only tables, can further boost this real-time access approach.

Accelerator usage within DataStage

Using the DB2 Analytics Accelerator with DataStage is possible in various ways.

For Data Stage jobs using DB2 connectors containing queries against DB2 data, the accelerator can be used to answer those queries. Depending on the complexity of the query and the number of returned rows, accelerator usage can be more or less useful. In case the accelerator-only tables are contained in the query, there is no way around using the accelerator.

Another scenario to use the accelerator is the integration of accelerator maintenance operations, such as table loads or unsynchronized tables detection within Data Stage jobs. Especially, table loads are necessary for job sequences where later jobs steps use the accelerator to answer queries or produce accelerator-only tables as final or intermediate results in the database.

The third scenario is optimizing existing jobs with InfoSphere DataStage Balanced Optimization. Depending on the used stages in the original jobs, this can involve the following actions:

- ▶ Acceleration of complex queries generated by Balanced Optimization
- ▶ Creation and usage of accelerator-only tables as the target for job steps
- ▶ Combination of the two previous points

Using Balanced Optimization to simplify jobs

Balanced Optimization (a licensed add-on to InfoSphere DataStage) can be used to optimize parallel jobs that use Teradata, IBM DB2, Netezza, or Oracle connectors and Big Data File Stage (BDFS) to access files on a Hadoop Distributed File System.

For a BDFS, Balanced Optimization produces JavaScript Object Notation (JSON) query language (Jaql) queries. For the database connectors, Balanced Optimization generates complex SQL statements that push the logic of the DataStage jobs into the database.

If the database connector is a DB2 connector, Balanced Optimization offers options to use the DB2 Analytics Accelerator to answer these complex queries. The user does not have to be an expert in SQL or Jaql to transform multistage DataStage jobs into their optimized and shortened versions.

Starting with Version 11.3 Fixpack 1 of InfoSphere Information Server, the corresponding Balanced Optimization code supports accelerator-only tables being generated throughout the optimization process.

The stages that are optimized when working with DB2 for z/OS

The optimization of DataStage jobs for database connectors in general, and for DB2 in particular, has some preconditions that are listed in total in “Processing stage types” (for Balanced Optimization).

Because native SQL does not provide all of the functionality available through DataStage job stages, be aware of the following constraints when designing jobs. The following job stages can be optimized by Balanced Optimization with preconditions:

- ▶ *Transformer stage*. If it is not a BASIC transformer stage or contains static and temporary variables.
- ▶ *Filter stage*. Unless Nulls value or Output rows only once options are set.
- ▶ *Aggregator stage*. Unless corrected sum of squares, missing value, preserve type, or summary aggregate function is used.
- ▶ *Remove Duplicates stage*. If it can be pushed to a source connector.
- ▶ *Join stage*. If it joins two inputs only.
- ▶ *Lookup stage*. Unless it is a range lookup or case-insensitive equality lookup on character-type column.
- ▶ *Copy stage*. Unless force options is set to true.
- ▶ *Sort stage*. Unless one of the following options is set to true:
 - Output Statistics
 - Create Key Change Column
 - Create Cluster Key Change Column
 - Sort as EBCDIC
 - Stable Sort

Alternatively, unless Nulls Position is set to Last.

- ▶ *Funnel stage*. Although not stated in the documentation, funnel stages are rewritten by Balanced Optimization into UNION expressions with SQL.

Balanced Optimization stops at any container boundaries. For local containers, use the Deconstruct option to make container contents part of the job. For shared container use the Convert to local container option and then deconstruct the local container.

Optimization of reject links is not widely supported, except for the Filter stage. The Transformer, Filter, and Copy processing stages can have multiple output links including reject links.

If a Transformer stage or Copy stage (used without a reject link) or a Filter stage have multiple output links, they can be optimized by being split into duplicate copies, one for each output link.

For a list of functions being used in a Transformer stage, see the overview of *Supported functions in SQL* in the IBM Knowledge Center:

https://www.ibm.com/support/knowledgecenter/SSZJPZ_11.3.0/com.ibm.swg.im.iis.ds.parrjob.dev.doc/topics/limitationsfunctions.html

7.3.2 Things to consider for configuration

InfoSphere Information Server offers different methods to access a database. In context of this book, and to use the how-tos that are shown later, a setup of a IBM DB2 Connect™ and an Open Database Connectivity (ODBC) connection to the database is needed. The ODBC connections are mainly used to import stored procedure definitions while connections through DB2 Connect-configured connector serves the actual call of stored procedures or statements.

Identify the DataDirect version

InfoSphere Information Server includes an embedded ODBC driver that is based on a DataDirect ODBC driver. To determine the version of the ODBC driver, see the information in the following Techdoc:

<http://www.ibm.com/support/docview.wss?uid=swg21574486>

The DataDirect driver version used in this book was 07.10.0022 (B0028, U0020). The DataDirect web page at the time of writing lists the version 7.1.5 to be the latest.

In case an update of the ODBC driver is necessary, the following Techdoc lists those steps:

<http://www.ibm.com/support/docview.wss?uid=swg21631331>

Configure ODBC connection to z/OS

To use an ODBC connection in DataStage, and also other components of the InfoSphere Information Server, you must set up the ODBC connector in its configuration files. Complete the following steps to configure the ODBC connection on the computer that serves as the processing engine:

1. Log in to the processing engine as user dsadm:

```
[dsadm@is-server ~]$ cd $DSHOME
[dsadm@is-server DSEngine]$ pwd
/opt/IBM/InformationServer/Server/DSEngine
[dsadm@is-server DSEngine]$
[dsadm@is-server DSEngine]$ . ./dsenv
```

2. Modify the .odbc.ini file to add an entry for a z/OS system:

```
[dsadm@is-server DSEngine]$ vi .odbc.ini
```

3. Add a new entry in the [ODBC Data Sources] section:

```
[ODBC Data Sources]
...
BB_CC_TMCC_123_25=Boeblingen Client Center TMCC-123-25
...
```

4. Provide the details for the new ODBC data source:

```
...
[BB_CC_TMCC_123_25]
Driver=/opt/IBM/InformationServer/Server/branded_odbc/lib/VMdb200.so
Description=Boeblingen Client Center TMCC-123-25
AddStringToCreateTable=
AlternateID=
Collection=NULLID
DynamicSections=100
GrantAuthid=PUBLIC
GrantExecute=1
IpAddress=tmcc-123-25.boeblingen.de.ibm.com
IsolationLevel=CURSOR_STABILITY
Location=CC825
LogonID=
Password=
Package=
PackageOwner=
TcpPort=446
WithHold=1
...
```

5. Edit `uvodbc.config` to add an entry to the value that was added to the `.odbc.ini`:

```
[dsadm@is-server DSEngine]$ vi uvodbc.config
```

```
[ODBC DATA SOURCES]
```

```
<localuv>
DBMSTYPE = UNIVERSE
network = TCP/IP
service = uvserver
host = localhost
```

```
...
<BB_CC_TMCC_123_25>
DBMSTYPE = ODBC
```

More information about how to manipulate the ODBC configuration files are available in *Enterprise Data Warehousing with DB2 9 for z/OS*, SG24-7637, and on the following website:

<http://datastage4you.blogspot.co.uk/2013/02/odbc-configuration-in-datastage.html>

To test the configuration, you can use the DataStage engine shell, as shown in Example 7-5. After sourcing `dsenv`, issue the commands shown.

Example 7-5 Use DataStage engine shell to test the ODBC connection

```
[dsadm@is-server DSEngine]$ bin/dssh
DataStage Command Language 11.3 Licensed Materials - Property of IBM
(c) Copyright IBM Corp. 1997, 2014 All Rights Reserved.
DSEngine logged on: Wednesday, April 22, 2015 15:22
```

```
>CONNECT BB_CC_TMCC_123_25
connect bb_cc_tmcc_123_25
BB_CC_TMCC_123_25> SELECT count(*) FROM STKNOL.REGION;
```

Replace the table `STKNOL.REGION` with one that exists in your setup. More details about testing the ODBC driver are also listed in the “Testing ODBC driver connectivity” IBM Knowledge Center on the following website:

http://www.ibm.com/support/knowledgecenter/SSZJPZ_8.1.0/com.ibm.swg.im.iis.ds.trouble.ref.doc/topics/ts_testingodbcdriverconnectivity.html

When trying to sent accelerated queries with the ODBC driver, consider the information in “Accelerating queries with the ODBC driver do not work” on page 145.

Create shared ODBC connection in DataStage

To create a shared ODBC connection in DataStage, complete the following steps:

1. Use the menu bar and select **Open File** → **New** → **Data Connection**.
2. In the following dialog, provide details about the ODBC connection. As shown in Figure 7-6, provide a name and a detailed description. For this chapter, the ODBC connection called BOE_CC_TMCC-123-25-ODBC is created.

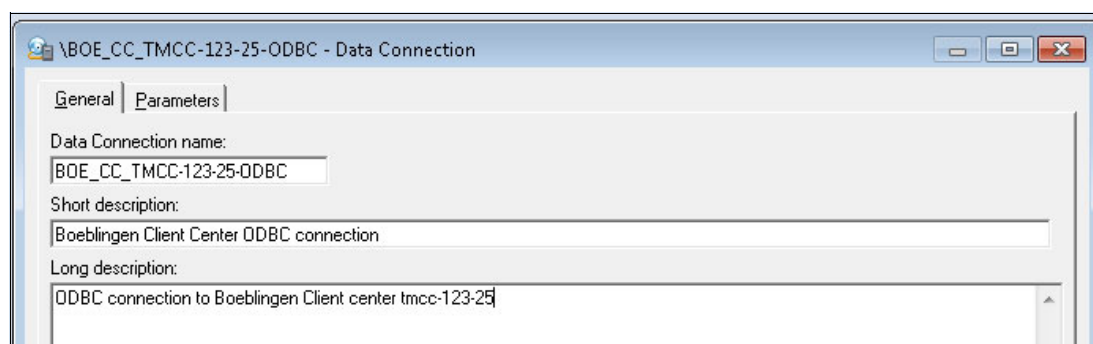


Figure 7-6 Creating a shared ODBC connector for further tests

3. After entering the information in the General tab, switch to the Parameters tab and select the Stage Type to be **ODBC Connection**. Insert the user name and password that you want to use for connectors that are associated with this shared data connection.
4. Click the **OK** button to store the shared container.

Identify the DB2 Connect version

Another way of connecting to DB2 is to use DB2 Connect. This requires the installation of a DB2 client, such as Connect Enterprise Edition.

If your InfoSphere Information Server installation already contains a DB2 client, log in to the processing engines and check their version. This is necessary to avoid the problem described in “Accelerated queries with DB2 Connect return wrong data” on page 146, where character data is not returned properly. To list the version of the DB2 client, use the command shown in Example 7-6 and make sure that the level is later than 10.5.0.2.

Example 7-6 Retrieve DB2 client level

```
[dsadm@is-server ~]$ db2ls
```

Install Path	Level	Fix Pack	Special Install Number	Install Date	Installer UID
/opt/IBM/DB2	10.5.0.5	5		Wed Apr 29 17:41:22 2015 CDT	0
/opt/IBM/db2/V10.5	10.5.0.5	5		Wed Apr 29 17:43:37 2015 CDT	0

```
[dsadm@is-server ~]$
```

For the InfoSphere Information Server installation used in this chapter, the metadata repository and the service tier share one computer, and a DB2 database is used to store this information.

Configure DB2 Connect to z/OS

The configuration of the DB2 client must be done on all DataStage processing engines. The “Configuring access to DB2 databases” website lists the necessary steps, such as adding the installation to the library path. The web address is as follows:

http://www.ibm.com/support/knowledgecenter/SSZJPZ_11.3.0/com.ibm.swg.im.iis.conn.common.usage.doc/topics/t_configuring_db2conn.html?lang=en

In addition to those steps, the connection to the DB2 system requires cataloging the access information. To do this, complete the following steps:

1. Log in to the processing engine as user dsadm and type db2 to start the DB2 command-line interface (CLI). Catalog a node similar to the following sample:

```
db2 catalog TCPIP4 node cc12325 REMOTE tmcc-123-25.boeblingen.de.ibm.com server
446 ostype mvs
```

The node is named cc12325 and connects through Transmission Control Protocol/Internet Protocol (TCP/IP) to port 446 on the server tmcc-123-25.boeblingen.de.ibm.com.

2. To validate that the node is properly cataloged, check the node directory using the **list** command:

```
db2 list node directory
```

3. As a next step, **catalog** the database, which in our case is named CC825:

```
db2 catalog database CC825 at node cc12325 authentication SERVER_ENCRYPT
```

4. To validate the database directory, use the **list** command:

```
db2 list db directory
```

5. To connect to the newly cataloged database, use the **connect** command:

```
DB2 Connect to CC825 user stkno1
```

6. To test the connection to DB2 and the accelerator, use the DB2 CLI again. It is possible to set the special register CURRENT QUERY ACCELERATION and run a read-only query. The accelerated query appears in the accelerators query monitor section in IBM Data Studio:

```
SET CURRENT QUERY ACCELERATION = ALL;
SELECT * FROM STKNOL.REGION for read only;
```

When testing the connection, also make sure to query some character data to avoid the problem described in “Accelerated queries with DB2 Connect return wrong data” on page 146.

Create DB2 connector in DataStage

To create a shared DB2 Connect connection within DataStage, complete the following steps:

1. Use the menu bar and select **Open File** → **New** → **Data Connection**. In the following dialog, provide details about the DB2 connection. As shown in Figure 7-7, provide a name and a detailed description. For this chapter, the DB2 Connect connection called BOE_CC_TMCC-123-25-DB2Conn is created.

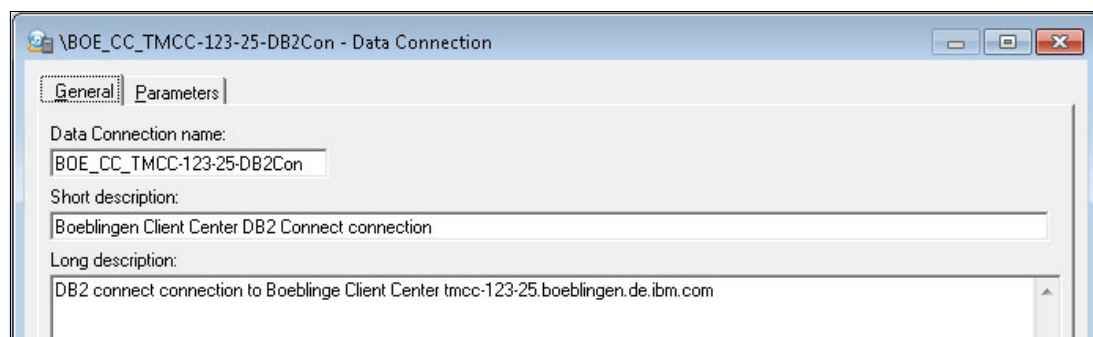


Figure 7-7 Creating a shared DB2 Connect connector for further tests

2. After entering the information in the General tab, switch to the Parameters tab and select the Stage Type to be **DB2 Connection**. Update the user name and password information to associate this information with DB2 connectors created within jobs.
3. Click the **OK** button to store the shared container.

7.3.3 Basic accelerator usage within Data Stage

This section covers the basic usage of an accelerator in DataStage. It describes how to accelerate queries and configure the specification of special registers. In addition, it demonstrates how to integrate accelerator management function like loading data through calls to stored procedures.

How to query accelerator shadow tables in DataStage

Consider a simple example where a query is either generated or specified within a connector stage to use the data for further processing (Figure 7-8). This job was stored in a project called SampleProject and the job name was Job001_002_DB2Con. It is possible to direct the SQL against the accelerator by specifying the special register setting in the connector stage.

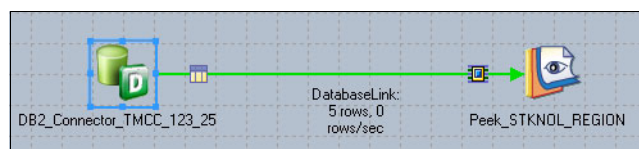


Figure 7-8 Sample job with connector to query an accelerator shadow table

To do so, add the specification of the special register specification in the Before SQL (node) section of the connector. For our example, we use the following statement:

```
SET CURRENT QUERY ACCELERATION = ALL
```

Let the connector stage generate a query for the table STKNOL.REGION. Of course, the table needs to be added and loaded on an accelerator that is paired with the DB2 subsystem that we are using. After compiling and running the job, the query appears in the query monitor section of the accelerator.

The correlation ID displays as osh, and the client application name displays as SampleProject:Job001_002_DB2Con, so you can easily correlate which job issued the query.

If you have troubles finding the query in the query monitor section within IBM Data Studio, there might be problem. Look at “Queries from DataStage are not running on the accelerator” on page 146 to find the cause.

Consider a shared Parameter Set for special register selection

Because specifying the special register for every connector is cumbersome, it is easier to provide a parameter set with the special register settings as a shared component. To do this, complete the following steps:

1. Create a new parameter set using **New** → **Other** → **Parameter Set**, give it a “speaking name”, such as QueryAccelOptionParameterSet, and a short and long description.
2. On the Parameter tab, add a Parameter name, which is to be used later within the connectors. As shown in Figure 7-9, the parameter can be named CurrQueryAccel.

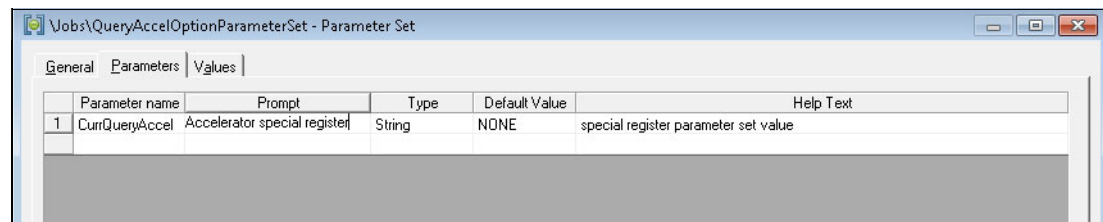


Figure 7-9 Parameter name and prompt in parameter set with default value

3. On the Values tab, specify the possible values for the special register (Figure 7-10) following the documentation of the values in the following IBM Knowledge Center:
http://www.ibm.com/support/knowledgecenter/SSEPEK_10.0.0/com.ibm.db2z10.doc.sqlref/src/tpc/db2z_currentqueryacceleration.dita?cp=SSEPEK_10.0.0%2F6-1-4-6

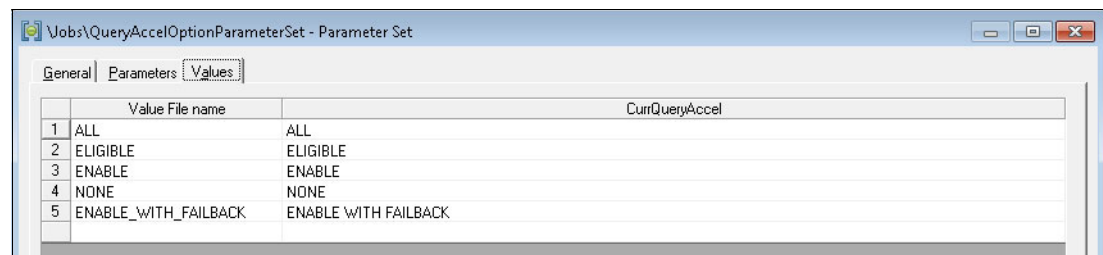


Figure 7-10 Special register specification in parameter set

4. After the Parameter Set is stored as a component in the metadata repository, go back to our previous job, which uses special register settings for the query. Open the job properties and modify the job parameters. On the Parameters tab, click **Add Parameter Set** and select the previously created parameter set.
5. Open the data connector that performs the query and modify the **Before SQL (node)** section of the connector. Only enter the following information:
 SET CURRENT QUERY ACCELERATION =

- Place the cursor at the end after the Equal (=) sign. Click **Use Job Parameter**, as shown in Figure 7-11, and select the parameter set. Copy the value and edit the **Before SQL (node)** again.

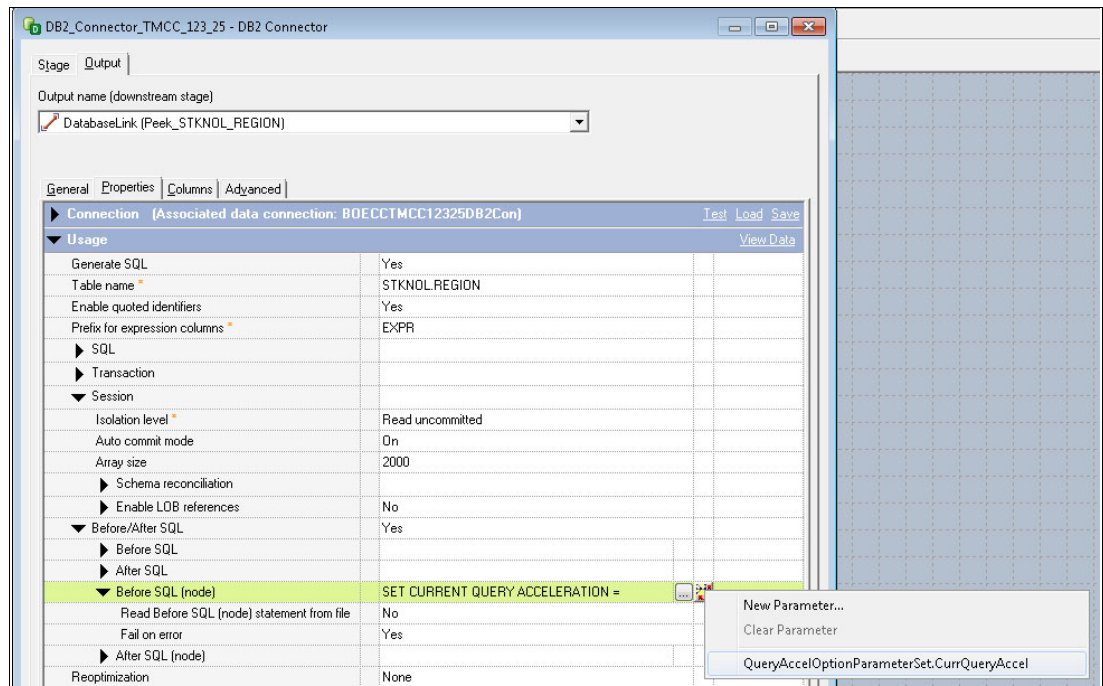


Figure 7-11 Parameter set selection within the database connector

- Overwrite the parameter set usage with a combination, so that it finally contains the following information:

```
SET CURRENT QUERY ACCELERATION = #QueryAccelOptionParameterSet.CurrQueryAccel#
```
- When running the job now, a pop-up window displays, prompting you to set the special register through a pull-down menu (Figure 7-12).

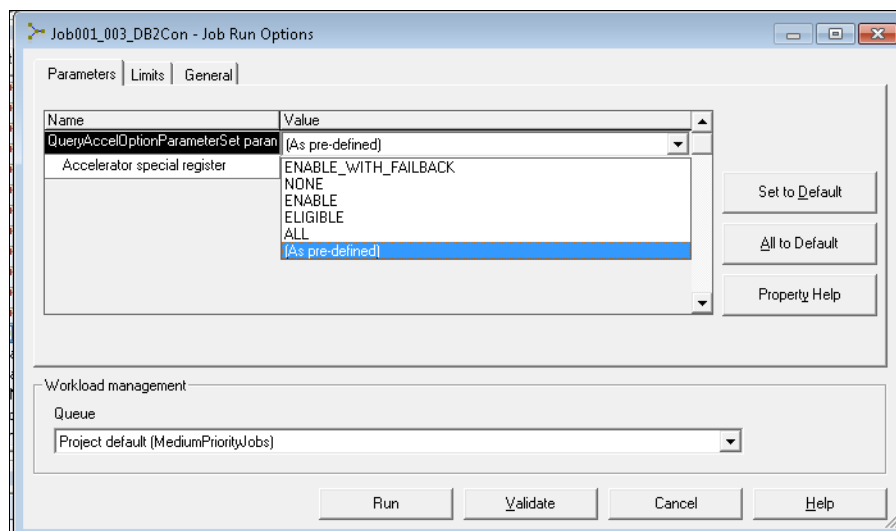


Figure 7-12 Prompt about special register value through Parameter Set

How to use accelerator-only tables in DataStage

It is possible to work with accelerator-only tables (AOTs) within DataStage as the target of a DML operation from a database connector. Be aware that multi-row insert operations currently are not working on AOTs. Because DataStage by default generates statements that use multi-row inserts, the connector property needs to be changed to work with AOTs. This has an effect on the performance of the operation.

Because AOTs do not support the specification of a primary key, see Chapter 2, “Accelerator-only tables” on page 11. One column should contain values that are unique to make update operations easier.

Inserting data into an AOT

It is possible to let AOTs be generated by the DB2 connector stage. In Figure 7-13, a row generator is used to generate 10 rows for three columns, as shown in Table 7-2.

Table 7-2 Generated columns by the row generator with rules

Column	Name	TYPE	Generation rules
1	KEY	INTEGER, NOT NULL	type=cycle, initial value = 0, increment = 1
2	NAME	CHAR(25)	algorithm=alphabet, String=abcdefghijklmnopqrstuvwxyz
3	COMMENT	VARCHAR(152)	algorithm=alphabet, String=abcdefghijklmnopqrstuvwxyz

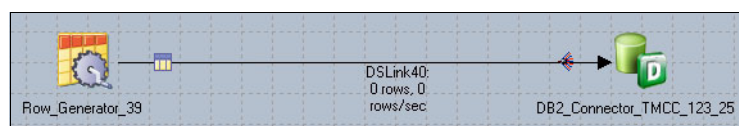


Figure 7-13 Sample job with generator and insert into AOT

The connector works in Write Mode Insert and generates the **INSERT** statement for a table named STKNOL.SAMPLEAOT2. Quoted identifiers must be disabled. The table action is **CREATE**, but we do not let the connector stage generate the table. Rather, use the Data Definition Language (DDL) shown in Example 7-7 to generate the accelerator-only table.

Example 7-7 Create accelerator-only table in a database

```
CREATE TABLE STKNOL.SAMPLEAOT2 (  
  KEY INTEGER NOT NULL,  
  NAME CHAR(25),  
  COMMENT VARCHAR(152)  
)  
IN DATABASE STKDB002  
IN ACCELERATOR STRIPER
```

To let the accelerator-only table be stored in a specific database, we use the **IN DATABASE** clause, and we also specify a hardcoded accelerator name in the **IN ACCELERATOR** clause. This can be made dynamically by referencing job parameters if necessary.

The record count for the transaction option can remain at 2000 or another number, but the array size setting for the session option *must* be set to 1. Otherwise the **INSERT** statement is generated as a multi-row insert.

It is also necessary to specify the following **Before SQL (node)** statement:

```
SET CURRENT QUERY ACCELERATION = ALL
```

All other settings can remain unchanged. Find an overview of the settings in Figure 7-14.

Connection [Associated data connection: BOECCTMCC12325DB2Con]	
Usage	
Write mode *	Insert
Generate SQL	Yes
Table name *	STKNOL SAMPLEAOT2
Enable quoted identifiers	No
Prefix for expression columns *	EXPR
SQL	
Table action *	Create
Generate create statement at runtime	No
Fail on error	Yes
Create statement *	CREATE TABLE STKNOL SAMPLEAOT2 (KEY INTEGER NOT ...
Generate drop statement at runtime	Yes
Generate truncate statement at runtime	Yes
Transaction	
Record count	2000
Session	
Isolation level *	Read uncommitted
Auto commit mode	On
Array size	1
Schema reconciliation	
Fail on row error	Yes
Insert buffering	Default
Logging	
Before/After SQL	Yes
Before SQL	
After SQL	
Before SQL (node)	SET CURRENT QUERY ACCELERATION = ALL
Read Before SQL (node) statement from file	No
Fail on error	Yes
After SQL (node)	
Reoptimization	None
Lock wait mode	Use the lock timeout database configuration parameter
Pad character	
Limit parallelism	No

Figure 7-14 Connector configuration for insert into AOT

It is also possible to use the write mode Delete then Insert and change the table action to Append. The AOT must be created by a database administrator (DBA) once, and the operations from DataStage clean the table and insert new rows as needed.

Performance measurements using the job in Figure 7-13 on page 109 are listed in Table 7-3. The Row insert statements column represents the number of single-row insert statements that were run against the accelerator in total, because we use a single-row insert.

Table 7-3 Insert rows through data stage sample job into an AOT with single row insert

Row insert statements	Rows inserted per statement	Time taken
1000	1	4 min 38 sec
2000	1	9 min 20 sec
1	10	Fails with SQL -4742, reason 22, requires multi-row insert

As a general rule, use this method to fill data into small lookup tables, but do not perform mass inserts.

The single-row inserts also show up in the query monitor section within the accelerator management view of IBM Data Studio.

Updating data in an AOT

It is also possible to update values in an AOT with this mechanism. We use a job that is similar to the one in Figure 7-13 on page 109. The row generator should have different generation rules like listed in Table 7-4 to validate the updates late on.

Table 7-4 Generator setting for AOT update example

Column	Name	TYPE	Generation rules
1	KEY	INTEGER, NOT NULL	type=cycle, initial value = 0, increment = 1
2	NAME	CHAR(25)	algorithm=alphabet, String=0987654321jihgfedcba
3	COMMENT	VARCHAR(152)	algorithm=alphabet, String=0987654321jihgfedcba

The settings of the connector must be different too. The write mode is Update then insert and the DML statement can be generated. The table action is Append and the **Before SQL (node)** statement must also have a special register setting.

Figure 7-15 lists all settings of the connector to run the operation.

Figure 7-15 Connector settings for update operation to AOT

Performance measurements about the updates are listed in Table 7-5. The Row update statements column reflects the number of update statements that have been run against the accelerator. Remember that we run each update as a single statement because it might become an insert if the update fails.

Table 7-5 Updated rows through data stage sample job into an AOT with single row update.

Row update statements	Rows updated per statement	Time taken
1000	1	5 min 25 sec
2000	1	10 min 29 sec
1	2000	06 sec

Preferably use one statement to update many rows rather than multiple statements updating one row only.

7.3.4 Accelerator maintenance through DataStage

The following section describes how to start accelerator maintenance operations to get information about the tables on the accelerator and loading data from DB2 to the accelerator.

ACCEL_LOAD_TABLES

To begin using the procedure to load data from DB2 to the accelerator, an ODBC connection to the mainframe, and a DB2 Connect connection must exist. See 7.3.2, “Things to consider for configuration” on page 101 to find information about how to set these up.

Before using the procedure, the procedure definition must be imported. To import the procedure definition, use the menu **Import** → **Table Definitions** → **Stored Procedure Definitions** and follow the dialogs. The import dialog uses the ODBC connection, as shown in Figure 7-16.

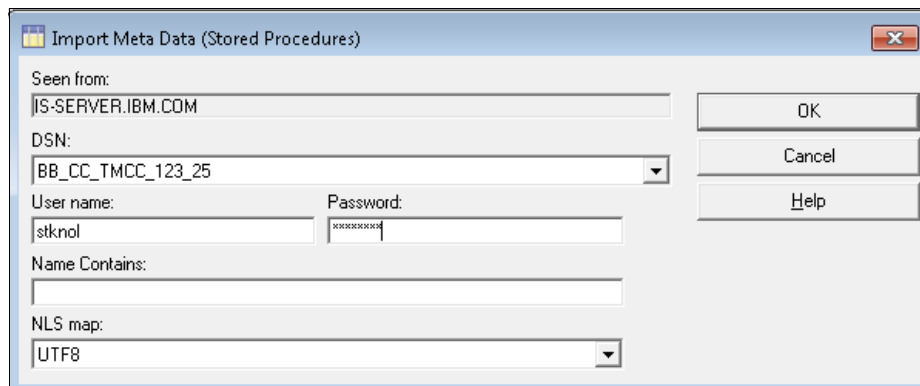


Figure 7-16 Procedure import dialog through ODBC connection

Without the ODBC connection, this step cannot be completed.

After selecting the procedure to import, in this case **SYSPROC.ACCEL_LOAD_TABLES**, the import dialog for the procedure requires the specification of values for the procedure. It is possible to specify the values for the individual parameters in a comma-separated list of quoted strings. In Figure 7-17 on page 114, the sample string in Example 7-8 is used.

Example 7-8 Argument list input for SYSPROC.ACCEL_LOAD_TABLES import

```
'STRIPER','NONE','<?xml version="1.0" encoding="UTF-8" ?><dwa:tableSetForLoad
xmlns:dwa="http://www.ibm.com/xmlns/prod/dwa/2011" version="1.2" ><table
name="REGION" schema="STKNOL"
forceFullReload="false"></table></dwa:tableSetForLoad>','<?xml version="1.0"
encoding="UTF-8" ?><aqttables:messageControl
xmlns:aqttables="http://www.ibm.com/xmlns/prod/dwa/2011" version="1.0"
versionOnly="true"></aqttables:messageControl>'
```

The accelerator for this sample is named STRIPER. The `versionOnly="true"` attribute of the message parameter prevents the actual execution of the procedure, but makes it return version information only. This execution always succeeds. It is only necessary for the procedure import.

Figure 7-17 shows the sample string from Example 7-8 on page 113 as the Argument list.

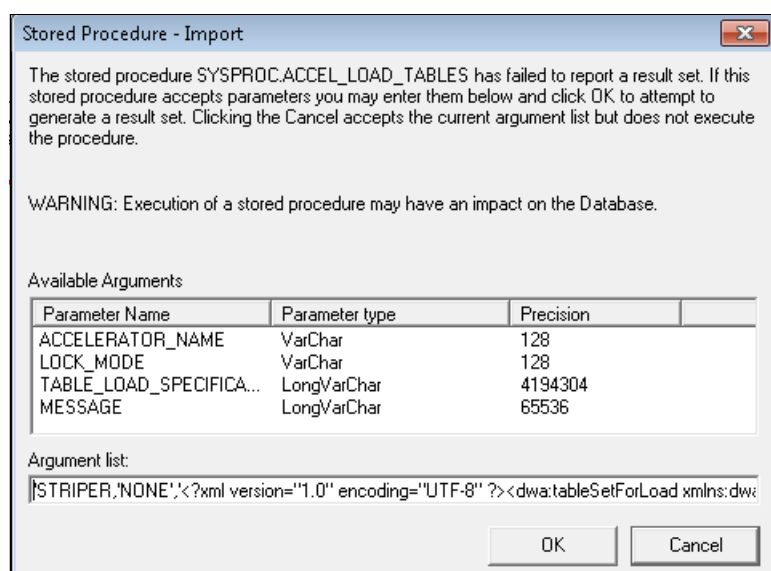


Figure 7-17 Parameter specification when importing stored procedures

Option1: Direct procedure call

To call the stored procedure without any further input, the procedure stage must be of type Source. Figure 7-18 lists a simple setup where the procedure output, in this case the status link information, is written to the job log. Because the load procedure does not list any information, the output should be checked for success or failures.

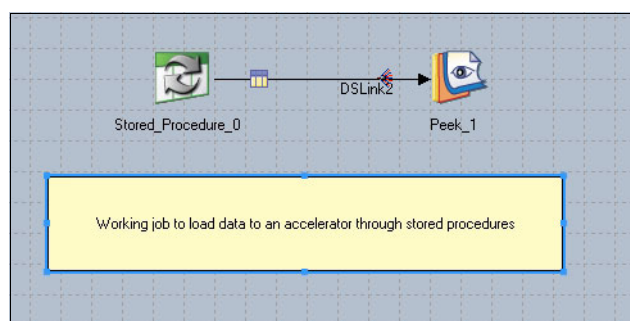


Figure 7-18 Use accelerator load procedure in a DataStage job

From a configuration perspective, DB2 as the database vendor must be configured for the procedure stage. The data source matches the value configured in DB2 Connect, as seen in Figure 7-19.

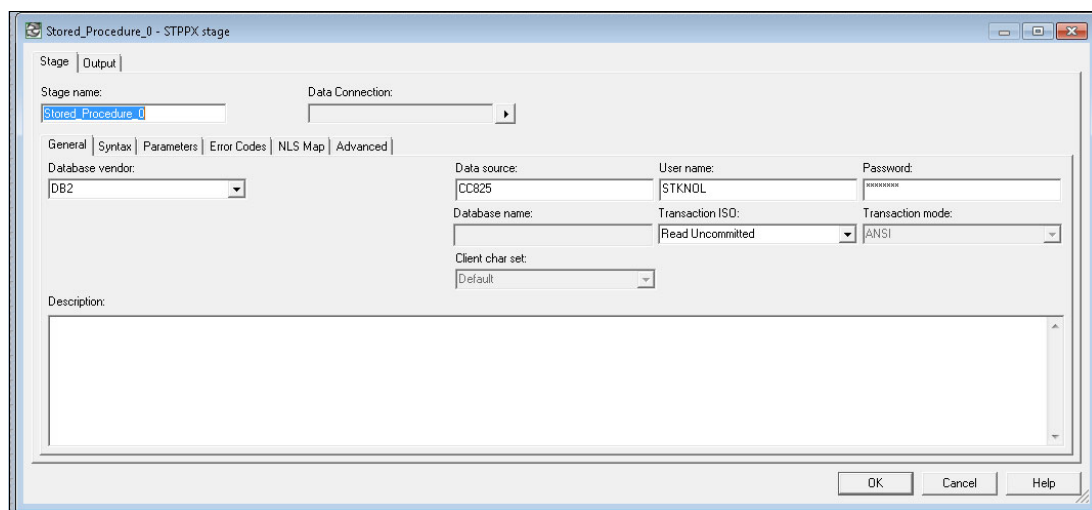


Figure 7-19 Specification of database and subsystem within the stored procedure stage

On the Syntax tab of the stage property, select the correct procedure name (**SYSPROC.ACCEL_LOAD_TABLES**). The procedure type must be Source. Clear the **Generate procedure call** option. It is possible to code the parameter values of the procedure call directly in the stage.

The call statement used in Figure 7-20 on page 116 ran a load against the accelerator named STRIPER with the lock mode of the procedure call set to NONE. Example 7-9 shows the complete procedure call with parameters.

Example 7-9 ACCEL_LOAD_TABLES call with parameters

```
CALL SYSPROC.ACCEL_LOAD_TABLES('STRIPER','NONE','<?xml version="1.0"
encoding="UTF-8" ?><dwa:tableSetForLoad
xmlns:dwa="http://www.ibm.com/xmlns/prod/dwa/2011" version="1.2" ><table
name="SUPPLIER" schema="STKNOL"
forceFullReload="false"></table></dwa:tableSetForLoad>',null);
```

The Extensible Markup Language (XML) text contains the table STKNOL.SUPPLIER to be loaded in this procedure invocation. By setting the last parameter to null, the procedure does not generate any trace information. It is possible to load multiple tables in one procedure call. To do that, repeat the <table> XML element in the call parameter of the procedure, as shown in Example 7-10. This loads two tables, STKNOL.SUPPLIER and STKNOL.REGION, in one procedure invocation.

Example 7-10 ACCEL_LOAD_TABLES call with parameters to load multiple tables

```
CALL SYSPROC.ACCEL_LOAD_TABLES('STRIPER','NONE','<?xml version="1.0"
encoding="UTF-8" ?><dwa:tableSetForLoad
xmlns:dwa="http://www.ibm.com/xmlns/prod/dwa/2011" version="1.2" ><table
name="SUPPLIER" schema="STKNOL" forceFullReload="false"></table><table
name="REGION" schema="STKNOL"
forceFullReload="false"></table></dwa:tableSetForLoad>',null);
```

By setting the `forceFullReload` flag, a complete table reload can be enforced. The following IBM Knowledge Center contains further information about the XML and the stored procedure `ACCEL_LOAD_TABLES`, including options to reload individual partitions of a table:

http://www.ibm.com/support/knowledgecenter/SS4LQ8_4.1.0/com.ibm.datatools.aqt.doc/sp_msg/SPs/sp_idaa_load_tables.html

It is important to have the table being loaded already added to the accelerator. This can be done through IBM Data Studio.

The call statement used in Figure 7-20 ran a load against the accelerator named `STRIPER` with the lock mode of the procedure call set to `NONE`.

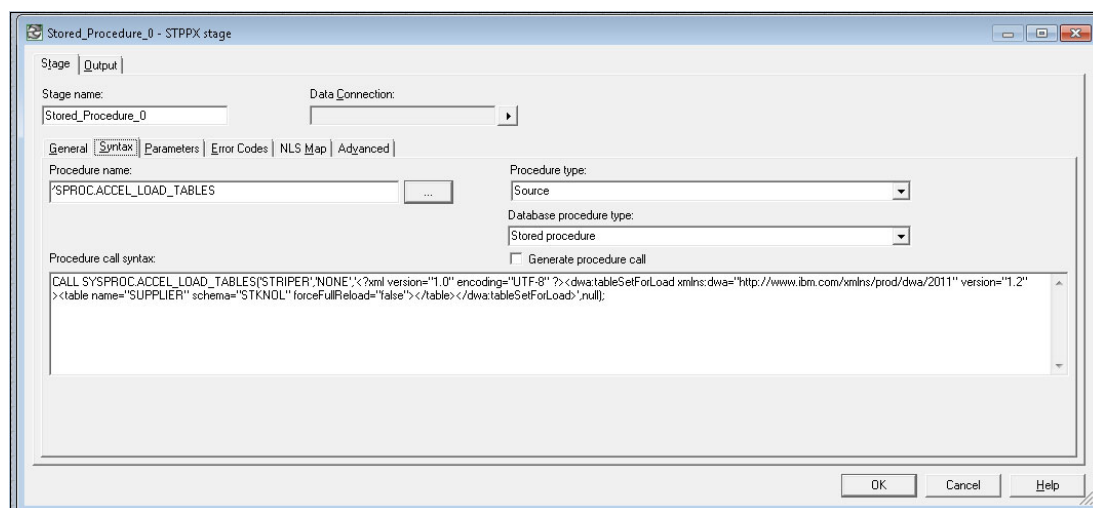


Figure 7-20 Specifying the parameters of the procedure call within the procedure stage

When checking the other tabs of the procedure stage dialog, the **Parameters** tab does not display the parameters of the procedure. This is because the stage type is set to **Source**. To get the return code of the procedure, the **Procedure status to link** option on the **Output** General tab should be selected. So the procedure stage has two output columns, named `ProcCode` and `ProcMess`. All other options of the stage can be left to their standard settings.

To test the procedure call, add a table to the accelerator through IBM Data Studio, but do not load it. Set up a **DataStage** job as described previously, and modify the XML input documents accordingly.

It might happen that the table does not get loaded. This typically happens when the XML input containing the table to load has a problem. To debug this problem, an alternative way of calling the procedure is available.

Option 2: Procedure call through helper table

To see the output messages of the procedure stage, and to be more flexible when calling the procedure, a second mechanism to load data to the accelerator exists. It includes a helper table that actually contains the parameters of the procedure call. But it also enables you to access the message parameter after the actual call. In case of errors, this parameter contains a description and the error message number.

The helper tables can look like Example 7-11, and contain the parameters of the procedure.

Example 7-11 Generation of helper table within DB2 for z/OS

```
CREATE DATABASE "STKDB003"
STOGROUP "SYSDEFLT"
CCSID UNICODE;

-- create table spaces (shortened)
CREATE TABLESPACE "ACCHL" IN "STKDB003" .....
CREATE LOB TABLESPACE "ACCHLX" IN "STKDB003" ....
CREATE LOB TABLESPACE "ACCHLX2" IN "STKDB003" ....

-- create tables
CREATE TABLE "STKNOL"."LOADCONTROL"
(
"ACCELERATOR_NAME" VARCHAR(128)NOT NULL,
"LOCK_MODE" VARCHAR(128)NOT NULL,
"TABLE_LOAD_SPECIFICATION" CLOB(4M)NOT NULL,
"MESSAGE" CLOB(64K)
)
IN "STKDB003"."ACCHL";

CREATE AUX TABLE "STKNOL"."LOADCONTROL_AUX_1"
  IN "STKDB003"."ACCHLX"
  STORES "STKNOL"."LOADCONTROL"
  COLUMN "TABLE_LOAD_SPECIFICATION";

CREATE INDEX "STKNOL"."LOADCONTROL_IDX1"
  ON "STKNOL"."LOADCONTROL_AUX_1";

CREATE AUX TABLE "STKNOL"."LOADCONTROL_AUX_2"
  IN "STKDB003"."ACCHLX2"
  STORES "STKNOL"."LOADCONTROL"
  COLUMN "MESSAGE";

CREATE INDEX "STKNOL"."LOADCONTROL_AUX_IDX2"
  ON "STKNOL"."LOADCONTROL_AUX_2";
```

Perform a select statement against the table to make sure that its creation succeeded. Expect no output but a successful execution of the query. After that, insert the tuple that contains the parameter for the load procedure call, as shown in Example 7-12.

Example 7-12 Insert rows into load helper table

```
INSERT INTO STKNOL.LOADCONTROL (ACCELERATOR_NAME, LOCK_MODE,
TABLE_LOAD_SPECIFICATION, MESSAGE)
VALUES ('STRIPER', 'NONE', '<?xml version="1.0" encoding="UTF-8"
?><dwa:tableSetForLoad xmlns:dwa="http://www.ibm.com/xmlns/prod/dwa/2011"
version="1.2" ><table name="SUPPLIER" schema="STKNOL"
forceFullReload="false"></table></dwa:tableSetForLoad>', null);
```

This example is the same input as shown in “Option1: Direct procedure call” on page 114. To store multiple tuples in the table, the DDL should be extended to contain a primary key. The key can be used later on in the DataStage job to select the appropriate tuple for the procedure call.

Before using the table in data stage, import the table definition by selecting **Import** → **Table Definitions** → **Start Connector Import Wizard**.

The DataStage job using this helper table needs to have a DB2 connector to get the tuple with the parameters from the helper table and feed them to the procedure stage. The combination of the connector and the procedure stage is shown in Figure 7-21.

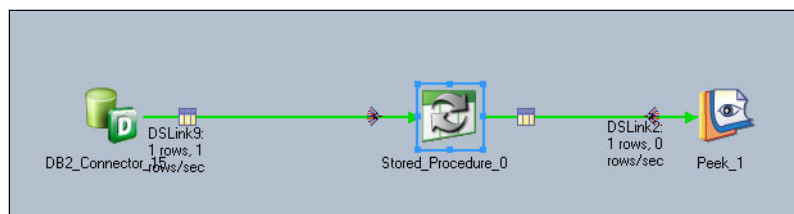


Figure 7-21 Helper table values feed parameter of the procedure stage

The connector stage can access the helper table, and must have the output columns of the helper table. If multiple tuples in the parameter table exist, a primary key column should be used to select the appropriate tuple. If this is not done, the connector stage might output all tuples, and the whole job can perform multiple procedure invocations within one job execution. This can be wanted behavior, but when loading multiple tables at the same time, this is not the best-performing option.

When using the procedure stage in this setup, the procedure type must be set to **Transform**. Select the **Generate procedure call** option so that the call statement contains placeholders for the parameters, as seen in Figure 7-22.

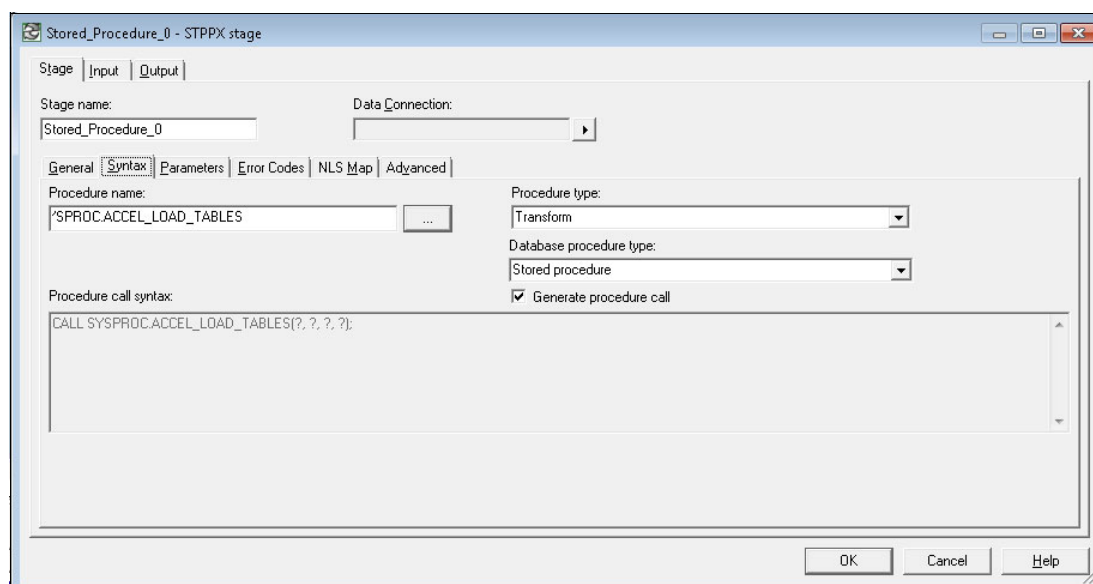


Figure 7-22 Procedure stage configured as Transform

When checking the other configuration tabs of the procedure stage dialog, the **Parameters** tab displays the parameters of the procedure. The parameters must be mapped to the input columns coming from the connector stage. Because the helper table columns have the same name as the parameters of the procedure, the column *Maps to Column* in Figure 7-23 on page 119 looks like redundant information, but it contains the input column names.

It is important to notice that the first three parameters of the procedure are input columns, while the **Message** parameter serves as an Input or Output parameter.

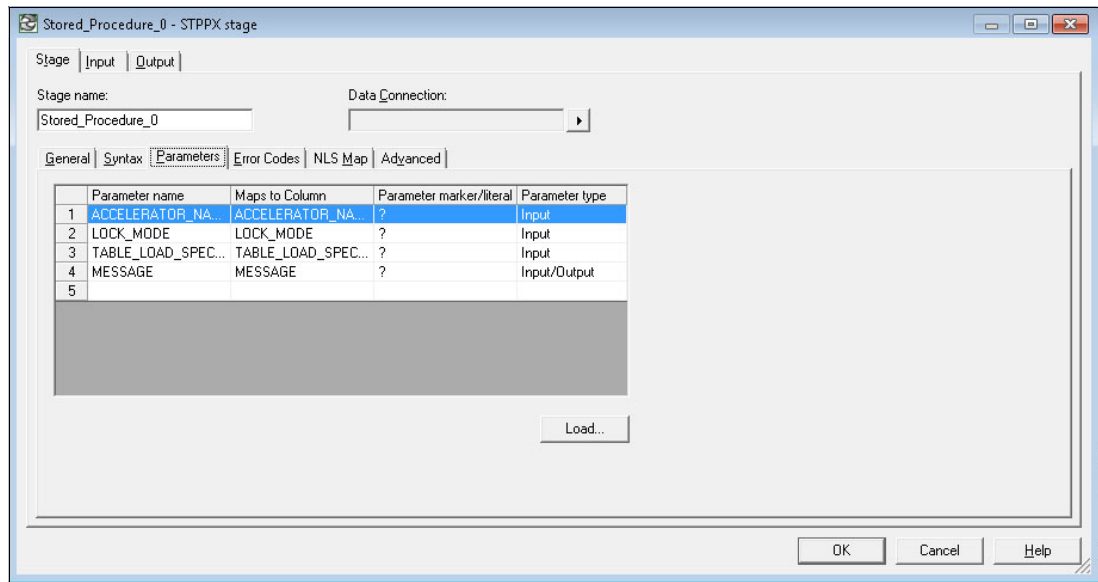


Figure 7-23 Parameter to input column mapping

To make the return code of the procedure available to following stages, select the **Procedure status to link** option on the Output > General tab. The procedure stage has multiple output columns. Two of them are named ProcCode and ProcMess, where the remaining columns are the same as the input columns. See Figure 7-24.

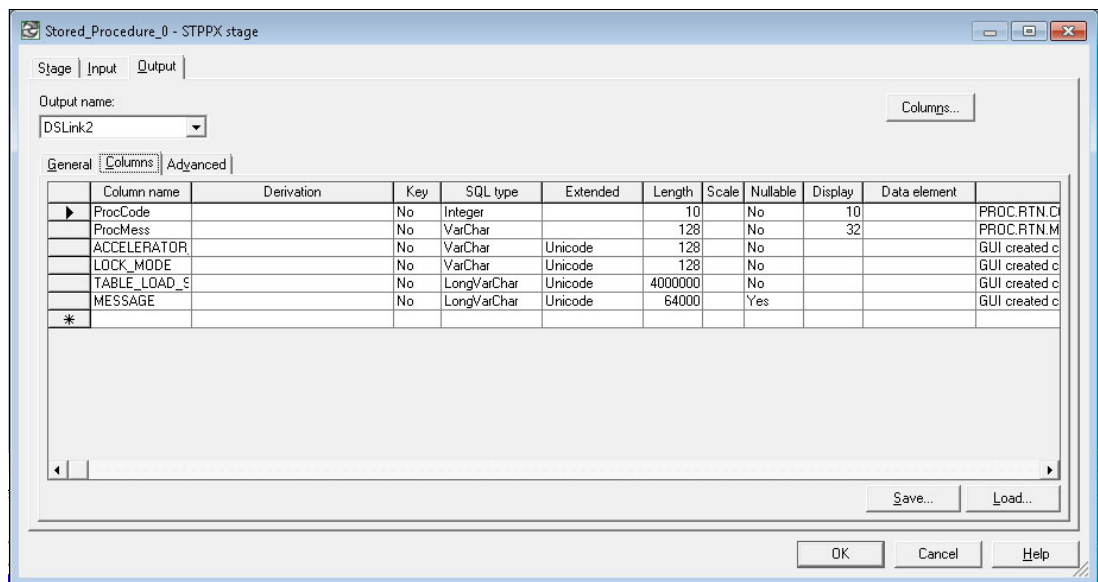


Figure 7-24 Output columns of a procedure stage of type Transform

Because the Message column is of type Input or Output, the column named Message will afterward contain the output for this parameter. In case of errors, there will be an error description in this column including an accelerator error message number.

There exist more options to call the procedure. It is also possible to use a row generator stage in combination with a transformer to generate the input for the stored procedure stage rather than using a helper table.

ACCEL_GET_TABLES_DETAILS

To begin using the stored procedure to get detailed information about an accelerator shadow table, an ODBC connection and a DB2 Connect connection to DB2 for z/OS must exist. See 7.3.2, “Things to consider for configuration” on page 101 to find information about how to set these up.

Import the stored procedure definition by using the menu **Import** → **Table Definitions** → **Stored Procedure Definitions** and follow the dialogs. The import dialog uses the ODBC connection, as shown in Figure 7-16 on page 113.

After selecting the procedure to import, in this case SYSPROC, the import dialog for the procedure requires the specification of values for the procedure. It is possible to specify the values for the individual parameters in a comma-separated list of quoted strings. In Figure 7-25, the sample string shown in Example 7-13 is used.

Example 7-13 Argument list input for SYSPROC.ACCEL_GET_TABLES_DETAILS import

```
'STRIPER','<?xml version="1.0" encoding="UTF-8"?><dwa:tableSet
xmlns:dwa="http://www.ibm.com/xmlns/prod/dwa/2011" version="1.0"><table
name="LINEITEM" schema="STKNOL"/></dwa:tableSet>', '<?xml version="1.0"
encoding="UTF-8" ?><aqttables:messageControl
xmlns:aqttables="http://www.ibm.com/xmlns/prod/dwa/2011" version="1.0"
versionOnly="true"></aqttables:messageControl>'
```

The accelerator for this sample is named STRIPER. The versionOnly="true" attribute of the message parameter prevents the actual execution of the procedure but makes it return version information only. This execution always succeeds. It is only necessary for the procedure import.

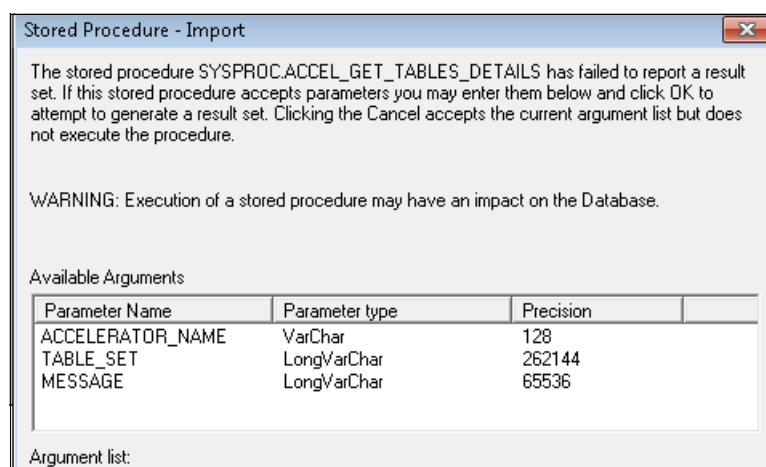


Figure 7-25 Import ACCEL_GET_TABLES_DETAILS

Option 1: Calling the stored procedure statically

The first option to call the procedure statically is to make the procedure stage of type Source. This enables the static specification of the call statement for the procedure inside the procedure stage. A sample showing how to do this is available in “Option1: Direct procedure call” on page 114. The disadvantage of this method is that the parameters of the procedure are not usable as output to other stages.

When the **ACCEL_GET_TABLES_DETAILS** lists table information, we must be able to access the output of the procedure call. Rather than hardcoding the call statement, we generate the input for the procedure stage, which enables us to configure the procedure stage to be of type Transform. By doing this, access to the result set of the procedure is possible. Figure 7-26 shows how a sample job to do this might look. The output of the procedure is “peeked” to the job log, but can also be input to other stages.



Figure 7-26 Generating the parameters for the **ACCEL_GET_TABLES_DETAILS** call

The row generator can be configured to generate one or more rows, although one is the value used in the following description. The output column of the generator stage should map to the parameters of the **ACCEL_GET_TABLES_DETAILS** procedure with the following settings:

- ▶ **ACCELERATOR_NAME** (VarChar, Unicode, length 128, nullable)
- ▶ **TABLE_SET** (LongVarChar, Unicode, length 262144, nullable)
- ▶ **MESSAGE** (LongVarChar, Unicode, length 65536, nullable)

The values being generated by the row generator do not have to follow any logic, because they are transformed in the following transformer stage. As seen in Figure 7-27 on page 122, the **MESSAGE** parameter can be set to null and the accelerator name is fed with the according value. For the **TABLE_SET** parameter, the following string value can be set in the derivation section:

```
<?xml version="1.0" encoding="UTF-8"?><dwa:tableSet
xmlns:dwa="http://www.ibm.com/xmlns/prod/dwa/2011" version="1.0"><table
name="LINEITEM" schema="STKNOL"/></dwa:tableSet>
```

DataStage escapes the double quotation marks in that string automatically. But it also generates another double quotation mark and parenthesis around the given string. Remove them, or the call to the procedure will have invalid XML. The correct escaped string must look like Example 7-14.

Example 7-14 Escaped string with XML input to stored procedure

```
"<?xml version=":'":'1.0':"'":' encoding=":'":'UTF-8":'":'?"><dwa:tableSet
xmlns:dwa=":'":'http://www.ibm.com/xmlns/prod/dwa/2011":'":'
version=":'":'1.0":'":' "><table name=":'":'LINEITEM":'":'
schema=":'":'STKNOL":'":'"/></dwa:tableSet>"
```

The string is double quoted at the start and the beginning, but all other double quotation marks within are escaped with ":'":'".

Example 7-14 returns the information for table STKNOL.LINEITEM only. It is possible to list more tables by adding the other tables in additional `<table />` XML elements.

Figure 7-27 shows the **MESSAGE** parameter value of SetNull().

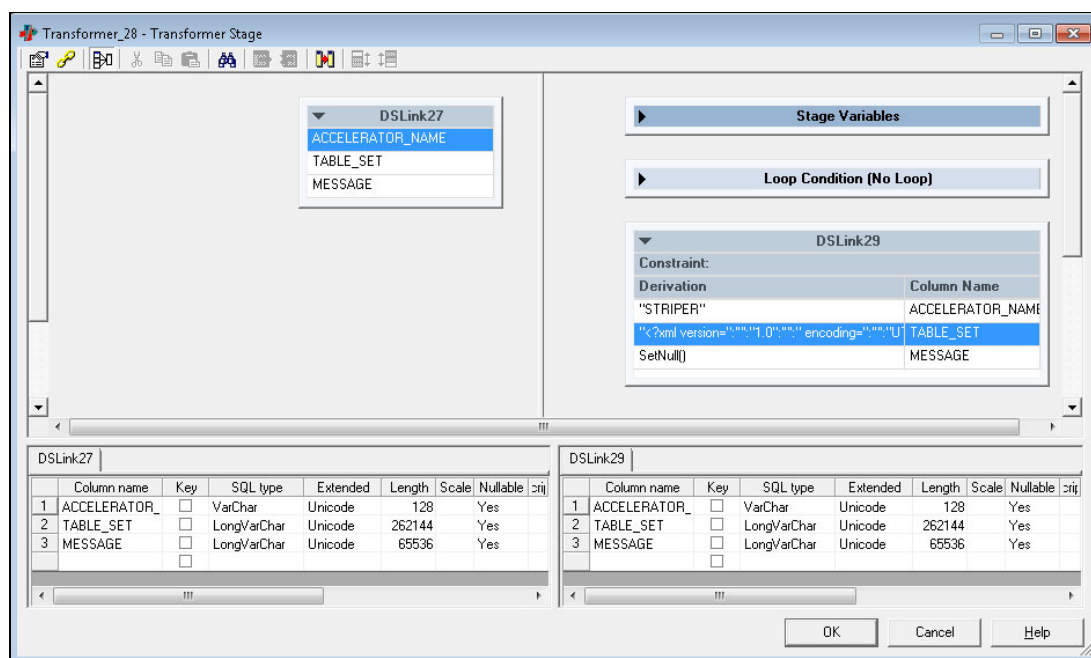


Figure 7-27 Derivation of procedure parameter input for ACCEL_GET_TABLES_DETAILS

Within the procedure stage, the database vendor of course must be DB2, and the user name and password must be given.

Complete the following steps:

1. On the Syntax tab, select the Procedure name and set the Procedure type to **Transform** to access the result set returned by the procedure. Let the stage generate the procedure call, as shown in Figure 7-28.

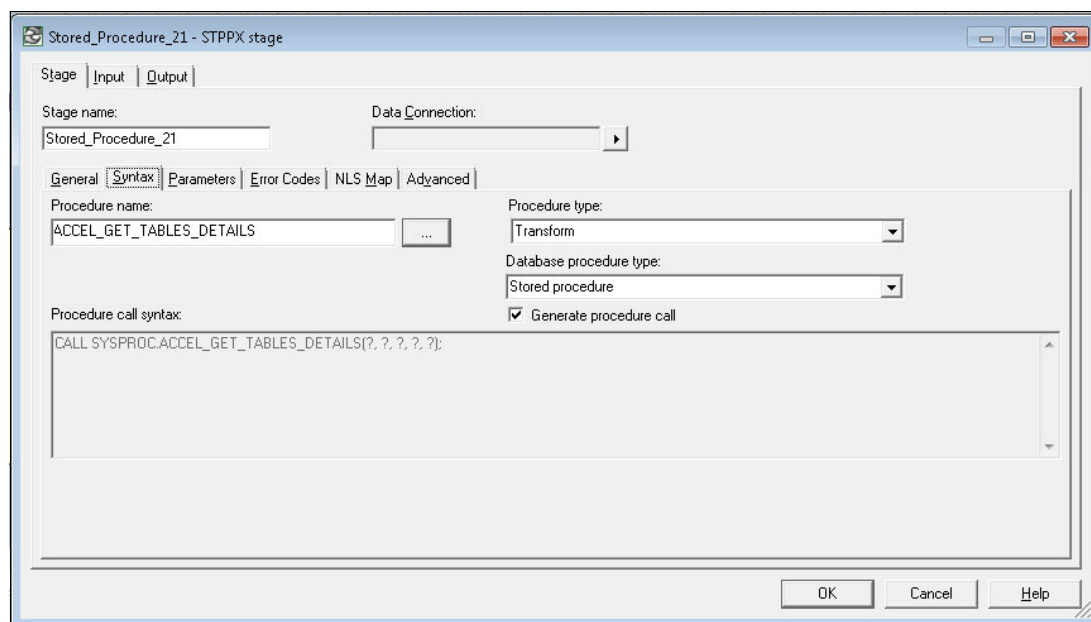


Figure 7-28 Syntax definition of procedure stage for ACCEL_GET_TABLES_DETAILS

- On the Parameters tab of the procedure stage, map the parameter names of the procedure to the input columns coming from the Transformer stage. In addition, two more output parameters are necessary.
- The **SEQID** and the **TABLES_DETAILS** parameters, which must map to an output column and the parameter type needs to be set to **CursorOutput**, as seen in Figure 7-29. The two parameters are the values of the result set returned by the stored procedure.

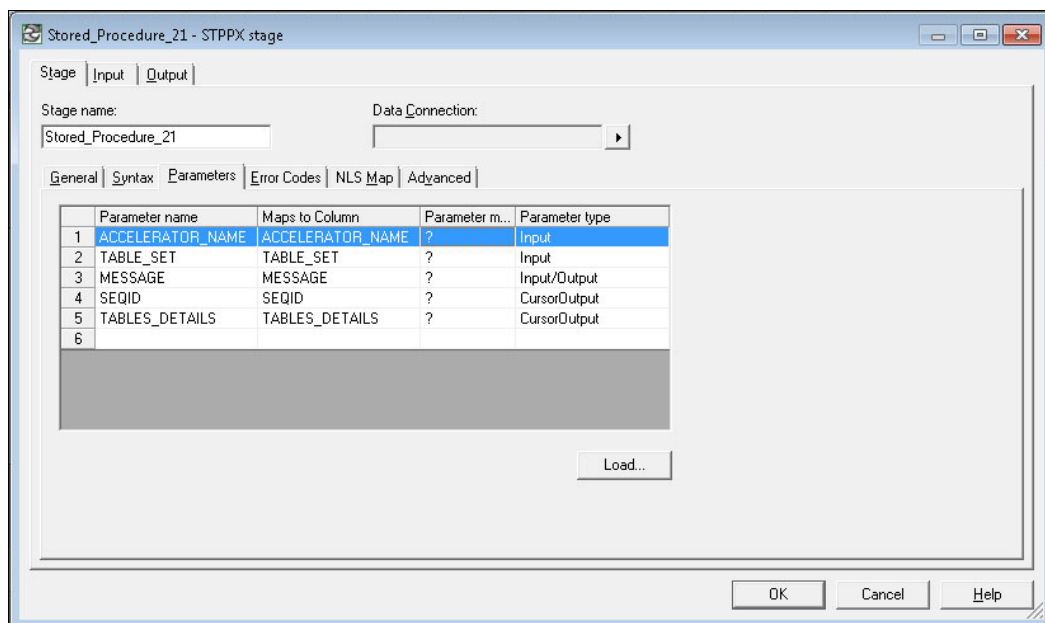


Figure 7-29 Additional parameters to fetch result set of procedure call

- On the General tab for the input columns, select the **Forward row data** options and **run procedure for each row**. For the General tab of the output columns, enable **Procedure status to link** and **Procedure returns multiple rows**. The last option for the output column is important if the result set contains multiple rows. All of them must be fetched to reconstruct the resulting XML text. The additional parameters, that were added in Figure 7-29 must map to additional output columns.

Those columns are displayed in Figure 7-30 on page 124 and must be added manually. The SEQID and TABLES_DETAILS columns are defined in the same way as for the global temporary table DSNAQT.ACCEL_TABLES_DETAILS. This temp table is filled during the execution of the ACCEL_GET_TABLES_DETAILS procedure and contains the resulting XML document in multiple rows.

By adding the output columns and mapping them to the parameters, it is possible to process the output of the procedure after the execution. SEQID needs to be of type integer and can be null, where TABLES_DETAILS is of type LongVarChar with length of 32698 and can be nullable as well.

Figure 7-30 shows the output columns.

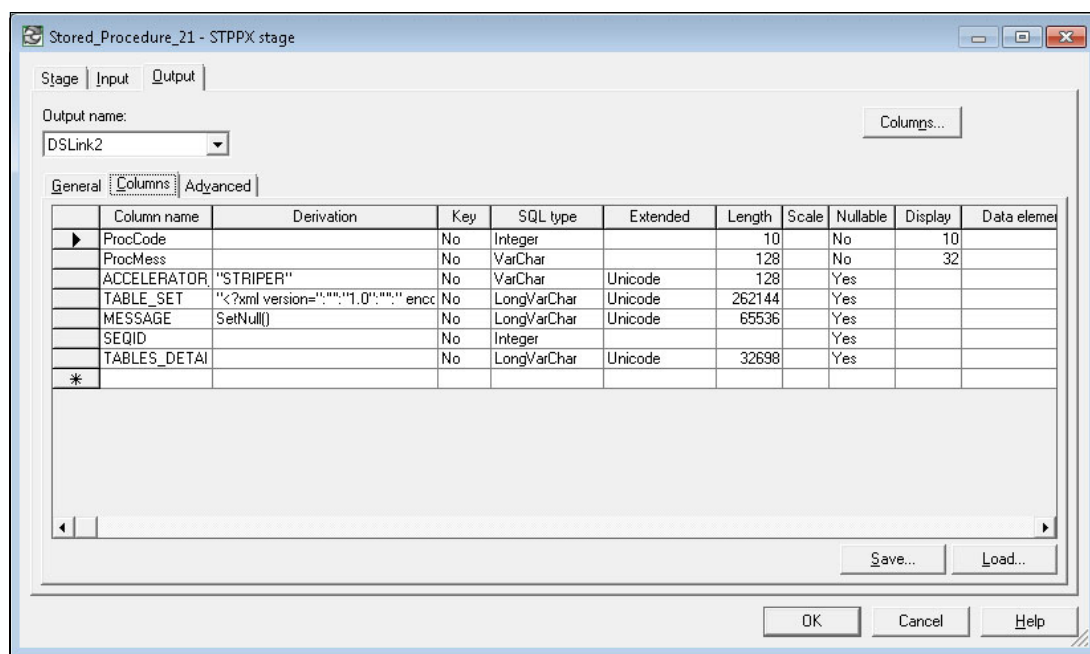


Figure 7-30 Output column including the table details information for a table

The peek stage of the job must peek all input columns so that you can be sure that all of the rows of the result set are fetched.

When running the job, the results of the procedure call are available through the TABLES_DETAILS column, as shown in Figure 7-31 on page 125. Feeding the column into another Transformer stage enables the concatenation of the individual lines to one big text document that can be parsed later on.

When running this sample job, and depending on your InfoSphere Information Server configuration, an error occurs as described in “Parallel jobs fail with “the record is too big to fit in a block” error” on page 144.

If the output column does not contain readable data or no data at all, check the text that is given in the MESSAGE column after the execution of the procedure. There is likely something wrong with the input. Only of the message columns contains the text AQT10000I, the execution completed successfully.

Log Event Detail		
Server:	IS-SERVER.IBM.COM	
User:	dsadm	
Job Name:	Job003_DB2Con_GetTablesDetails	
Invocation:		
Event type:	Info	
Project:	SampleProject	
Timestamp:	8/6/2015 5:37:57 PM	
Job Number:	20	
Wave:	1	Event Number: 63
Message Id:	IIS-DSEE-TOPK-00003	
<pre> Peek_1.0: ProcCode:0 ProcMess: ACCELERATOR_NAME:STRIPER TABLE_SET:<?xml version="1.0" encoding="UTF-8"?><dwa:tableSet xmlns:dwa="http://www.ibm.com/xmlns/prod/dwa/2011" version="1.0"><table name="LINEITEM" schema="STKNOL"/></dwa:tableSet> MESSAGE:<?xml version="1.0" encoding="UTF-8" ?><dwa:messageOutput xmlns:dwa="http://www.ibm.com/xmlns/prod/dwa/2011" version="1.0"> <message severity="informational" reason-code="AQ110000I"><text>The operation was completed successfully.</text><description>Success message for the XML MESSAGE output parameter of each stored procedure.</description><action></action></message></dwa:messageOutput> SEQID:0 TABLE_DETAILS:<?xml version="1.0" encoding="utf-8" standalone="no" ?><aqt:tableSetDetails xmlns:aqt="http://www.ibm.com/xmlns/prod/dwa/2011" version="1.0" xmlns:dwa="http://www.ibm.com/xmlns/prod/dwa/2011"><table name="LINEITEM" schema="STKNOL"><changeInformation category="NONE" dataSizeInMB="236" lastLoadTimestamp="2015-07-20T09:49:19.796704Z" sharedTablespace="false" type="NoChange" /></table></aqt:tableSetDetails> </pre>		

Figure 7-31 Output of stored procedure call

Option 2: Procedure call through helper table

The second option to call the **ACCEL_GET_TABLES_DETAILS** procedure is by using a helper table that contains the input parameter rather than a generator and transformer stage. The helper table should contain the necessary columns for the procedure invocation (Example 7-15).

Example 7-15 DDL for helper table to call **SYSPROC.ACCEL_GET_TABLES_DETAILS**

```

CREATE DATABASE "STKDB003" STOGROUP "SYSDEFLT" CCSID UNICODE;

-- create table spaces
CREATE TABLESPACE "ACCGTD" IN "STKDB003" ....
CREATE LOB TABLESPACE "ACCGTDX" IN "STKDB003" ...
CREATE LOB TABLESPACE "ACCGTDX2" IN "STKDB003" ....

-- create tables
CREATE TABLE "STKNOL"."GETTABLESCONTROL"
(
  "ACCELERATOR_NAME" VARCHAR(128) NOT NULL,
  "TABLE_SET" CLOB(256K) NOT NULL,
  "MESSAGE" CLOB(64K)
)
IN "STKDB003"."ACCGTD";

CREATE AUX TABLE "STKNOL"."GETTABLESCONTROL_AUX_1"
  IN "STKDB003"."ACCGTDX"
  STORES "STKNOL"."GETTABLESCONTROL"
  COLUMN "TABLE_SET";

CREATE INDEX "STKNOL"."GETTABLESCONTROL_IDX1"
  ON "STKNOL"."GETTABLESCONTROL_AUX_1";

CREATE AUX TABLE "STKNOL"."GETTABLESCONTROL_AUX_2"
  IN "STKDB003"."ACCGTDX2"
  STORES "STKNOL"."GETTABLESCONTROL"
  COLUMN "MESSAGE";

CREATE INDEX "STKNOL"."GETTABLESCONTROL_AUX_IDX2"
  ON "STKNOL"."GETTABLESCONTROL_AUX_2";

```

Perform a **SELECT** statement against the table to make sure that its creation succeeded. Expect no output but a successful execution of the query. After that, insert the tuple that contains the parameter for the procedure call, as shown in Example 7-16.

Example 7-16 Inserting the tuple

```
INSERT INTO STKNOL.GETTABLESCONTROL (ACCELERATOR_NAME, TABLE_SET, MESSAGE) VALUES
('STRIPER', '<?xml version="1.0" encoding="UTF-8"?><dwa:tableSet
xmlns:dwa="http://www.ibm.com/xmlns/prod/dwa/2011" version="1.0"><table
name="LINEITEM" schema="STKNOL"/></dwa:tableSet>', null);
```

This example provides the XML to get the tables details of table STKNOL.LINEITEM. To store multiple tuples in the table, the DDL should be extended to contain a primary key. The key can be used later in the DataStage job to select the appropriate tuple for the procedure call.

Before using the table in data stage, import the table definition through **Import → Table Definitions → Start Connector Import Wizard**.

The DataStage job using this helper table needs to have a DB2 connector to get the tuple with the parameters from the helper table and feed them to the procedure stage. The combination of the connector and the procedure stage is shown in Figure 7-32.

The connector stage can access the helper table, and must have the output columns of the helper table. If multiple tuples in the parameter table exist, a primary key column should be used to select the appropriate tuple. If this is not done, the connector stage might output all tuples, and the whole job can perform multiple procedure invocations within one job execution. This can be wanted behavior, but when loading multiple tables at the same time, this is not the best-performing option.

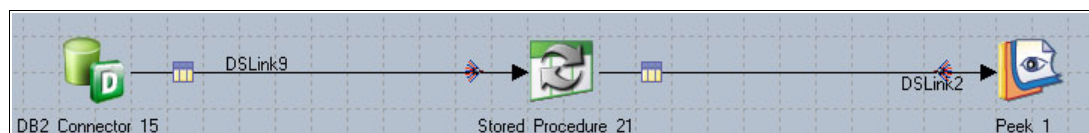


Figure 7-32 Connector to access helper table and feed the procedure stage

This setup is similar to the approach listed in “Option 2: Procedure call through helper table” on page 125 of “ACCEL_LOAD_TABLES” on page 113.

The output columns of the connector are input to the procedure stage, where the database vendor and the appropriate procedure must be selected. The type of the procedure must be Transform and the call must be generated. Extensions to the parameters of the procedure must be made to include the SEQID and TABLES_DETAILS columns of type CursorOutput.

When running this sample job, and depending on your InfoSphere Information Server configuration, an error occurs as described in “Parallel jobs fail with “the record is too big to fit in a block” error” on page 144.

If the output column does not contain readable data or no data at all, check the text that is given in the MESSAGE column after the execution of the procedure. There is likely something wrong with the input. Only if the message columns contains the text AQT10000I, the execution completed successfully.

ACCEL_GET_TABLES_INFO

The stored procedure ACCEL_GET_TABLES_INFO returns two result sets. The first one contains the tables definition and the first result set contains the columns of DSNAPT.ACCEL_TABLES_INFO_SPEC. The second result set contains the tables status and contains the columns of DSNAPT.ACCEL_TABLES_INFO_STATES.

When working with this stored procedure in DataStage, only the rows of the first result are fetchable, even if multiple parameters for the various result sets are specified. To circumvent this, a wrapper procedure that returns the table's status information as the first result set is needed. The wrapper procedure should have same parameters as the ACCEL_GET_TABLES_INFO procedure.

The procedure can be generated from this sample through a SQL script in IBM Data Studio. Use the DDL shown in Example 7-17.

Example 7-17 DDL for SQL wrapper stored procedure

```
CREATE PROCEDURE STKNOL.ACCEL_GET_TABLES_INFO_STATUS_WRAPPER(  
  IN ACCELERATOR_NAME VARCHAR(128),  
  IN TABLE_SET CLOB(256K),  
  INOUT MESSAGE CLOB(64K))  
  VERSION V1  
  LANGUAGE SQL  
  ALLOW DEBUG MODE  
  PARAMETER CCSID UNICODE  
  ISOLATION LEVEL CS  
  DYNAMIC RESULT SETS 1  
  SP: BEGIN  
  
  DECLARE X CURSOR WITH RETURN TO CALLER FOR  
    SELECT SEQID, TABLE_STATES FROM DSNAPT.ACCEL_TABLES_INFO_STATES;  
  
  CALL SYSPROC.ACCEL_GET_TABLES_INFO(ACCELERATOR_NAME, TABLE_SET, MESSAGE);  
  OPEN X;  
  
END SP  
@
```

The statement terminator must have the At sign (@).

After creating the wrapper procedure, import its definition into DataStage by selecting **Import** → **Table Definitions** → **Stored Procedure Definitions** and following the dialog steps. When being prompted for the argument list of the procedure, use the following concatenated string, shown in Example 7-18.

Example 7-18 Argument list input for wrapper stored procedure import

```
'STRIPER','<?xml version="1.0" encoding="UTF-8"?><dwa:tableSet  
xmlns:dwa="http://www.ibm.com/xmlns/prod/dwa/2011" version="1.0"><table  
name="LINEITEM" schema="STKNOL"/></dwa:tableSet>', '<?xml version="1.0"  
encoding="UTF-8" ?><aqttables:messageControl  
xmlns:aqttables="http://www.ibm.com/xmlns/prod/dwa/2011" version="1.0"  
versionOnly="true"></aqttables:messageControl>'
```

STRIPER is the name of accelerator used in the example.

After having the procedure definition imported, a job can make use is created that uses the helper stored procedure. As shown in Figure 7-33, the job can use a row generator and a transform to feed the input to the wrapper stored procedure.

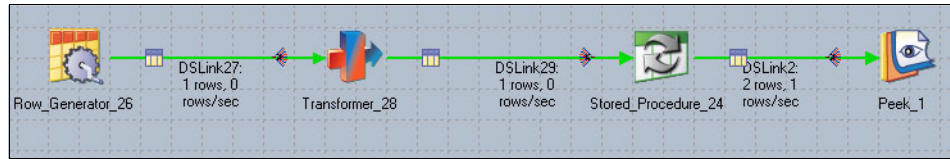


Figure 7-33 Calling the wrapper stored procedure to get tables status

The row generator has provided three output parameters that can be set in the transformer stage. The `ACCELERATOR_NAME` is set to a fixed string of the accelerator being queried, and the remaining parameters both can be set to null, as shown in Figure 7-34. This returns information about all tables that are currently on the accelerator.

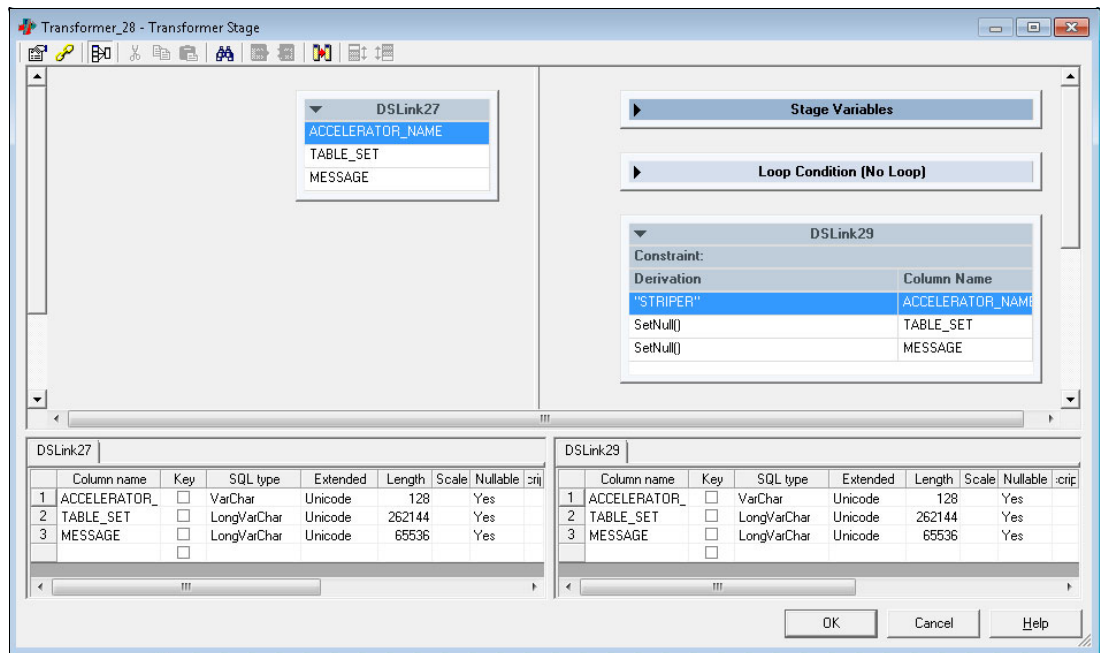


Figure 7-34 Modifying the column values to valid stored procedure input

The procedure stage uses the DB2 database vendor and the wrapper stored procedure and needs to be of procedure type Transform. The call can be generated, and the parameters of the procedure stage must be mapped to the input columns. As shown in Figure 7-35 on page 129, the `SEQID` and `TABLE_STATE` parameter must be introduced as Parameter Type `CursorOutput` and mapped to two additional output columns of the procedure stage.

On the General tab of the input columns, select **Forward row data** and for the output column, select **Status to link** and **Procedure returns multiple rows**. Especially the last option is necessary, because an accelerator can have many tables, and the result set certainly contains multiple rows.

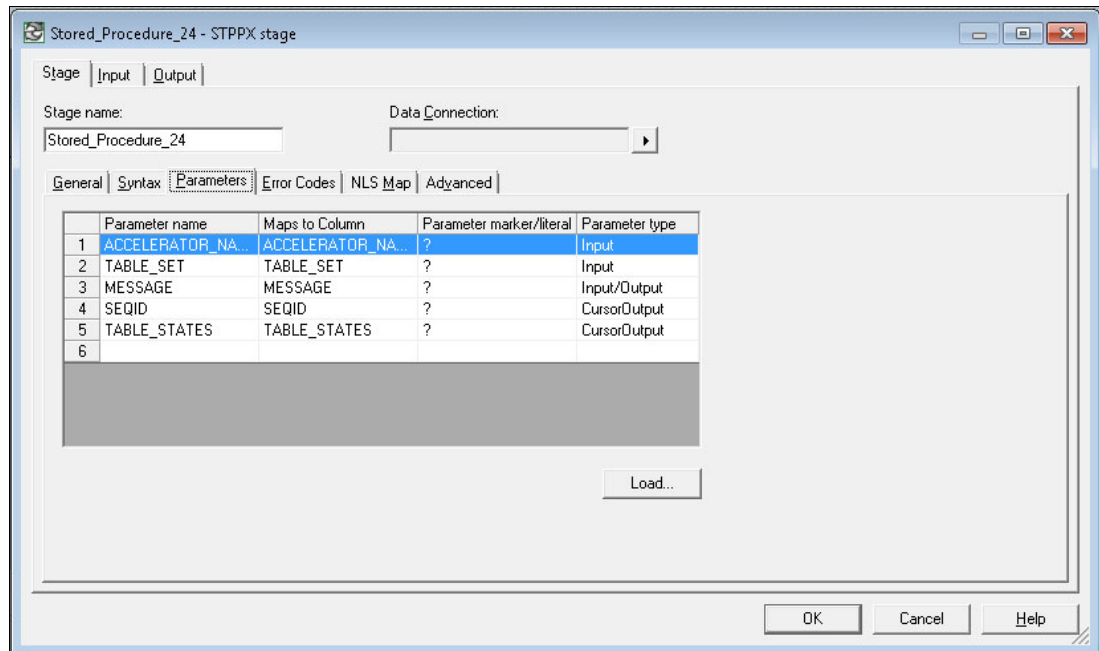


Figure 7-35 Adding parameters to fetch the table status properly

The additional output column must match the definition of `DSNAQT.ACCEL_TABLES_INFO_STATES` where the `SEQID` is of type Integer and the `TABLE_STATES` is a LongVarChar with 32698 length. The definition of those columns is depicted in Figure 7-36.

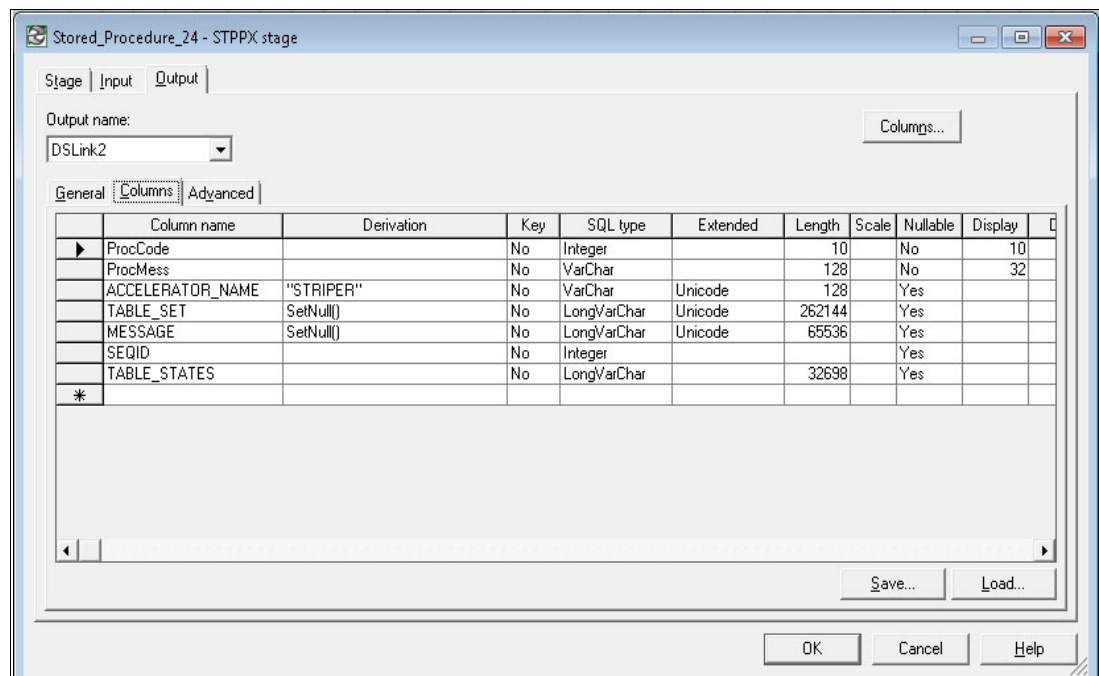


Figure 7-36 Output columns to access the table states

When running this sample job, and depending on your Information Server configuration, an error as described in “Parallel jobs fail with “the record is too big to fit in a block” error” on page 144.

If the output column does not contain readable data, or contains no data at all, check the text that is given in the MESSAGE column after the execution of the procedure. There is likely something wrong with the input. Only if the message columns contains the text AQT10000I, the execution completed successfully.

After a successful execution, the job log (Figure 7-37) might contain multiple rows that in concatenation of the TABLE_STATES columns form the XML document returned by the wrapper stored procedure for ACCEL_GET_TABLES_INFO.

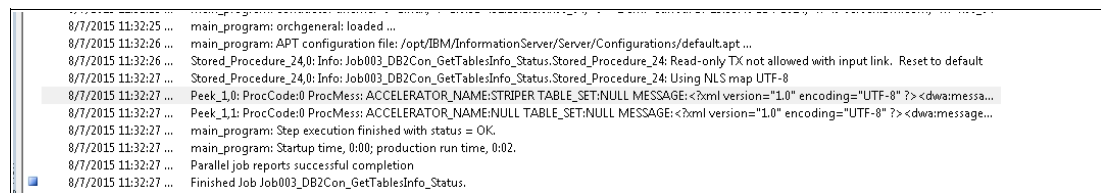


Figure 7-37 Table status peeked to job log

7.3.5 Optimizing existing DataStage jobs

The following section describes a sample ETL process that creates a data mart out of operational data. The example is based on table definitions following the TPC-H schema (<http://www.tpc.org/tpch/>). The amount of data is roughly 1 Gigabyte (Gb), and was generated with the **dbgen** tool of the TPC-H suite. The data mart is taken from the definition of the star schema benchmark, as described in Revision 3 from June 5th, 2009 (published by Prof. Patrick O’Neil, Department of Math and C.S, UMass/Boston):

<http://www.cs.umb.edu/~poneil/StarSchemaB.PDF>

ETL example description

The following description summarizes a fictional business scenario:

A data mart is needed by a line of business to generate various reports (for example, sales or inventory analysis). The report generation involves complex SQL that affects daily business when being run on operational data. In addition, costs for storing the data mart had to be low, so the data mart tables are stored in a different database management system after transformation. Because of the amount of data, it takes a couple of hours to produce the data mart.

The process to build the data mart already exists as DataStage jobs.

The line of business stated the requirement to have more up-to-date information for their reports, but cannot afford a high investment of many hours to change the report-generating queries. In the future, it is planned to rework the queries generating the reports, but more up-to-date reports need to be available today.

The IT department has an accelerator that serves a different line of business today.

It is possible to make the source tables of the scenario available as accelerator shadow tables, but this immediately involves changing the reporting queries. From a staged approach, sticking to the mart-generating process and optimizing the existing data stage jobs using AOTs is possible.

To demonstrate this fictional scenario, eight tables (see Figure 7-38) from the TPC-H schema are used as sources for the ETL process. They contain the operational data.

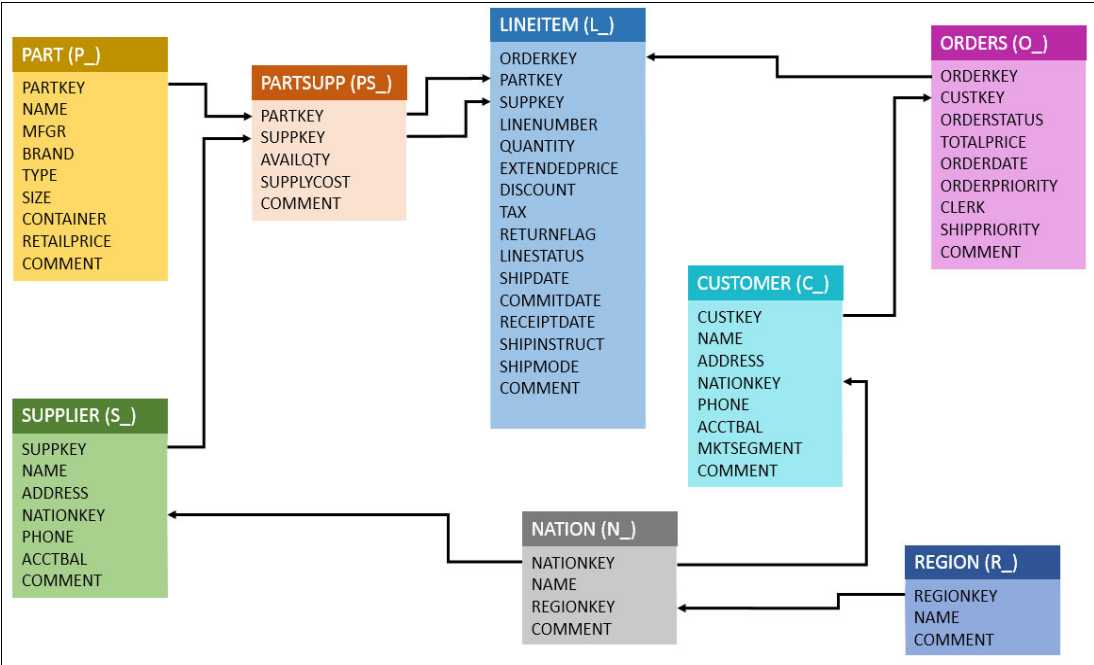


Figure 7-38 Source tables for sample ETL process, based on TPC-H

The actual row counts of the tables are shown in Table 7-6, and represent 1 GB of data.

Table 7-6 RowCount information about source tables of sample ETL process

Table	RowCount
PART	200,000
SUPPLIER	10,000
PARTSUPP	800,000
CUSTOMER	150,000
ORDERS	1,500,000
NATION	25
REGION	5
LINEITEM	6,001,215

During the ETL process, the data is transformed to a star schema representation (Figure 7-39 on page 132) to form a data mart. The CUSTOMER, PART, and SUPPLIER tables become dimensions to a fact table that combines LINEITEM and ORDERS. In addition, a new dimension table called DATE is generated based on data information that was present in the ORDERS and LINEITEMS table before.

The dimension tables also contain new derived columns like COLOR, CATEGORY, or CITY, which are marked in red. The colors of the newly generated tables are the same as in Figure 7-38. If tables are prejoined to build the tables, the newly included columns are marked in the same color as the source tables shown in Figure 7-38.

Figure 7-39 shows the data transformed to a star schema representation.

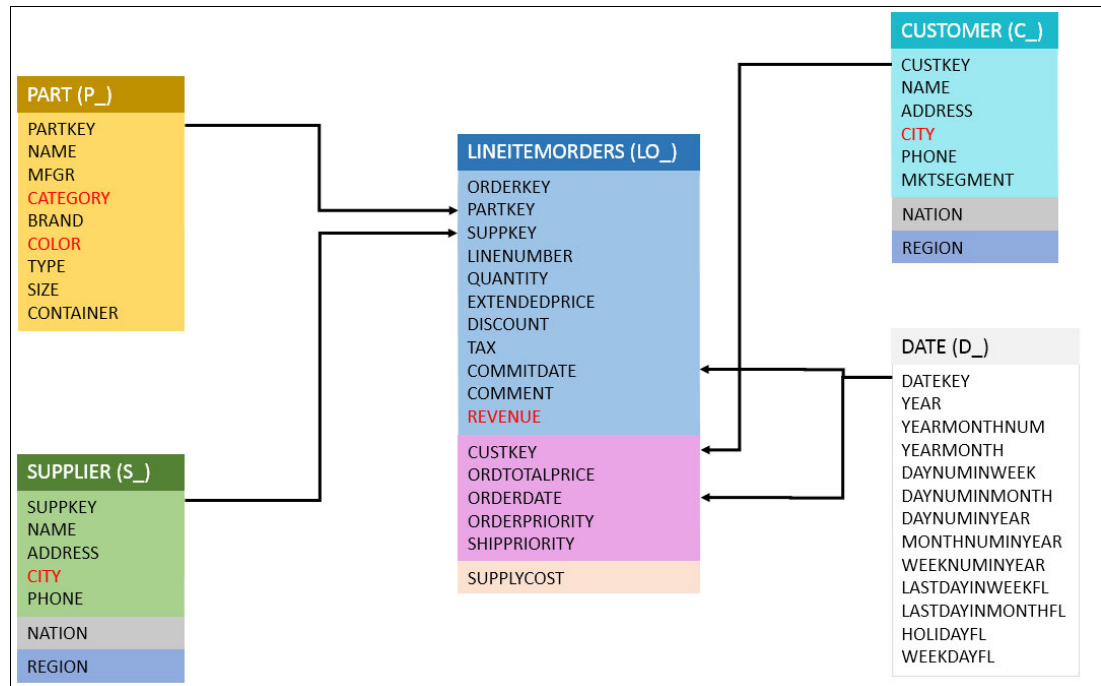


Figure 7-39 Star schema representation with colored column origin information

Transformation job recommendation

The transformation of such data can be done within one DataStage job, as seen on Figure 7-40 and Figure 7-41 on page 133.

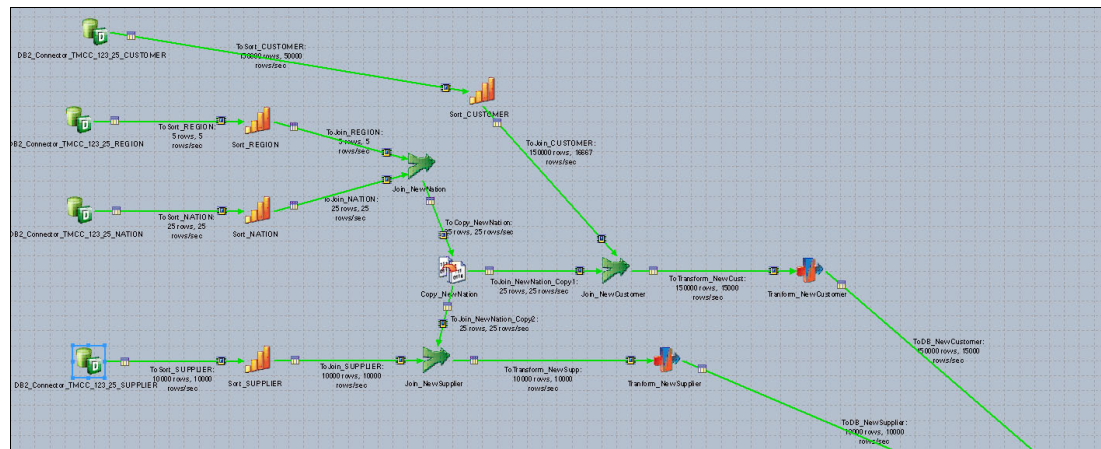
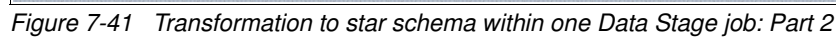


Figure 7-40 Transformation to star schema within one Data Stage job: Part 1



Note: Try to use lightweight jobs with fewer stages if possible. Concatenate them in a job sequence. Using Balanced Optimization on these jobs has a higher chance to successfully transform them into an *in-database transformation* job where everything is done within the accelerator. Accelerator-only tables can be used as staging tables between these jobs if necessary.

Figure 7-42 Sample job sequence for data mart construction

Chapter 7. In-database transformations **133**

This might be fast when not optimizing the jobs and using the accelerator. To use the full power of Balanced Optimization and AOTs in combination with the accelerator, DataStage should write the results to a target table on the accelerator. This supports pushing the transformation process to the accelerator, and avoids long-lasting data extractions from the accelerator to the Information Server system.

Note: To use in-database transformation as efficient as possible, make the target of the transformation a DB2 table (which can be optimized to an accelerator-only table) to push transformation logic (SQL) into the accelerator.

When trying to use the accelerator's in-database transformation, the shown job sequence can remain unchanged. Except for adding a parameter set to the job properties that enables specification of the accelerator name. Modification might become necessary within the jobs that are triggered in the sequence. The jobs are basically a subset of the logic shown in Figure 7-40 on page 132 and Figure 7-41 on page 133.

Using Balanced Optimization on job chain

The necessary precondition to work with Balance Optimization at all, is that tables being used as the source for a transformation must be on the accelerator. As accelerator shadow tables, they must be added, loaded, and maintained regularly.

Building and optimizing the SUPPLIER dimension table

The star schema dimension table SUPPLIER is formed by joining two other tables and deriving additional column values. Figure 7-43 shows the flow of the job that uses transformer stage to produce the derivation of an added column name CITY, which is a subset of the value in the NAME column of the NATION table.

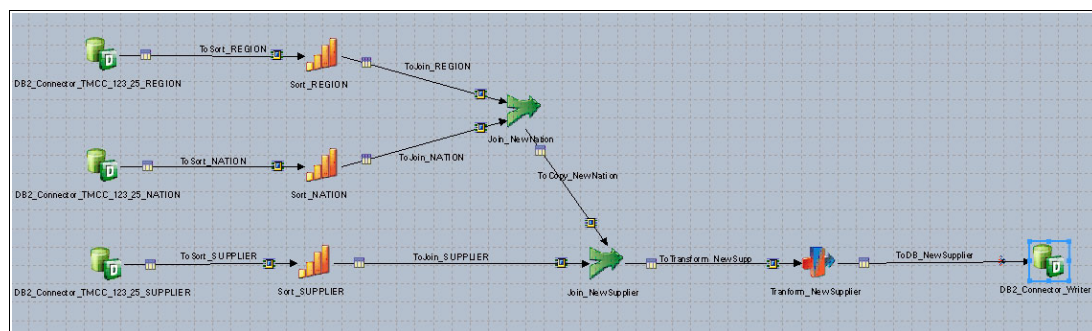


Figure 7-43 Building the supplier dimension table

The target of the transformation is a table to enable DataStage to optimize the whole job. If the target were a file, for example, the job could still be optimized, but the performance benefit might be smaller. The target connector is configured for write mode Delete then insert and table action Append. For table action Create or Truncate, DataStage generates a second table that will affect the optimized behavior, too.

Note: Do not let the target database stage generate the output table (either a DB2 table or an accelerator-only table). Generate the table through a DDL statement.

DataStage generates a second temporary table named DB2CCTEMP_.... to perform environment and schema reconciliation. When using AOT, the target database stage generates a second accelerator-only table that follows the naming schema of the temporary table (DB2CCTEMP_....). The temporary table must be cleaned up manually.

In addition, the target database connector is configured to work in execution mode `Sequential` to avoid deadlocks on the target table when multiple nodes insert data with large transactions.

To start Balanced Optimization, select **File** → **Optimize** and select DB2 and the accelerator as the target database (Figure 7-44). Make sure that the table name in the target database connector is the name of an existing accelerator-only table. A detailed description about the options is available on the following IBM Knowledge Center:

https://www.ibm.com/support/knowledgecenter/SSZJPZ_11.3.0/com.ibm.swg.im.iis.ds.parc.job.dev.doc/topics/balancedoptimizationoptions.html

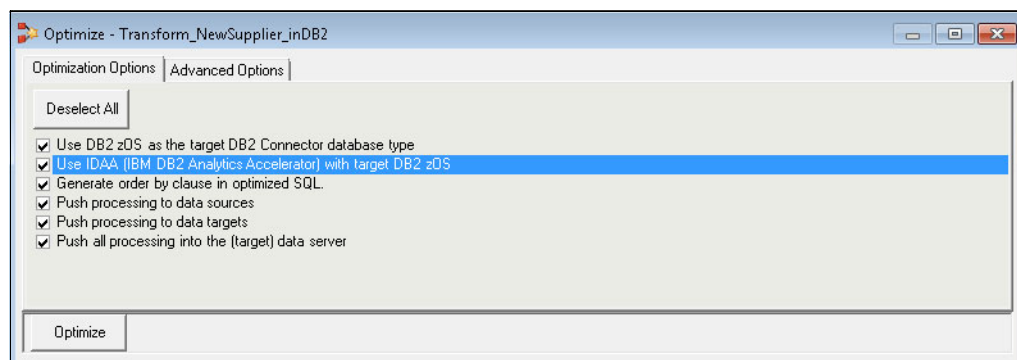


Figure 7-44 Balanced Optimization configuration for accelerator usage

After clicking the **Optimize** button, a new job is generated, which ideally looks like the one in Figure 7-45, where no additional stages exist. The initiator does not generate any rows, but triggers the remaining connector, which contains the generated SQL statement as an **After SQL** option.

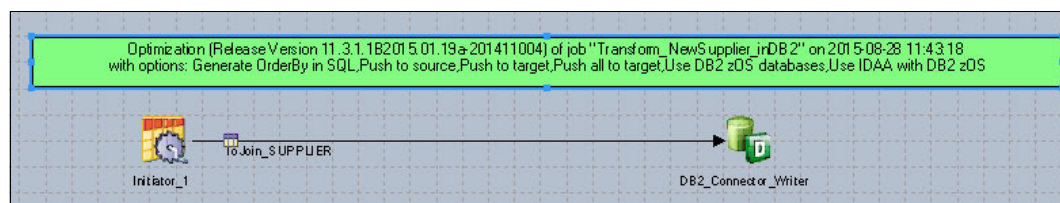


Figure 7-45 Balanced optimized job

The generated SQLs start with the specification of the special register to run the statement on the accelerator. By default, this is set to `ALL`, so the statement is forced to run on the accelerator:

```
SET CURRENT QUERY ACCELERATION ALL;
```

When using the write mode `Delete` then `insert` in the source job, the generated SQL contains a **DELETE** statement next to remove existing rows, as shown in Example 7-19.

Example 7-19 Generated **DELETE** statements for the accelerator-only table

```
delete from STKNOL.NEWSUPPLIER_AOT
where
exists (select 1
from
(select S_SUPPKEY as NS_SUPPKEY, S_NAME as NS_NAME, S_ADDRESS as NS_ADDRESS,
SUBSTR(N_NAME,1,10) as NS_CITY, N_NAME as NS_NATIONNAME, R_NAME as NS_REGIONNAME,
S_PHONE as NS_PHONE
```

```

from
(select BALOP_3.S_SUPPKEY as S_SUPPKEY, BALOP_3.S_NAME as S_NAME,
...

```

Because the delete statement contains a subselect, it can become large. It is an option to shorten the delete statement manually, and drop the subselect if all data should be deleted:

```
delete from STKNOL.NEWSUPPLIER_AOT;
```

The next part contains the **INSERT FROM SELECT** statement (which forms the transformation logic (Example 7-20). It uses subselects and becomes large, too.

Example 7-20 Generated INSERT FROM SELECT statement for the accelerator-only table

```

insert into STKNOL.NEWSUPPLIER_AOT (NS_SUPPKEY, NS_NAME, NS_ADDRESS, NS_CITY,
NS_NATIONNAME, NS_REGIONNAME, NS_PHONE)
select CAST(BALOP_26.NS_SUPPKEY AS INTEGER) as NS_SUPPKEY, CAST(BALOP_26.NS_NAME
AS CHARACTER(25)) as NS_NAME, CAST(BALOP_26.NS_ADDRESS AS VARCHAR(40)) as
NS_ADDRESS, CAST(BALOP_26.NS_CITY AS CHARACTER(10)) as NS_CITY,
CAST(BALOP_26.NS_NATIONNAME AS CHARACTER(25)) as NS_NATIONNAME,
CAST(BALOP_26.NS_REGIONNAME AS CHARACTER(25)) as NS_REGIONNAME,
CAST(BALOP_26.NS_PHONE AS CHARACTER(15)) as NS_PHONE
from
(select S_SUPPKEY as NS_SUPPKEY, S_NAME as NS_NAME, S_ADDRESS as NS_ADDRESS,
SUBSTR(N_NAME,1,10) as NS_CITY, N_NAME as NS_NATIONNAME, R_NAME as NS_REGIONNAME,
S_PHONE as NS_PHONE
from
...

```

The whole SQL is completed by a **COMMIT** statement:

```
COMMIT WORK
```

The optimized job also adds a parameter set named BalOpDB2z0SParams to the job parameters. When running the job, the parameter set asks for the IP address of the z/OS system and the accelerator name. When only running the **INSERT FROM SELECT** statement on the accelerator, the z/OS IP does not need to be specified. The accelerator name is only necessary when the table action is Create or Truncate. In that case, also see the following note.

Note: When setting the table action for optimized jobs to Create or Truncate, make sure that you add a database name to the **CREATE TABLE** statement of the accelerator-only table. Do this by adding the clause **IN DATABASE "<dbname>"** to the **CREATE TABLE** statement. Otherwise, an implicitly created database is used.

Also, consider adding the database name to the Balanced Optimization parameter set, and reference the parameter set value within the statement for more flexibility.

Building and optimizing the PART dimension table

When transformation the PARTS table to a dimension table, the transformer stage used a loop variable and a transformation function, when designed initially. The whole job is listed in Figure 7-46.

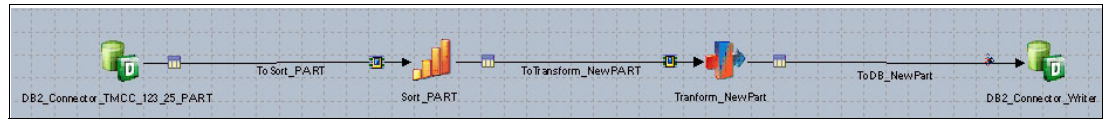


Figure 7-46 Job and stages to build the PART dimension table

The loop variable handles nulls within the P_NAME of the source table. The following transformation function is used to produce the equivalent in the dimension table:

```
NullToValue(ToTransform_NewPART.P_NAME,"unknown")
```

The derivations to fill the columns MFGR and COLOR use the following functions:

```
Change(ToTransform_NewPART.P_MFGR,"Manufacturer","MFGR")  
Field(ToTransform_NewPART.P_NAME,"",1)
```

Figure 7-47 shows the content of the transformer stage.

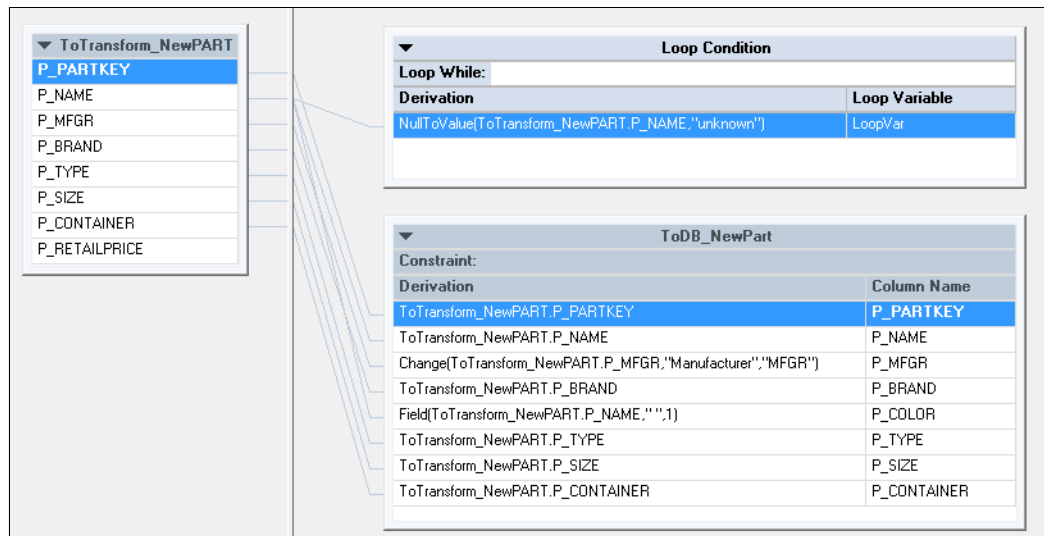


Figure 7-47 Transformation functions being used to produce PART dimension table

When trying to optimize this job, the transformer stage cannot be optimized into one SQL statement, and the transformer stage still exists. This situation leads an SQL that inserts data to an accelerator-only table as shown in Figure 7-48. The performance of this job is not the best.

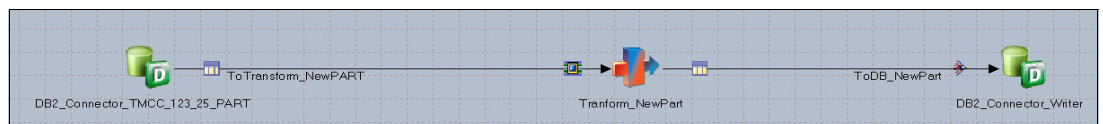


Figure 7-48 Incomplete optimized job due to derivation function usage

The usage of the `Change()` function prevents the optimization of the whole job and the removal of the transformer stage. An alternative solution might be to extract a substring and build a new concatenation:

```
"MFGR#" :Field(ToTransform_NewPART.P_MFGR,"#",2)
```

Because the `Field()` function uses parameter values higher than 1 as the last option, this function cannot be optimized. Therefore, the rewrite to the following statement is necessary to circumvent this situation:

```
"MFGR" :
Right(ToTransform_NewPART.P_MFGR,LEN(ToTransform_NewPART.P_MFGR)-LEN("Manufacturer
"))
```

Although the rewrite and usage of the `Right()` function does not exactly replace all occurrences of “Manufacturer” with “MFGR”, it serves the purpose for this job. In addition, this derivation is optimized through Balanced Optimization. The other functions being used, and the loop variable with its derivation, are no problem for the optimization process. The resulting job (Figure 7-49) does not contain any remaining stages.

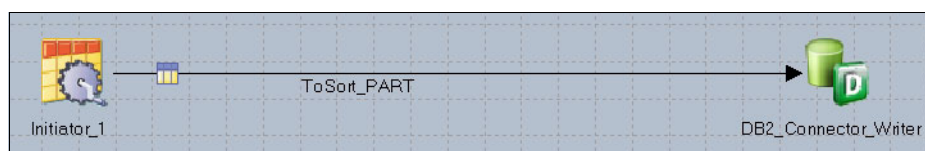


Figure 7-49 Optimized job for PART dimension

Note: When using functions within a stage, it is preferable to use the ones that support optimization through Balanced Optimization. The following overview lists many functions:

https://www.ibm.com/support/knowledgecenter/SSZJPZ_11.3.0/com.ibm.swg.im.iis.ds.parjob.dev.doc/topics/c_deeref_Functions_functions.html?lang=en

A couple of them cannot be optimized:

https://www.ibm.com/support/knowledgecenter/SSZJPZ_11.3.0/com.ibm.swg.im.iis.ds.parjob.dev.doc/topics/limitationsfunctions.html

The customer dimension table is built by joining three tables. Rather than doing this in one join step, multiple join steps are introduced (Figure 7-50). This configuration provides a complete optimization. The transformer stage is necessary to generate the CITY column by extracting a substring from the NAME column.

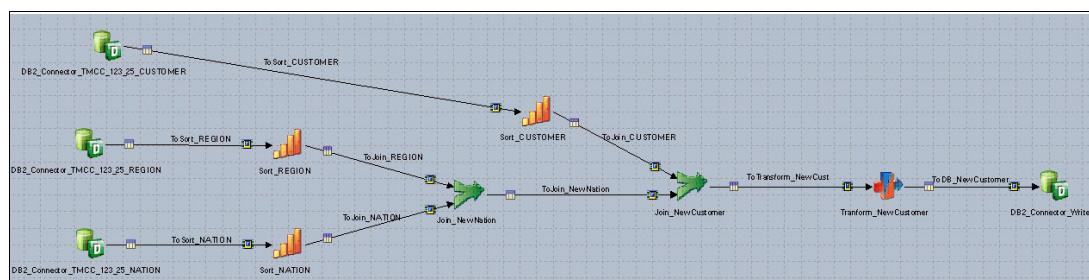


Figure 7-50 None optimized job to build CUSTOMER dimension

The optimization of the job is possible without reworking any of the stages.

Note: Try to work with two inputs into the join stage only. Optimization join stages with more input links will not be optimized.

Build and optimize the DATE dimension table

To build the DATE dimension table, the COMMITDATE column from table LINEITEM and ORDERDATE of table ORDERS are extracted and replace by a foreign key value to this dimension table. The values within the dimension table are derived from the actual time value, and cover information like the day number in the year or the day in a month. The job to do the construction of the dimension table is listed in Figure 7-51.

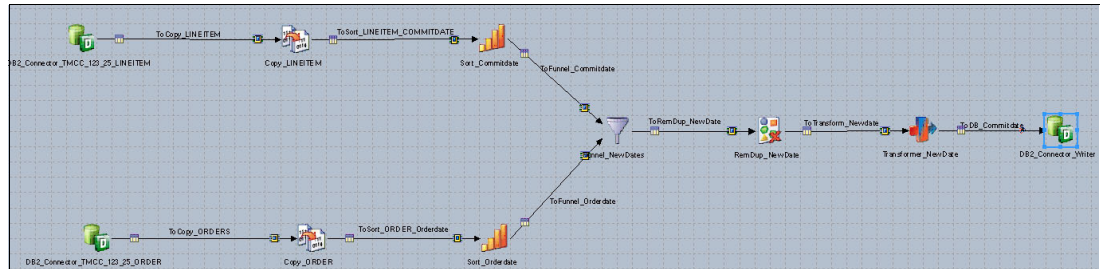


Figure 7-51 DATE dimension construction using funnel stage and a remove duplicate stage

A funnel stage is used to combine the date values contained in the two source tables, while the following transformer stage extracts the information about the date value. To funnel the date extracted from the LINEITEMS and ORDERS table, the sorts for the two tables do an implicit column renaming, as shown in Figure 7-52. In addition, the sort for the first table has the Output Statistics option set to true. It might not seem important, but it has a big effect on the generated SQL statement when optimized later.

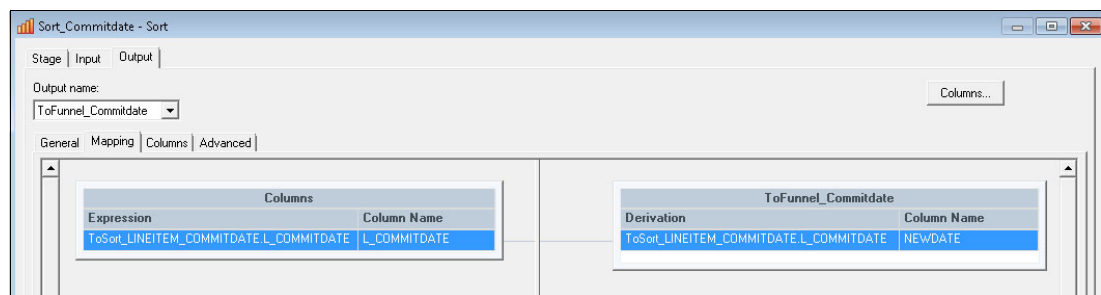


Figure 7-52 Input output column renaming done within sort stage

The transformer stage in Figure 7-51 only gets one input column, and uses a derivation function to build the various values of the DATE dimension. Not all columns can initially be generated easily, and are left blank or set to 0.

The functions being used can be seen in Figure 7-53.

ToTransform_Newdate	
NEWDATE	

ToDB_Commitdate	
Constraint:	
Derivation	Column Name
DateToDecimal(ToTransform_Newdate.NEWDATE, "%yyyy%mm%dd")	D_DATEKEY
YearFromDate(ToTransform_Newdate.NEWDATE)	D_YEAR
DateToDecimal(ToTransform_Newdate.NEWDATE, "%yyyy%mm")	D_YEARMONTHNUM
WeekdayFromDate(ToTransform_Newdate.NEWDATE, "Monday")	D_DAYNUMINWEEK
MonthDayFromDate(ToTransform_Newdate.NEWDATE)	D_DAYNUMINMONT
YeardayFromDate(ToTransform_Newdate.NEWDATE)	D_DAYNUMINYEAR
MonthFromDate(ToTransform_Newdate.NEWDATE)	D_MONTHNUMINYE
YearweekFromDate(ToTransform_Newdate.NEWDATE)	D_WEEKNUMINYEAF
If (WeekdayFromDate(ToTransform_Newdate.NEWDATE, "Monday") = 7) Then	D_LASTDAYINWEEK
If ((DaysInMonth(ToTransform_Newdate.NEWDATE) - MonthDayFromDate(ToT	D_LASTDAYINMONT
0	D_HOLIDAYFL
0	D_WEEKDAYFL

Figure 7-53 Date function used to populate the DATE dimensions columns

When trying to optimize the job, a couple of stages prevent that process, as seen in Figure 7-54. The first sort stage is not removed. This is caused because the sort stage Output statistics option is set to true.

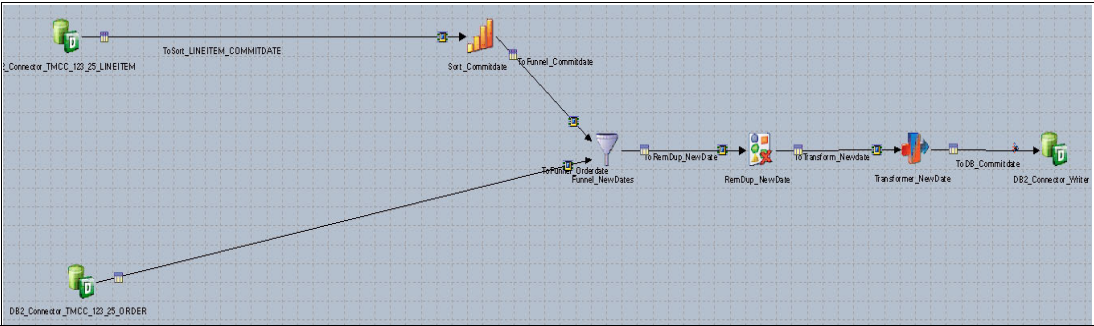


Figure 7-54 Poorly optimized job

To procedure the statistics, DataStage must do the sorting in application logic, because SQL does not provide the necessary information.

Note: Avoid statistics collection for sort stages to Balance Optimize them.

The next stage that prevents a complete optimization is the funnel. To remove the funnel stage, a join stage followed by a transformer stage can have the same effect. The join stage must do an outer join, which might introduce null values. The following transformer stage can use the function NullToValue() to copy output values that have been on the right side of the join but not in the left to the resulting set.

Figure 7-55 shows a portion of the function, which in total is the following function:
NullToValue(ToTransform_Union.left_NEWDATE, ToTransform_Union.rightRec_NEWDATE)

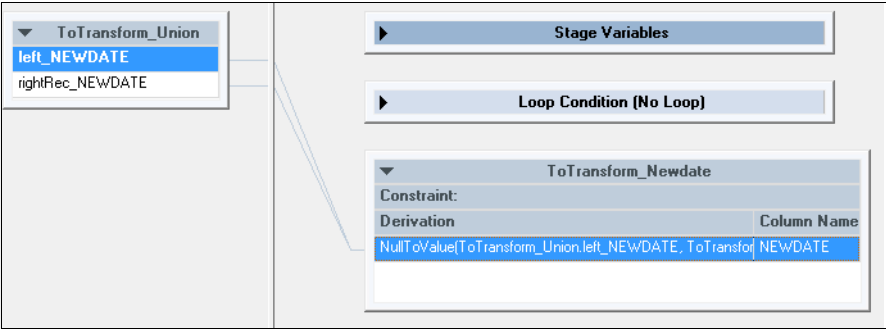


Figure 7-55 Null value elimination after full outer join

Note: During the creation of the DataStage jobs, it seemed like the funnel stage could not be optimized. In a later experiment, it was observed that funnel stages do get optimized to UNION expressions in SQL.

If funnel stages do not get optimized, determine whether previous stages prevent the optimization, or whether the stage has too many input links.

The original job contained one **remove duplicate** stage immediately after the funnel stage. As the funnel is replaced with a join stage, reducing the number of rows being joined can speed up the job execution. This is achieved by duplicating the **remove duplicate** stage and moving them just before the join stage after the sort stages. The final remodeled job looks like the one in Figure 7-56.

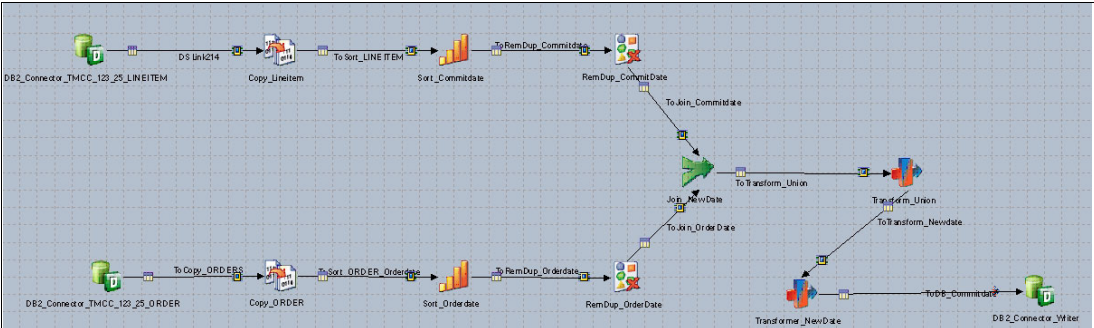


Figure 7-56 Remodeled job before Balanced Optimization

As said before, the implicit column renaming done within the sort stage causes problems within the optimized SQL statement.

The copy stages in front of the sort stage take care of the renaming, as seen in Figure 7-57, and introduce an additional subselect in the optimized SQL later. Optimization of the job is possible even without the copy stage, but when running the optimized statement, expect to receive SQL return with SQL code -206.

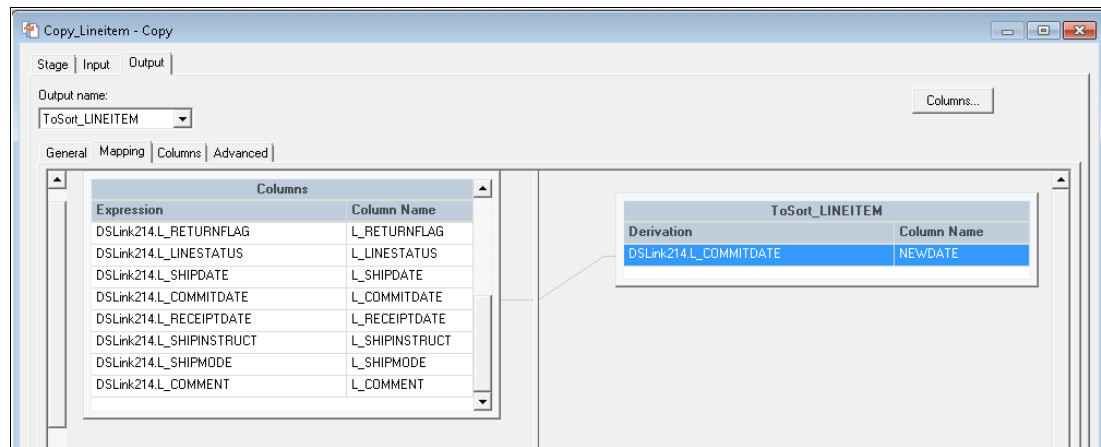


Figure 7-57 Renaming of column with the copy stage rather than the sort stage

The last stage that prevents a complete optimization is the last transformer stage, which uses the various derivation functions.

Because the `DateToDecimal()` function prevents optimization, the following expression can be replaced:

```
DateToDecimal (ToTransform_Newdate.NEWDATE, "%yyyy%mm%dd")
```

That expression can be replaced with the following expression:

```
(YearFromDate(ToTransform_Newdate.NEWDATE)*10000)+(MonthFromDate(ToTransform_Newdate.NEWDATE)*100)+(MonthDayFromDate(ToTransform_Newdate.NEWDATE))
```

If necessary, a cast from Integer to Decimal can be done later. But the dimension table should store the value as an integer anyway.

Some derivations use IF, THEN, and ELSE constructs:

```
If (WeekdayFromDate(ToTransform_Newdate.NEWDATE, "Monday") = 7) Then 1 Else 0
```

Make sure that these functions and expressions within the branches can get optimized. IF, THEN, and ELSE get optimized into CASE expressions in SQL. If not, the contained expressions might prevent this.

For column `D_DAYNUMINWEEK`, the usage of the following derivation function will cause an error:

```
WeekdayFromDate(ToTransform_Newdate.NEWDATE, "Monday")
```

The error is caused when the query using it is directed against DB2 and the accelerator. Although this function is listed to work with DB2, the statement using it receives the following error message:

```
Cannot map BASIC dialect function "WEEKDAYFROMDATE" to dialect DB2
WARNING: Dialect DB2 has no equivalent for
func(BASIC,WEEKDAYFROMDATE,[var(ToTransform_Newdate.NEWDATE),
```


The optimized job sequence that is based on database transformation now runs in approximately 32 seconds in our lab environment. No data is moved between the database system and DataStage any longer, and final results are stored within the accelerator.

Further hints to optimize jobs

If optimization of jobs is not possible, try to split the job into multiple subjobs, and use accelerator-only tables as staging tables between them. Optimize the subjobs as far as possible.

In case it is not possible to balance optimize a job, consider accelerator use and try to push as much job logic as possible to the source connector. This only makes sense when the accelerator does *not* have to return millions of rows, but only a couple of thousands. The more complex the SQL, the better. Do the remaining transformation logic in DataStage. If the number of resulting rows is small (below 500), consider updating a final accelerator-only table with **INSERT** statements.

Otherwise, store the results in a file and load them through a bulk insert into DB2. Add an additional load step from DB2 to the accelerator by using the `ACCEL_LOAD_TABLES` procedure as described in “`ACCEL_LOAD_TABLES`” on page 113. If the DB2 Analytics Accelerator Loader tool exits, consider using this to load the data directly into an accelerator shadow table. See 7.3.7, “Loading large amounts of data through DataStage to the Accelerator” on page 147 for further details.

7.3.6 Pitfalls

This section describes pitfalls that occurred while working on the scenarios in this book, and provides possible solutions.

Parallel jobs fail with “fail with ‘fork() failed’” error

The execution of a parallel job might fail with the error message “fail with fork() failed”. According to the following Techdoc, increasing the process limit might resolve the issue:

<http://www.ibm.com/support/docview.wss?uid=swg21393359>

In our lab environment, the process limit was already high enough, but we run multiple nodes on one virtualized computer. To resolve the problem, the number of parallel nodes must be decreased from 4 to 2.

Parallel jobs fail with “the record is too big to fit in a block” error

Depending on the InfoSphere Information Server configurations, running jobs with multiple stages where rows having columns with a large length (for example, `VARCHAR(320000)`) float between the stages, might fail with the following error:

```
DB2_Connector_15,0: Fatal Error: Virtual data set.; output of "DB2_Connector_15":  
the record is too big to fit in a block; the length requested is: 640268, the max  
block length is: 131072.
```

The following IBM Techdoc lists potential ways to solve this problem:

<http://www.ibm.com/support/docview.wss?uid=swg21416107>

For some sample jobs in this book, the APT_DEFAULT_TRANSPORT_BLOCK_SIZE environment variable has been set to 1048576 in the properties of the job (see Figure 7-61).

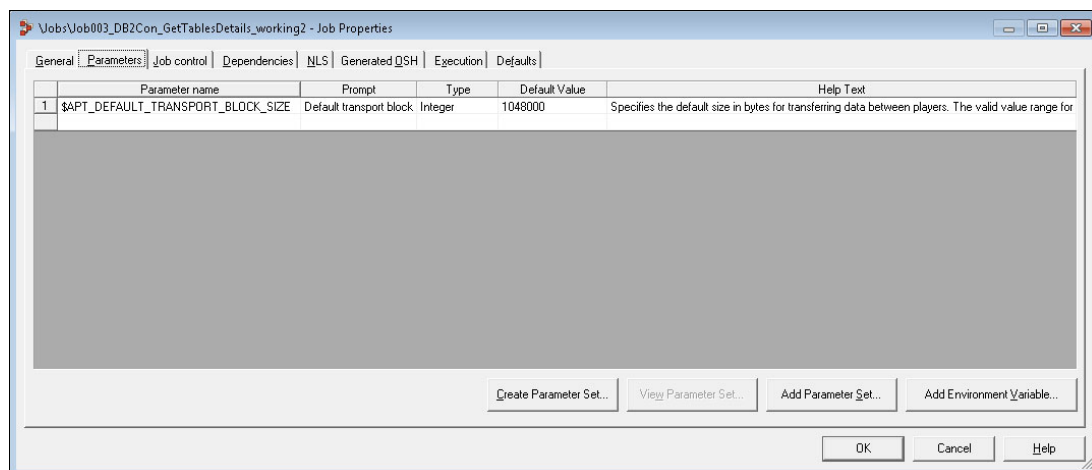


Figure 7-61 Set a higher transport block size to avoid job execution errors

Accelerating queries with the ODBC driver do not work

The ODBC driver enables you to set the special register for query acceleration, either in a DataStage job or through the DataStage engine shell. But running these consecutive queries fails, as shown in Example 7-21.

Example 7-21 Failing accelerated query run through ODBC

```
[dsadm@is-server DSEngine]$ bin/dssh
DataStage Command Language 11.3 Licensed Materials - Property of IBM
(c) Copyright IBM Corp. 1997, 2014 All Rights Reserved.
DSEngine logged on: Wednesday, April 22, 2015 15:22

>CONNECT BB_CC_TMCC_123_25
connect bb_cc_tmcc_123_25
BB_CC_TMCC_123_25> SET CURRENT QUERY ACCELERATION = ALL;
BB_CC_TMCC_123_25>
BB_CC_TMCC_123_25> SELECT count(*) FROM STKNOL.REGION;
[ODBC:1248] [IBM(DataDirect OEM)][ODBC DB2 Wire Protocol driver]Unhandled code
point: 1C03
BB_CC_TMCC_123_25>
```

The named IBM DRDA® codepoint 1C03 is a synonym for the ManagerLevelOverwrites that are negotiated between DRDA clients. The accelerator and DB2 use this protocol, and it seems like DB2 hands this code point to the ODBC driver as well. For accelerator usage, the possible values for this code point changed, and the ODBC driver might not have implemented this change yet. But this can change in the future.

This behavior is known for using other ODBC drivers in combination with the accelerator, and is described in the following Techdoc:

<http://www.ibm.com/support/docview.wss?uid=swg21622424>

Accelerated queries with DB2 Connect return wrong data

When the accelerator is used to answer queries issued by DataStage jobs, or through the DB2 CLI program on the DataStage processing engine, it can happen that character data is not returned properly.

To validate that this happens, let the query also return the hex values for the character columns. Also run the query against DB2 without acceleration and DB2 with accelerator (control this by setting the special register `CURRENT QUERY ACCELERATION` to `NONE` and `ALL`).

The cause can be an older version of the DB2 client. With Command Line Processor for DB2 Client 10.5.2, this behavior occurred and was fixed by upgrading to Fixpack 4 or higher.

Investigating this problem by using DataStage job output only is difficult. Try to store the output of the queries in a file and compare them.

Queries from DataStage are not running on the accelerator

When you experiment with accelerating queries through DB2 connectors, you might not find the query in the query monitoring section of the accelerator. This can happen if the query was either run on DB2 or the execution failed.

Consider a case in which the execution failed. Try to force a query to the accelerator by setting the special register for query acceleration to the following value:

```
SET CURRENT QUERY ACCELERATION = ALL
```

When running the job afterward, look at the job log and try to capture messages like the following messages:

```
DB2_Connector_TMCC_123_25,0: SQLExecute reported: SQLSTATE = 560D5: Native Error  
Code = -4,742: Msg = [IBM][CLI Driver][DB2] SQL0969N There is no message text  
corresponding to SQL error "-4742" in the message file on this workstation. The  
error was returned from module "DSNXODML" with original tokens "12".  
SQLSTATE=560D5 (CC_DB2DBStatement::executeSelect, file CC_DB2DBStatement.cpp, line  
2,074)
```

DB2 tells us that forcing the query to the accelerator was not possible (with `SQLCODE -4742`) and also gives us the reason (12). A lookup in the IBM Knowledge Center reveals that the error cause (12) originates from having the table disabled for acceleration:

http://www.ibm.com/support/knowledgecenter/SSEPEK_10.0.0/com.ibm.db2z10.doc.codes/src/tpc/n4742.dita

Another reason why the query does not appear in the monitoring section of IBM Data Studio is because DB2 runs the query. Verify the setting of the special register, and remember that the values `ENABLE` and `ENABLE WITH FAILBACK` let the DB2 optimizer make the decision whether it makes sense to run the query on the accelerator. DB2 might be the quickest to answer the query, so it ends up in DB2. Reconsider either the special register setting or the complexity of the query that you try to accelerate.

7.3.7 Loading large amounts of data through DataStage to the Accelerator

Loading data directly to the accelerator-only tables without using **INSERT** operations is supported with the DB2 Analytics Accelerator loader enterprise edition. As an alternative, either because this software product is not present or because of other problems, another alternative involves loading data into DB2 and afterward load the data to the accelerator, as described in “ACCEL_LOAD_TABLES” on page 113.

To do this, DataStage provides the bulk load capability. The combination of UNIX System Services pipes that serve as input for the load utility, and a File Transfer Protocol (FTP) transfer writing to the UNIX System Services pipe, enables loading large amounts of data to the DB2 side of an accelerator shadow table. Although this is an indirect way of storing large amounts of data on the accelerator, it is much faster than inserting many thousands or millions of rows through an **INSERT** operation.

The configuration of performing a bulk load is done on the target database connector of a job. Figure 7-62 on page 148 shows the properties for a load done from one node (total number of player processes = 1). The pipes are created in /tmp/ on the mainframe, because this is a DB2 on z/OS load. All default subsystem names (DSNs) are prefixed with STKNOL.ACCLD, where STKNOL is the user ID used to perform the load.

The retry option is set to No and the information, such as z/OS system, user name, and password, are taken from parameters questions during execution of the job. Because the table already exists, the table action is Append. All data existing in the table is dropped.

Figure 7-62 shows the load properties from one node.

DB2_Connector_1 - DB2 Connector

Stage Input

Input name (upstream stage)

DSLink4 (Peek_3)

General Properties Columns Advanced Partitioning

Usage View Data

Write mode *	Bulk load
Generate SQL	No
Table name *	STKNOL.REGIONCOPY
Enable quoted identifiers	No
Prefix for expression columns *	EXPR
SQL	
Table action *	Append
Transaction	
Session	
Logging	
Before/After SQL	No
Reoptimization	None
Lock wait mode	Use the lock timeout database configuration parameter
Pad character	
Limit parallelism	Yes
Total number of player processes *	1
Bulk load to DB2 on z/OS	Yes
Load method	USS pipe(s)
Transfer	
Transfer type	ftp
Transfer to *	#TransferToHost#
User	#FTP User#
Password	#FTP Password#
Transfer command	
Retry on connection failure	No
Number of retries *	3
Interval between retries *	10
DSN prefix	STKNOL.ACCLD
Batch pipe system ID *	
USS pipe directory *	/tmp
File(s) only	No
Device type	SYSDA
Partition number	
Row count estimate	1000
Statistics	None
Utility ID	DBZZLOAD
Load with logging	No
Set copy-pending	No
Encoding	UNICODE
Image-copy function	No
Data file attributes	
Load data (Info)	

OK Cancel Help

Figure 7-62 Bulk load configuration for mass data insertion

When using Balanced Optimization on a job with a bulk load insert, this becomes the write mode `Delete` then `Insert` when run on the accelerator. The table action remains the same as in the source job.

The following numbers were seen in our lab environment and might differ on other systems.

The sequence shown in “Transformation job recommendation” on page 132, whose execution takes around 22 minutes when running with **INSERT** statements only, can be shortened to around 3:30 minutes when performed with a bulk load.

An optimized job sequence with accelerator usage runs in approximately 32 seconds in our lab environment. The reduced data extraction time, and the in-database transformation queries that produced the accelerator-only tables, cause this speedup.



Accelerator-only tables supporting data scientists' ad hoc analysis

This chapter explains how accelerator-only tables can be used for ad hoc analysis performed by data scientists. Data scientists might want to interactively explore, analyze, and visualize data. Several packages are available to support these tasks, such as R and Matlab.

In the last several years, data analysis with Python, specifically with Python notebooks, has become more and more popular.

We show an example with healthcare insurance claims data from an operational system and how data scientists could create a copy of this data, filter, and transform it. Finally, they could use Python pandas and Matplotlib for further processing, and to render results.

The concept is to bring ad hoc analysis close to the data source, and perform extensive data processing with the help of IBM DB2 Analytics Accelerator ("Accelerator").

This chapter covers the following topics:

- ▶ Data science and ad hoc analysis
- ▶ Interactive analysis with notebooks: Python and Jupyter
- ▶ Example with insurance claim data in DB2 for z/OS
- ▶ More data scientist aspects

8.1 Data science and ad hoc analysis

Data scientists are getting insight from large volumes of data. The analysis that they apply could range from ad hoc data lookup to complex predictive and advanced analytics.

Compared to traditional business intelligence reporting, data science mostly deals with specific questions, and not repetitive requirements. In our context, we consider large volumes of data from operational systems, stored in IBM DB2 for z/OS on IBM z Systems. In finance, insurance, or retail, this could be core data from banking transactions, insurance claims, or purchase orders.

Compared to traditional (and regular) reporting, a data scientist works on a specific question, and explores, transforms, and processes data to get to an answer. This kind of request is more of an ad hoc and interactive nature and might only be run once.

An information technology (IT) organization might face multiple challenges, such as the following requests from their data scientists:

- How to provide the data for a data science environment?

Source systems can easily have multiple terabytes (TB) of data, which does not fit on a small-scale personal computer (PC) for analysis. Transferring all data takes time and results in security exposure.

- How to provide an environment with sufficient performance to run such an analysis on a large amount of data?

Most non-trivial analysis requires scanning and processing all source data, which could take too long on traditional disk-based systems. If all reasonable analysis on source data takes a prohibitively long time, a data scientist might be forced to reduce scope or complexity, and as a result of that compromise analysis quality.

- How to define a suitable security model and govern data access according to company policies?

It is always a challenge if all relevant data is transferred from governed production systems to other platforms for further processing. IT organizations might want to keep control and manage access based on business needs.

- How to keep data sources for data science up-to-date and enable the most current analysis?

Some kind of analysis might require the most recent data, and cannot work on an older snapshot. If data scientists need to work on a manual credit card fraud detection process, they also need the latest transactions.

- How to integrate with additional non-transactional data from other sources?

Though the transactional data hosted on DB2 for z/OS would be considered the most important source, additional data sources (for example, publicly available data from the Internet) must be considered for data science analysis as well.

Now, consider how accelerator-only tables could support these requirements, and how they could provide an attractive foundation for data science on DB2 for z/OS transactional data.

Figure 8-1 shows a transaction processing system with customer transactions stored on DB2 for z/OS. DB2 Analytics Accelerator is used today to speed up traditional reporting queries against this data. The data on the Accelerator might be kept up-to-date using the incremental update feature.

The IT organization could now create work areas for their data scientists *John* and *Jane*. These are just traditional DB2 for z/OS databases where John and Jane have permissions to create their database objects. Further rules restrict table space and disk volume usage. These rules also set special registers for Accelerator use in a way that they can essentially run workloads only on the Accelerator, and do not affect production work.

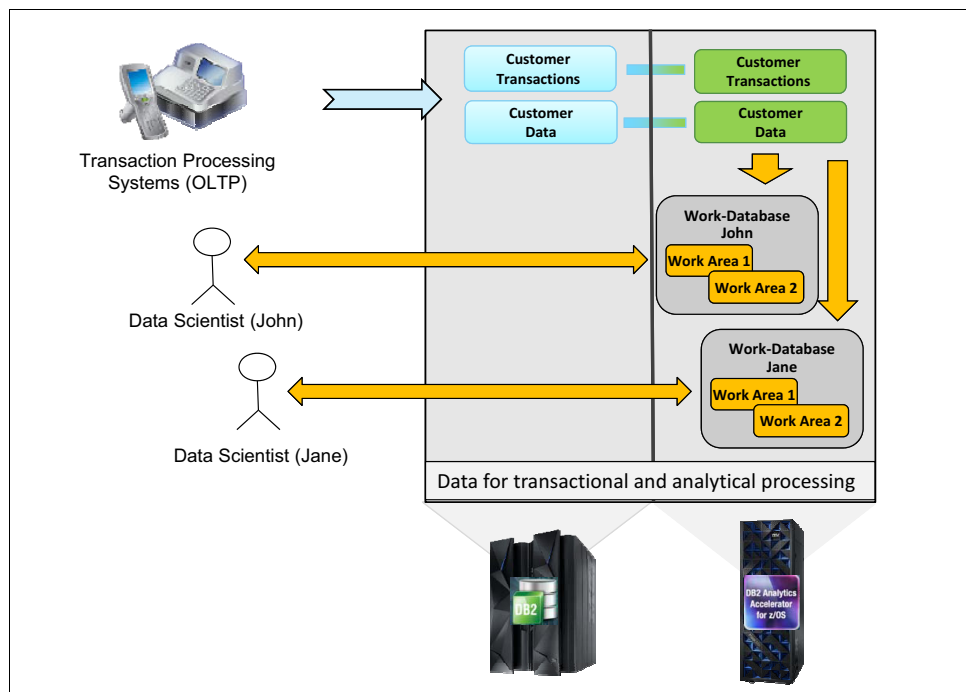


Figure 8-1 Data scientist work areas

Next, these data scientists can access current customer transactions, take their snapshot, filter, transform, and group data, and finally process the outcome interactively and efficiently in their data scientist environment.

With this setup, IT organizations obtain the following benefits:

- Compliance to existing security standards and centralized management

Access to production data is controlled and governed by DB2 for z/OS and IBM RACF®. The source of the data is and remains the mainframe. There are no uncontrolled copies of the entire transactional data in other places.

- Flexibility and self-service operation for data scientists

John and Jane can access data and create objects and transformations on their own and restricted to “their” database. They do not affect production processing.

- Performance with access to a large amount of data

The proposed concept assumes that all expensive pre-calculations are run on DB2 Analytics Accelerator. The Accelerator is designed and built to run massively parallel processing against large amount of data efficiently.

- Proximity to operational system (data currency)

With the assumption that data is being replicated to the Accelerator already, data scientists can access data in almost real time, and do not have to care about the data maintenance process and how to gather the last update for their private work area environment.

- Integration of non-transactional data from other sources

Chapter 9, “Integrating more data sources and archiving for analytics” on page 165 explains how to load data from external sources into the Accelerator. When the data is available at the Accelerator, it can be easily integrated and combined with other DB2 for z/OS data.

8.2 Interactive analysis with notebooks: Python and Jupyter

Notebooks, such as IPython¹, started to become popular vehicles to run interactive data analytics. A web browser interface enables easy data manipulation, filtering, mining, and processing. Notebooks can be saved for later recall, and to be exchanged with others.

The Python script language evolved over the past few years, and now includes many no-charge packages to perform data analysis and data transformation in memory. Particularly interesting are NumPy and pandas, which add a rich set of functions for multi-dimensional data handling and processing.

The book *Python for Data Analysis*, ISBN 978-1-449-31979-3² provides an overview about Python and Python notebook usage in the context of Data Analysis and Data Science.

After IPython 3.x, the project moved to a new project under the name *Jupyter* (<https://jupyter.org/>), now also covering other popular languages, such as *R*, *Julia*, and *Ruby*.

In our environment, we use Anaconda, a no-charge Python distribution that can be downloaded from the following website:

<http://continuum.io/downloads>

In addition to Python itself, Anaconda includes many additional packages for science, math, engineering, and data analysis. It also includes a ready-to-use installation of Jupyter.

8.3 Example with insurance claim data in DB2 for z/OS

The sample data used in this chapter is based on (anonymous) health insurance claims data. We illustrate the following actions:

- How a snapshot of this data source is accessed and initially analyzed
- How transformation and filtering are applied
- How the result is processed and graphically rendered using a Python notebook

¹ See Fernando Pérez, Brian E. Granger, *IPython: A System for Interactive Scientific Computing*, *Computing in Science and Engineering*, vol. 9, no. 3, pp. 21-29, May/June 2007, doi:10.1109/MCSE.2007.53. URL: <http://ipython.org>

² Wes McKinney, *Python for Data Analysis*, O'Reilly, 2013

Figure 8-2 depicts the high-level structure of data and processing.

On the left, we have claims data with 1,805,933 records, each representing a health insurance claims request. This is considered to be an accelerator-shadow table from an operational system, and could be potentially large.

The objective of our example is to understand which diagnosis groups are the most frequent ones, and also provide a graphical rendering of the “top 20” groups.

The claims data includes a diagnosis ID, which we can use as a starting point. Additional master data helps us to translate these diagnosis ID values into diagnosis groups keys. An additional lookup table maps diagnosis group keys to descriptive text.

Figure 8-2 shows how we derive snapshot and transformed data to store them in accelerator-only tables (AOTs), and how processing continues in Python pandas to finally render diagnosis group information of the 20 most frequently specified diagnosis.

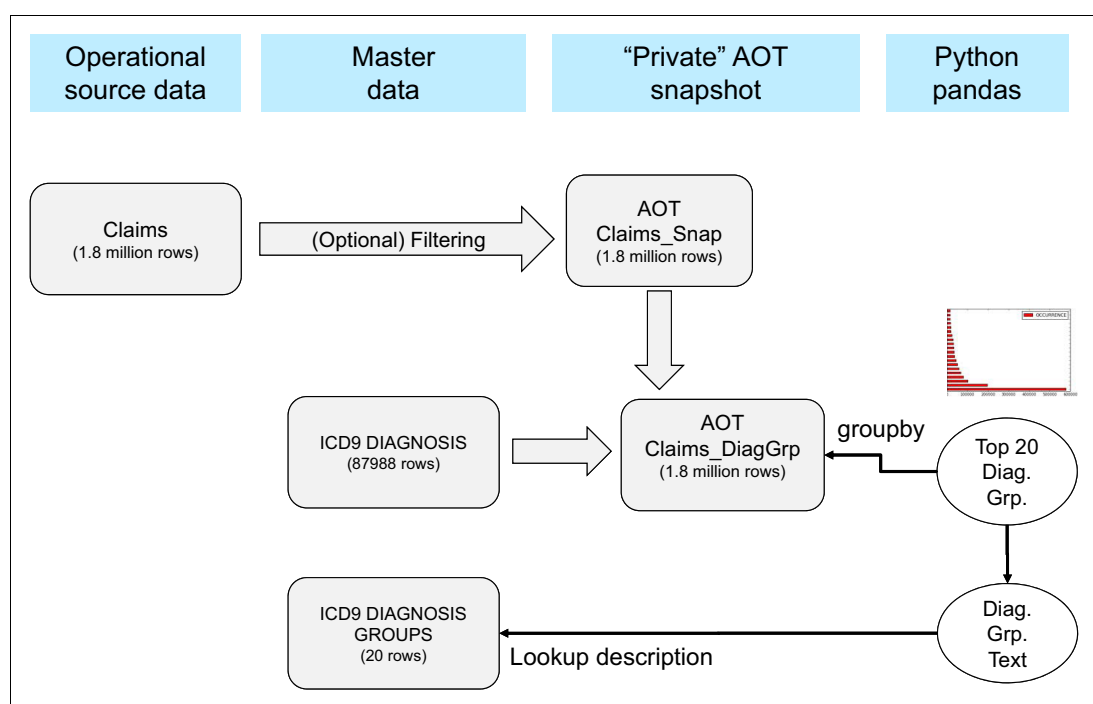


Figure 8-2 Structure and layout of sample data and processing

It is important to note that we run all initial transformation and grouping on the 1.8 million claims records on the Accelerator in accelerator-only tables. Python code continues with a smaller and aggregated subset of the data.

For the sample that we use in this book, it might be feasible to run the entire sequence in memory on a common notebook computer. However, real production data of larger organizations could easily reach sizes that do not fit easily into the random access memory (RAM) of commodity systems.

The proposed approach has a price. It requires handling data with two data manipulation techniques (Structured Query Language (SQL) and Python pandas). However, it also provides the benefits of scalability and performance. The code examples in 8.3.3, “Data analysis examples with accelerator-only tables” on page 159 show how these two concepts can be integrated with little effort.

8.3.1 Sample data layout in DB2 for z/OS

For this chapter, we use three tables for the insurance claims data.

The first table (CLAIMS) has claims with patient information, age, gender, total charge amount, and diagnosis. Example 8-1 shows the table layout for this data.

Example 8-1 CLAIMS table definition

```
CREATE TABLE CLAIMS
(PATIENT_ID          CHAR(25) FOR SBCS DATA NOT NULL,
 BIRTH_DATE         DECIMAL(3, 0) WITH DEFAULT NULL,
 GENDER_ID          CHAR(1) FOR SBCS DATA WITH DEFAULT NULL,
 AGREEMENT_ID       CHAR(10) FOR SBCS DATA WITH DEFAULT NULL,
 CLAIM_ID           CHAR(14) FOR SBCS DATA NOT NULL,
 TYPE_ID            CHAR(5) FOR SBCS DATA WITH DEFAULT NULL,
 CLEAN_RECEIPT_DATE DATE WITH DEFAULT NULL,
 PROCESS_DATE       DATE WITH DEFAULT NULL,
 TOT_CHARGE_AMOUNT  DECIMAL(15, 0) WITH DEFAULT NULL,
 DIAGNOSIS_ID       VARCHAR(10) FOR SBCS DATA
    WITH DEFAULT NULL,
 HPD_ID             CHAR(10) FOR SBCS DATA WITH DEFAULT NULL,
 PROCEDURE_MODIFIER_ID CHAR(5) FOR SBCS DATA
    WITH DEFAULT NULL)
```

To analyze diagnosis information, two additional site tables with master data, ICD_DIAGNOSIS and ICD9_DIAGNOSIS_GRP, are included.

The ICD_DIAGNOSIS table maps a diagnosis number to a group, and also includes a textual description. For the data in our database, this table has 87,988 rows, and its layout is shown in Example 8-2.

Example 8-2 ICD9_DIAGNOSIS table definition

```
CREATE TABLE ICD9_DIAGNOSIS
(DIAGNOSIS          CHAR(8) FOR SBCS DATA WITH DEFAULT NULL,
 ICD9_DX_GROUP_NBR  CHAR(3) FOR SBCS DATA WITH DEFAULT NULL,
 FIELD3             CHAR(3) FOR SBCS DATA WITH DEFAULT NULL,
 FIELD4             CHAR(2) FOR SBCS DATA WITH DEFAULT NULL,
 DX_DESC            CHAR(30) FOR SBCS DATA WITH DEFAULT NULL,
 DX_DESC_LONG       CHAR(250) FOR SBCS DATA WITH DEFAULT NULL)
```

Data in column DIAGNOSIS almost maps to DIAGNOSIS_ID in the CLAIMS tables. There is some further processing required to manage a join, though.

Table ICD9_DIAGNOSIS_GRP has descriptive text for diagnosis groups. It includes 200 rows and the table layout shown in Example 8-3.

Example 8-3 ICD9_DIAGNOSIS_GRP

```
CREATE TABLE ICD9_DIAGNOSIS_GRP
(ICD9_DX_CTG_CD     CHAR(4) FOR SBCS DATA WITH DEFAULT NULL,
 ICD9_DX_GROUP_NBR  CHAR(4) FOR SBCS DATA WITH DEFAULT NULL,
 ICD9_DXGRP_DSCRPTN CHAR(44) FOR SBCS DATA WITH DEFAULT NULL,
 SHORT_DSCRPTN      CHAR(32) FOR SBCS DATA WITH DEFAULT NULL)
```

8.3.2 Accessing DB2 for z/OS data in Python

We need to access DB2 for z/OS data in a Python script, but also submit SQL statements, for example, to create accelerator-only tables or to set the special register for query acceleration.

There are a couple of options with additional packages, mostly provided by IBM to connect to a DB2 database management systems.

ibm_db

This application programming interface (API) is defined by IBM, and provides the best support for advanced features. In addition to issuing SQL queries, calling stored procedures, and using IBM pureXML®, you can access metadata information.

The Python database interface (DBI) driver for DB2 (IBM DB2 for Linux, UNIX, and Windows; IBM z/OS; and IBM i5/OS™) and IBM Informix® Dynamic Server (IDS) is available for download from the following website:

https://pypi.python.org/pypi/ibm_db

The included readme file describes support for the following databases:

- ▶ IBM DB2 Database on Linux, UNIX, and Windows 8.2 and onwards
- ▶ Informix Dynamic Server Cheetah version 11.10 onwards
- ▶ Remote connections to IBM i5/OS (iSeries)
- ▶ Remote connections to z/OS (DB2 for z/OS)
- ▶ DB2 for Mac OS

ibm_db_dbi

Implements Python Database API Specification v2.0, which provides basic functions for interacting with databases but does not offer the advanced features provided by `ibm_db`.

This API implements Python Database API Specification v2.0. Because the `ibm_db_dbi` API conforms to the specification, it does not offer some of the advanced features that the `ibm_db` API supports. If you have an application with a driver that supports Python Database API Specification v2.0, you can easily switch to `ibm_db`. The `ibm_db` and `ibm_db_dbi` APIs are packaged together.

For more details about Python Database API Specification, see the following website:

<https://www.python.org/dev/peps/pep-0249/>

ibm_db_sa

This adapter supports SQLAlchemy, which offers a flexible way to access IBM Database servers. SQLAlchemy is a popular open source Python SQL toolkit and object-to-relational mapper (ORM).

The package is available from the following website:

https://pypi.python.org/pypi/ibm_db_sa

Supported databases include DB2 for Linux, UNIX, and Windows versions 9.7 onwards.

SQLAlchemy would be an interesting option for our task, because it provides rich metadata access and manipulation options. For example, we could use SQLAlchemy to derive table metadata from operational tables, adjust it for our analysis needs, and create accelerator-only tables with the wanted layout. However, `ibm_db_sa` currently does not support DB2 for z/OS.

An attempt to retrieve database table metadata from a DB2 for z/OS data source results in an error due to different catalog table names, as shown in Example 8-4.

Example 8-4 Catalog names error

```
DBAPIError: (ibm_db_dbi.Error) ibm_db_dbi::Error: SystemError('error return
without exception set',) [SQL: 'SELECT "SYSCAT"."COLUMNS"."COLNAME",
"SYSCAT"."COLUMNS"."TYPENAME", "SYSCAT"."COLUMNS"."DEFAULT",
"SYSCAT"."COLUMNS"."NULLS", "SYSCAT"."COLUMNS"."LENGTH",
"SYSCAT"."COLUMNS"."SCALE", "SYSCAT"."COLUMNS"."IDENTITY",
"SYSCAT"."COLUMNS"."GENERATED" \nFROM "SYSCAT"."COLUMNS" \nWHERE
"SYSCAT"."COLUMNS"."TABSCHEMA" = ? AND "SYSCAT"."COLUMNS"."TABNAME" = ? ORDER BY
"SYSCAT"."COLUMNS"."COLNO"'] [parameters: ('FNEUMAN', 'ICD9_DIAGNOSIS')]
```

Therefore, `ibm_db_sa` cannot be used in our context.

Jupyter configuration for DB2 for z/OS access

We define a set of variables in a file (named `10-dbsetup.py`), in the startup directory of the `.ipython` profile directory. These variables are used to access the DB2 for z/OS database in the notebook. In our test environment, we set the following definitions (the password for the user ID is read and referenced from a local read-protected file):

```
dbName = 'CC825'
dbHostname = 'tmcc-123-25.boeblingen.de.ibm.com'
dbPort = '446'
dbUsername = 'fneuman'
dbPassword = password
```

Sample data exploration

To get things started, look at the provided sample data and determine how it can be accessed in our Python environment using `pandas` and `Matplotlib`:

1. First, we access our DB2 for z/OS database using `ibm_db`, then also derive an `ibm_db_dbi` connection object. Example 8-5 creates both objects for use in the following examples.

Example 8-5 Connecting to DB2 for z/OS from Python

```
import ibm_db
import ibm_db_dbi
ibm_db_conn = ibm_db.connect(dbName, dbUsername, dbPassword)
conn = ibm_db_dbi.Connection(ibm_db_conn)
```

2. Then, we use `pandas` to get grouped data for `GENDER_ID`. The `read_sql()` call creates a `pandas DataFrame`, which could be plotted using `Matplotlib`. Example 8-6 shows how the gender object is created. The `GENDER_ID` attribute is defined as index so that those values appear on the first axis of the bar chart.

Example 8-6 Creating pandas from a database query and rendering the result as bar chart

```
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
from pandas import Series, DataFrame
gender = pd.read_sql("SELECT GENDER_ID, COUNT(*) number FROM CLAIMS GROUP BY
GENDER_ID", conn)
gender.index = gender.GENDER_ID
gender.plot(kind='bar')
```

Figure 8-3 shows the inline output of this sample. Obviously, more claim requests have been submitted from male patients than from female patients.

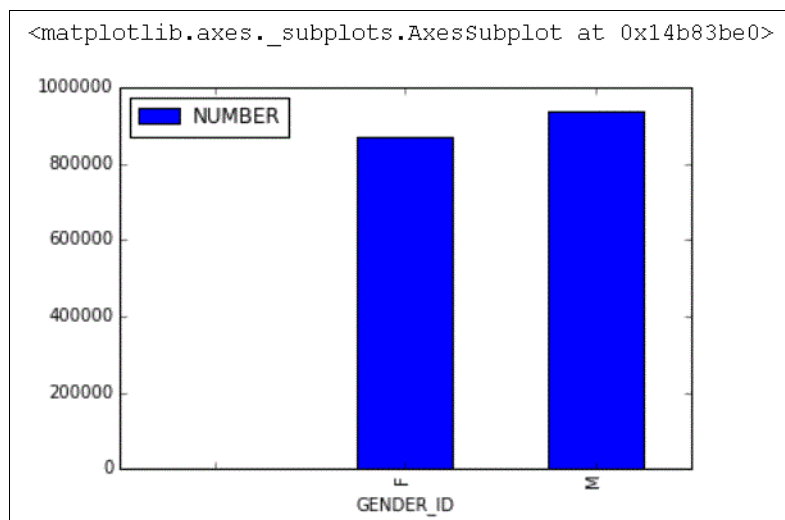


Figure 8-3 Using Matplotlib to render female and male claims distribution

8.3.3 Data analysis examples with accelerator-only tables

In a first step, we take a snapshot of the CLAIMS table content and copy the content with some filtering and transformation into an accelerator-only table CLAIMS_SNAP. The structure of CLAIMS_SNAP includes columns for age, gender, total charge amount, and diagnosis. Furthermore, for this exercise, we are only interested in data newer than 2012. This illustrates how to filter data to potentially reduce its size for further analysis.

Then, we also transform data in the DIAGNOSIS column, because we need data in a different format to combine (join) the data with our ICD9 diagnosis data.

Example 8-7 starts with a verification of whether the new table already exists. We are using the `table()` function on the `ibm_db_dbi` connection object to return either an empty list, or the name of the table if it exists. In that case, it is dropped before we proceed using an `exec_immediate()` call.

For creating the new accelerator-only table CLAIMS_SNAP, we select a subset of the columns and use a `CREATE` statement followed by “IN ACCELERATOR STRIPER”, which is the name of the defined Accelerator. This example also assumes that a database DSCDB exists, where the current user ID is permitted to create objects.

An insert-select statement reads data from table CLAIMS (our accelerator-shadow table with production data) and performs transformations for the `diagnosis_ID` field to include a Period (.) in the fourth place. This is required to match definitions in our master table, and shows how data transformations can be implemented during such a step.

Example 8-7 shows you how to create an accelerator-only table.

Example 8-7 Creating an accelerator-only table with filtering and transformation

```
if len(conn.tables('FNEUMAN', 'CLAIMS_SNAP')) > 0:
    # CLAIMS_SNAP exists already, dropping
    ibm_db.exec_immediate(ibm_db_conn, "DROP TABLE CLAIMS_SNAP");

ibm_db.exec_immediate(ibm_db_conn, """
CREATE TABLE CLAIMS_SNAP(AGE DECIMAL(3), GENDER char(1), TOT_CHARGE_AMOUNT
DECIMAL(15), DIAGNOSIS VARCHAR(8)) IN ACCELERATOR STRIPER IN DATABASE DSCDB
""")

ibm_db.exec_immediate(ibm_db_conn, """
INSERT INTO CLAIMS_SNAP
SELECT BIRTH_DATE, GENDER_ID, TOT_CHARGE_AMOUNT,
(CASE WHEN (LENGTH(STRIIP(DIAGNOSIS_ID)) > 3)
    THEN LEFT(DIAGNOSIS_ID,3) || '.' || SUBSTR(DIAGNOSIS_ID,4)
    ELSE diagnosis_id END) DIAGNOSIS
FROM CLAIMS
WHERE PROCESS_DATE > '2012-01-01'
""")
```

Running the insert-select statement for our 1.8 million rows takes about 1.5 seconds on our test system. Before reading and processing the data in our Python notebook environment, we further prepare the data and add diagnosis group information to each claim. This requires a join between the larger CLAIMS table and our two additional site tables with diagnosis group information.

For this, we create an extra accelerator-only table following the same pattern as before, but now performing a join on the Accelerator with another table to include diagnosis group information as well. Note that the join operation only works with an accelerator-only table if the other referenced table (ICD9_DIAGNOSIS in this case) is on the Accelerator as well.

Example 8-8 starts with a similar sequence as in Example 8-7, but then runs a SQL statement with a join of accelerator-only table CLAIMS_SNAP and accelerator shadow table ICD9_DIAGNOSIS.

Example 8-8 Creating a new accelerator-only table CLAIMS_DIAGGRP with joined data

```
if len(conn.tables('FNEUMAN', 'CLAIMS_DIAGGRP')) > 0:
    # CLAIMS_DIAGGRP exists already, dropping
    ibm_db.exec_immediate(ibm_db_conn, "DROP TABLE CLAIMS_DIAGGRP");

ibm_db.exec_immediate(ibm_db_conn, """
CREATE TABLE CLAIMS_DIAGGRP(AGE DECIMAL(3), GENDER char(1),
TOT_CHARGE_AMOUNT DECIMAL(15), DIAGNOSIS VARCHAR(8), DIAG_GRP CHAR(4))
IN ACCELERATOR STRIPER IN DATABASE DSCDB
""")

ibm_db.exec_immediate(ibm_db_conn, """
INSERT INTO CLAIMS_DIAGGRP
SELECT S.AGE, S.GENDER, S.TOT_CHARGE_AMOUNT, S.DIAGNOSIS, D.ICD9_DX_GROUP_NBR
FROM CLAIMS_SNAP S, ICD9_DIAGNOSIS D
WHERE S.DIAGNOSIS = D.DIAGNOSIS
""")
```

Figure 8-4 shows the first entries of the resulting new table with the defined attribute.

	AGE	GENDER	TOT_CHARGE_AMOUNT	DIAGNOSIS	DIAG_GRP
1	58	F	161	327.23	77
2	49	M	59	327.23	77
3	59	M	237	327.23	77
4	74	M	324	496	176
5	45	M	161	327.23	77
6	39	M	60	327.23	77
7	41	M	161	327.23	77
8	73	F	230	715.95	180
9	64	F	324	496	176
10	67	M	62	491.20	176
11	73	F	143	719.45	139
12	52	M	60	327.23	77

Figure 8-4 Content of CLAIMS_DIAGGRP accelerator-only table after transformation

Now that we have data preparation completed with the help of our accelerator-only tables, we use Python pandas to sort by occurrence, take the top 20 only, and merge the resulting data with diagnosis text. Finally, the data is plotted using a barchart.

Example 8-9 starts with building diagGrpText, a pandas object that is used to map diagnosis group identifiers to descriptive text. Then claimDiagGrp is built as a result of a group-by SQL statement. Note that this statement only returns 197 rows, and is significantly smaller than the original 1.8 million rows.

With both pandas objects available, we now sort by occurrence, and merge the top 20 entries with matching diagnosis group text information. The merge function automatically selects matching columns.

Example 8-9 Data processing and rendering with Python pandas

```

ibm_db.exec_immediate(ibm_db_conn, "SET CURRENT QUERY ACCELERATION=ELIGIBLE")
# Read diagnosis group text into pandas
diagGrpTxt = pd.read_sql("SELECT ICD9_DX_GROUP_NBR DIAG_GRP, ICD9_DXGRP_DSCRPTN
FROM ICD9_DIAGNOSIS_GRP", conn)
# Read number of diagnosis groups from frames from acceleraor-only table
claimDiagGrp = pd.read_sql("SELECT DIAG_GRP, COUNT(*) OCCURRENCE FROM
CLAIMS_DIAGGRP GROUP BY DIAG_GRP", conn)
# Get the top 20 diagnosis groups and merge it with description text
top20DiagGrp =
claimDiagGrp.sort_index(by='OCCURRENCE',ascending=False)[:20].merge(diagGrpTxt)
top20DiagGrp.index = top20DiagGrp.ICD9_DXGRP_DSCRPTN
top20DiagGrp.plot(kind='barh',color='r')

```

Figure 8-5 shows the result, and we see that “Neurologic Disorders” is by far the most frequent diagnosis captured in our data.

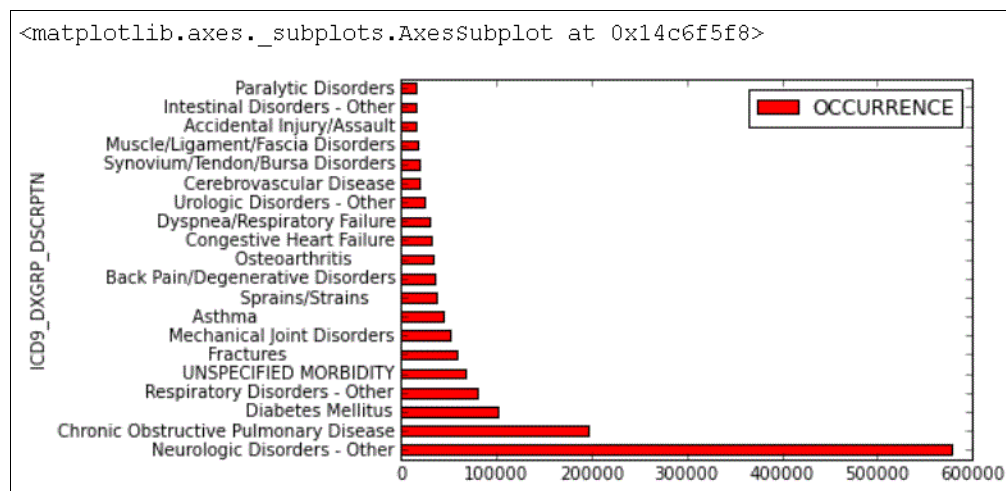


Figure 8-5 Top 20 diagnosis groups by occurrence

For the next example, we are now interested in exploring the total charge amount per claim, and how this amount is distributed. Even though it takes some time to load individual claims data into pandas, we do it here with `diag_grp` and `total amount` data. In our test environment, it took 75.7 seconds to read the data and build the pandas object for the 1.8 million rows.

Example 8-10 shows that we use a SQL select statement, casting the `DIAG_GRP` value as an integer for later rendering. We then create a scatter plot to get a first visual idea how data is distributed.

Example 8-10 Creating a scatter plot for total charge amount

```
ibm_db.exec_immediate(ibm_db_conn, "SET CURRENT QUERY ACCELERATION=ELIGIBLE")
# For all claims, get total charge amount and diagnosis group
chargeAmt = pd.read_sql("SELECT CAST(DIAG_GRP AS INTEGER) DIAG_GRP,
TOT_CHARGE_AMOUNT FROM CLAIMS_DIAGGRP", conn)
plt.scatter(chargeAmt['DIAG_GRP'],chargeAmt['TOT_CHARGE_AMOUNT'] )
```

Figure 8-6 shows that, as expected, there are many claims below USD 10,000, but a few get almost up to USD 70,000.

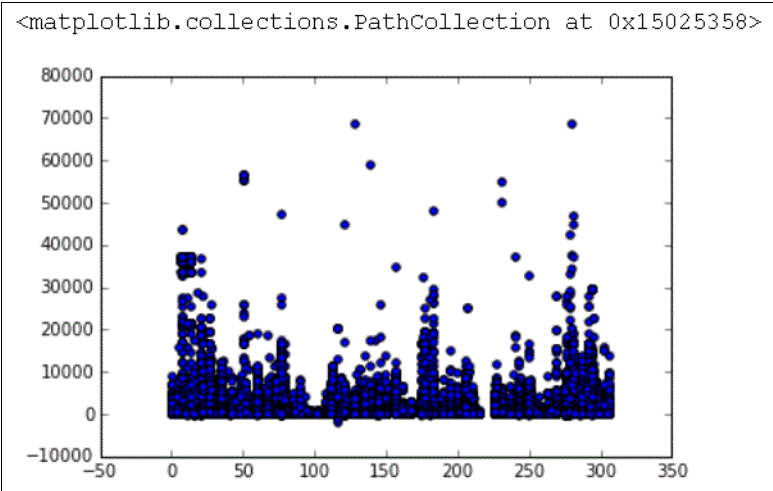


Figure 8-6 Scatter plot example for total charge amount and diagnosis group

To run a quick numerical analysis on the data in chargeAmt, the describe() function is called to calculate statistical values for both columns. The output in Figure 8-7 tells us that 50% of the 1.8 million claims are below USD 165 and 75% of them are below USD 343. The maximum, however, is USD 68,694.

In [99]: chargeAmt.describe()

Out[99]:

	DIAG_GRP	TOT_CHARGE_AMOUNT
count	1805545.000000	1805545.000000
mean	133.749792	358.213779
std	80.485436	815.840181
min	1.000000	-2000.000000
25%	77.000000	72.000000
50%	136.000000	165.000000
75%	178.000000	343.000000
max	306.000000	68694.000000

Figure 8-7 Output of the describe() function called for the total charge amount data

8.4 More data scientist aspects

The previous sections demonstrated how data scientists could use data available on DB2 Analytics Accelerator to run private analysis on large production data. The example we selected is certainly pretty simplified, and data scientists must deal with more sources and more analysis techniques than just filtering, joining, and aggregating.

For less demanding scenarios, data scientists might be able to express processing in SQL so that they could also use other tools than Python notebooks.

Although we said that data scientists' work is more ad hoc in nature, there are also cases where requirements come close to production workload. For example, suppose that a set of potentially fraudulent credit card transactions must be verified manually. Depending on the required procedure, access to payment history, and master data about the customer, needs to be accessed, including other similar transactions.

In Chapter 9, "Integrating more data sources and archiving for analytics" on page 165, we explain how to bring external data into accelerator-only tables. This capability is also key for many data scientist scenarios, because data needs to be joined or preprocessed with data that does not reside on DB2 for z/OS natively.



Integrating more data sources and archiving for analytics

This chapter describes use cases where data from other sources is loaded into the IBM DB2 Analytics Accelerator (Accelerator), and because of that provides integration with existing data in DB2 for z/OS tables. We introduce DB2 Analytics Accelerator Loader as the primary tool for loading data from other sources and other formats directly into the Accelerator.

Together with the accelerator-only tables (AOTs), this option opens a set of new use cases where the Accelerator and DB2 for z/OS can serve as a data integration hub, particularly if it comes to new concepts to enhance analytics capabilities for existing DB2 for z/OS operational data. Furthermore, we also describe cases where accelerator-only tables can be used for archiving, complementing the built-in high performance storage saver feature.

This chapter covers the following topics:

- ▶ DB2 Analytics Accelerator Loader for z/OS
- ▶ General function overview
- ▶ Support for accelerator-only tables
- ▶ Integrating other DBMS data with DB2 for z/OS data
- ▶ Architecture pattern: Incremental archiving with accelerator-only tables

9.1 DB2 Analytics Accelerator Loader for z/OS

The DB2 Analytics Accelerator Loader for z/OS V1.1 (5639-OLA), hereafter referred to as the Loader, provides additional possibilities and options for loading data into DB2 for z/OS and DB2 Analytics Accelerator.

9.2 General function overview

The Loader addresses several challenges for loading data into the Accelerator. Although a real-time synchronization is not important in most cases, a common goal is to minimize the application effect when loading data. Another aspect of flexibility that the Loader provides is to load a historical point-in-time into the Accelerator, while maintaining a consistent set of data being loaded.

The DB2 Analytics Accelerator Loader supports loading of non-DB2 data into DB2 Analytics Accelerator, and optionally into DB2 for z/OS when the data is compatible with the IBM DB2 LOAD utility, which can be achieved by extraction and transformation processes.

9.2.1 Group Consistent Load

Loading data from one or more operational DB2 tables into DB2 Analytics Accelerator, while maintaining data consistency to the same point in time, is called *Group Consistent Load*. This method of loading data into DB2 Analytics Accelerator maintains full availability of DB2 tables that are loaded into DB2 Analytics Accelerator.

DB2 tables that are part of a Group Consistent Load operation can be related by referential integrity (RI), or simply unrelated tables. The data can be loaded to DB2 Analytics Accelerator to current time, or to a user-specified point in time for historical data analysis.

Rather than unloading the data from the DB2 tables, DB2 Analytics Accelerator Loader leverages DB2 recovery resources (DB2 Image Copies and DB2 Logs) and processes (Recover and log-apply) to load data into DB2 Analytics Accelerator.

Figure 9-1 illustrates an overview of *Group Consistent Load*. The starting point for Group Consistent Load is the most recent *DB2 full image* copy that is available. DB2 Analytics Accelerator Loader then moves forward through the DB2 log and applies correlated DB2 log records until the specified point in time, either user-defined or current time.

To maintain transaction consistency, any uncommitted transactions at the specified point in time are not loaded to the DB2 Analytics Accelerator.

When loading to current point in time, DB2 Analytics Accelerator Loader can optionally create a new IBM FlashCopy® *Consistent Image Copy*. This is beneficial if the last full image copy was taken several days or more ago, and also eliminates log reading purposes.

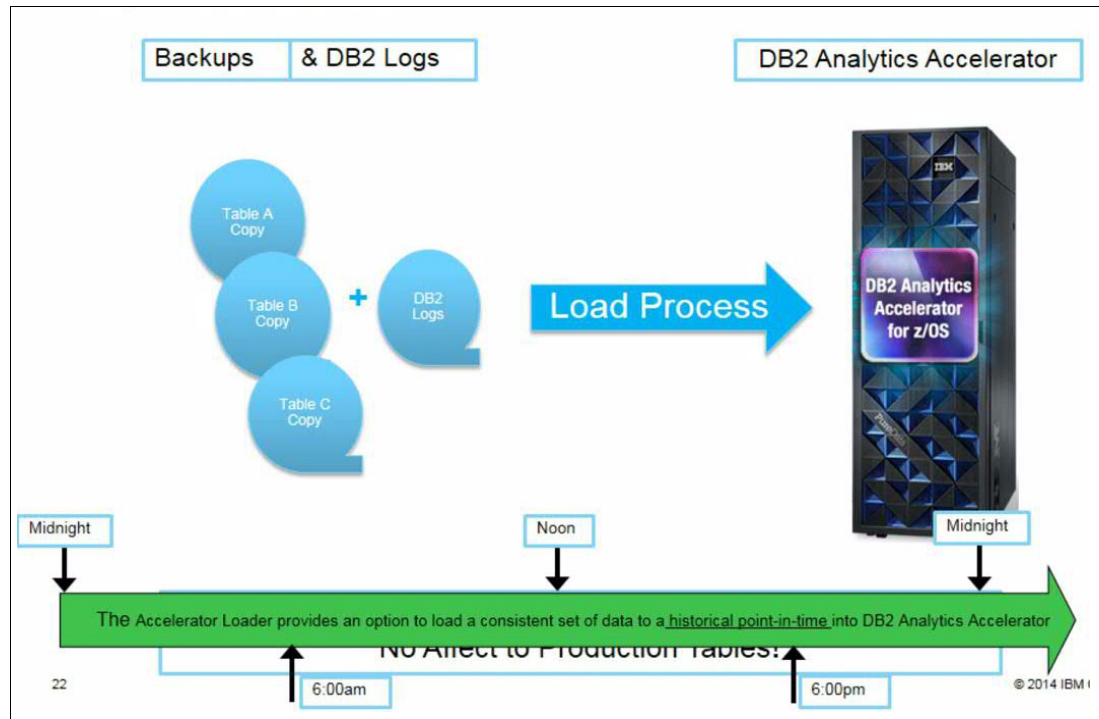


Figure 9-1 Group Consistent load

9.2.2 Dual load

In most common cases, when loading data into DB2 Analytics Accelerator, the data itself is already available in DB2. However, in an increasing number of cases, the intended data is external to DB2, because it originates from another DB2 subsystem on a different logical partition (LPAR), a non-relational database on IBM z/OS, or another non-z/OS data store.

In this case, the normal process is to load data into DB2 first, followed by loading it into DB2 Analytics Accelerator.

As illustrated in Figure 9-2, with Dual load, sometimes also referred to as *External load*, DB2 Analytics Accelerator provides the ability to load external data into DB2 for z/OS and DB2 Analytics Accelerator in parallel (IDAA_DUAL). It is also possible to only load DB2 Analytics Accelerator (IDAA_ONLY). The latter case leaves the DB2 table empty.

Dual load improves and simplifies loading external data by combining two steps into one. When DB2 Analytics Accelerator Loader uses IBM System z Integrated Information Processor (zIIP), significant processor (CPU) use savings can be achieved.

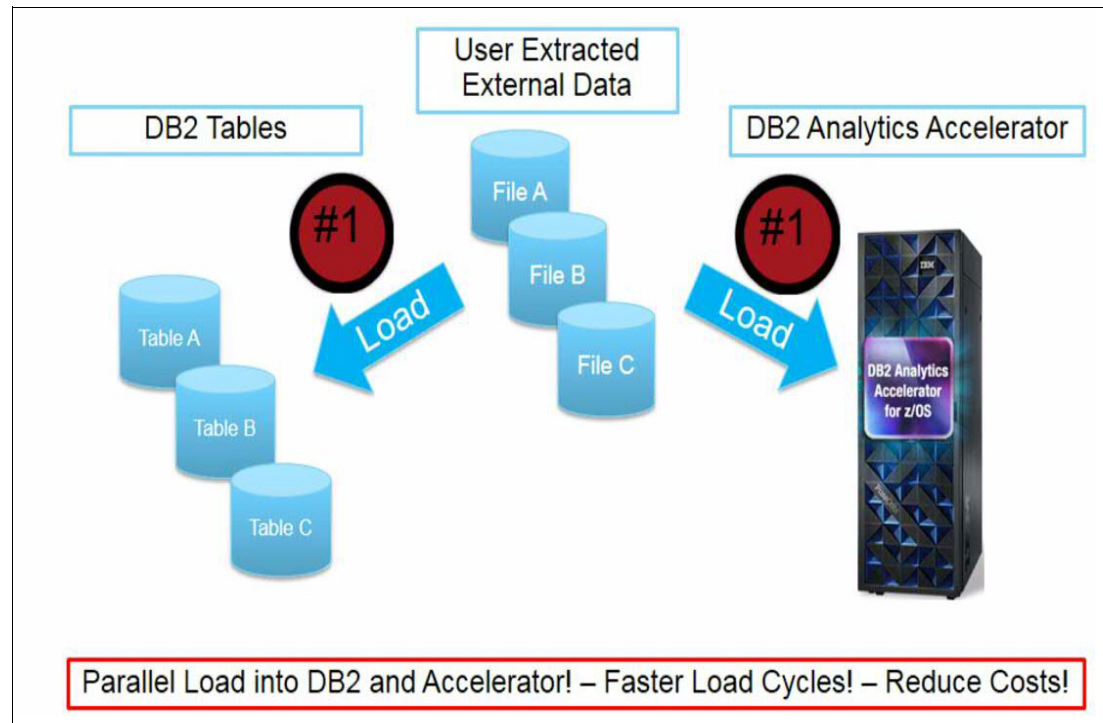


Figure 9-2 Overview of Dual load mode

9.2.3 Image Copy load

The term *Image Copy load* describes the ability to use an external or alternative image copy to load into DB2 Analytics Accelerator. For example, the source can be an online transaction processing (OLTP) image copy, or a copy of an alternative table of the same DB2 subsystem.

The point in time when the image copy was taken represents the point in time of the data. Image Copy load only restores the data of the image copy itself, and does not perform any log-apply like *Group Consistent load* would do.

Image Copy load was introduced by authorized program analysis report (APAR) PI21811, and provides the ability to consolidate from different DB2 systems into a DB2 Analytics Accelerator. Another use case is to perform a historical load to an alternative table for week-end or month-end analysis in the same system.

Figure 9-3 illustrates a brief overview of Image Copy load.

Time-to-market is key in today's business world. Application development and quality assurance (QA) cycles are decreased more and more. Ensuring proper QA is a challenge. With Image Copy load, source Operational Data Store (ODS) or data warehouse (DWH) image copies can be used for redirected loads or shadow table loads. Therefore, problem diagnosis or data performance analysis can be conducted without affecting existing queries.

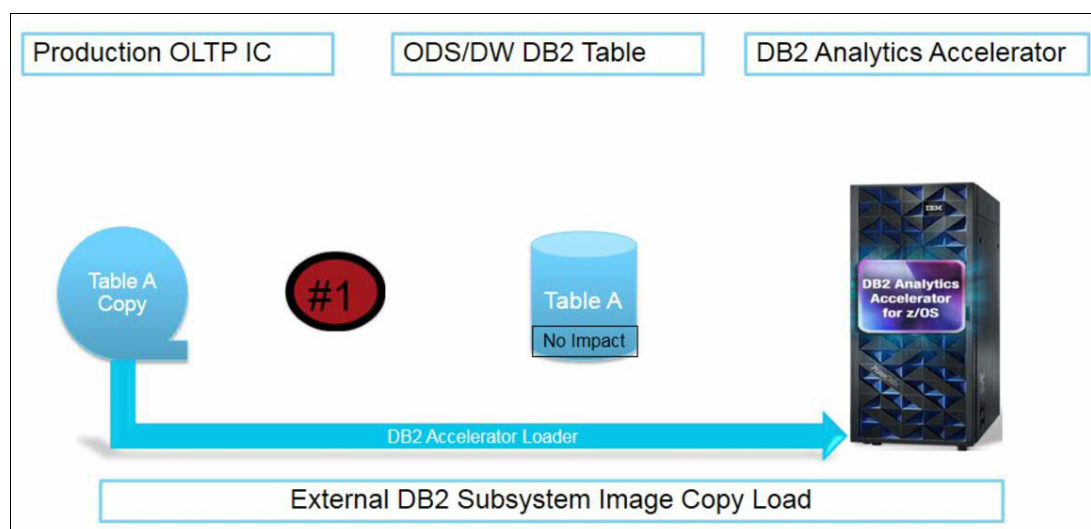


Figure 9-3 Loading the Accelerator-shadow table from an existing DB2 for z/OS Image Copy

9.3 Support for accelerator-only tables

DB2 Analytics Accelerator Loader also provides extended loading capabilities for accelerator-only tables with DB2 Analytics Accelerator Version 4 program temporary fix 6 (PTF 6) and the following PTFs:

- ▶ PI49338: HLO-1092: External Load Support for Accelerator Only Table
- ▶ PI49351: HLO-1091: Image Copy Load for Accelerator Only Table

DB2 provides standard SQL Data Manipulation Language (DML) **INSERT**, **UPDATE**, and **DELETE** (I/U/D) operations against accelerator-only tables. Although this works fine for small data amounts or regular updates, it can become challenging when loading huge data amounts or, simply spoken, *bulk load*.

Accelerator-only tables are naturally a great fit for non-DB2 data. When using DB2 Analytics Accelerator to take the advantage of the power of the system to gain insight into your application data.

A typical use case is credit card fraud detection of a global bank. Its common customer credit card information is stored in a Virtual Storage Access Method (VSAM) repository. The credit card fraud detection application is running off the mainframe in a distributed environment.

To fill this data warehouse, the actual customer data needs to be extracted from the VSAM data source, and transformed for the target data warehouse. This is followed by a load into the distributed system, which requires a transport mechanism, such as File Transfer Protocol (FTP) or similar.

With the use of accelerator-only tables, extraction and potential transformation of the VSAM data is still required. The loading process can be simplified. As for standard DB2 tables, the following general process is for loading external data into an accelerator-only table:

1. Create a DB2 table to match external data schema.
2. Use the DB2 **LOAD** utility to load external data into DB2.
3. Add the DB2 table to the Accelerator (accelerator-shadow table).
4. Populate the accelerator-shadow table by starting the ACCEL_LOAD_TABLES stored procedure.
5. Create an accelerator-only table in the Accelerator.
6. Execute **SQL DML INSERT ... SELECT FROM** to populate the accelerator-only table.

9.3.1 Loading accelerator-only tables using Accelerator Loader

DB2 Analytics Accelerator Loader provides the same loading capabilities for accelerator-only tables as for a standard DB2 table. However, accelerator-only tables only exist in DB2 Analytics Accelerator. Like **DEFINE NO** tables, accelerator-only tables have a place-holder in DB2, but not a physical VSAM data set associated to it. As a result of this, loading external data into DB2 for z/OS and DB2 Analytics Accelerator in parallel is not supported by the DB2 Analytics Accelerator Loader.

Building a historical data repository for corporate-revision purposes is easy to achieve using accelerator-only tables and DB2 Analytics Accelerator Loader. To implement this pattern, an accelerator-only table is created which corresponds to the data layout of the external data, for example, IMS.

The process of loading accelerator-only tables using Accelerator Loader is fairly simple:

1. Create an accelerator-only table to match the external data schema.
2. Start Accelerator Loader to load external data directly into the Accelerator-only table.

DB2 Analytics Accelerator Loader is then used to **LOAD APPEND** the external data to the existing accelerator-only table.

9.4 Integrating other DBMS data with DB2 for z/OS data

The accelerator-only tables also makes the Accelerator a superior analytics engine for non-DB2 data sources. This section shows a sample data workflow illustrating how to use the Accelerator to create analytics reports related to IBM Information Management System (IMS) using the IBM DB2 Analytics Accelerator Loader for z/OS. The process of using the Accelerator without accelerator-only tables for data from IMS is as follows:

1. Identify the target IMS data sources.
2. Extract data from IMS using tools such as **UNLOAD** or custom-written applications. The main challenge here is the amount of data that is required to be transferred.
3. Create DB2 for z/OS tables with structure that matches the extracted IMS data.
4. Load the extracted data into DB2 tables by using the DB2 **LOAD** utility.
5. Run the SYSPROC.ACCEL_LOAD_TABLES stored procedure to load the data from DB2 into the accelerator-shadow table.

After this step, the IMS data is accessible through the Accelerator.

There are two areas where the Accelerator Loader can improve the load process of IMS data into the Accelerator. Taken the analytics use case, the intent of transferring IMS data into the Accelerator is for improving report performance. It is usually not required to store the IMS data in DB2.

Using the Accelerator Loader with the accelerator-only (IDAA_ONLY) option, the IMS data can be loaded into the Accelerator directly, without needing to stage the data in DB2 first. This is where the accelerator-only table's objects are seamlessly part of this solution. So the Accelerator Loader improves elapsed time and processor consumption in this case.

Figure 9-4 shows the sample workflow using the IBM DB2 Analytics Accelerator Loader to load extracted IMS data into Accelerator.

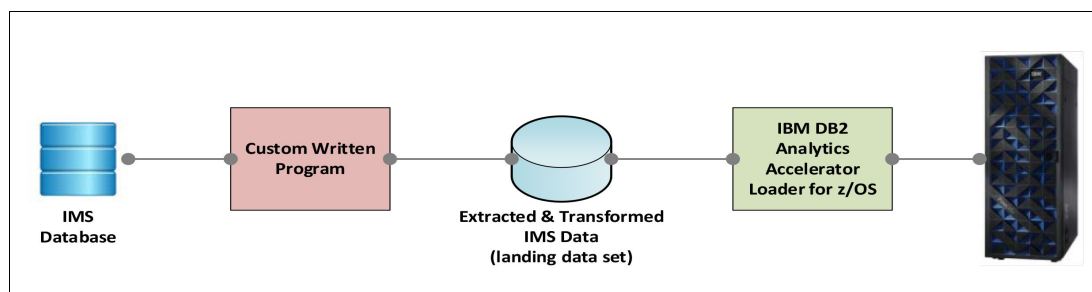


Figure 9-4 Extract IMS data using customer written applications

IBM InfoSphere DataStage supports data extraction and loading for IMS data to DB2 for z/OS. Many customers use IBM InfoSphere DataStage for extended data transformation (for example, cleansing and aggregation) for their IMS data.

Integrating the IBM DB2 Analytics Accelerator Loader into a DataStage data workflow is easy, and provides significant loading improvements. DataStage supports the standard way of loading data into the Accelerator. The IMS data is loaded into DB2 for z/OS through IBM DB2 Connect first. That data is then transferred into the Accelerator using the DB2 **SYSPROC.ACCEL_LOAD_TABLES** stored procedure.

A performance improvement alternative is to offload the IMS data into a data set through DataStage, then use the Accelerator Loader or DB2 LOAD utility to load the data from the data set into the Accelerator.

Because the Accelerator Loader also supports UNIX System Services named pipes as data input for improved performance. Rather than writing to a data set, DataStage can write the data to an UNIX System Services named pipe. The DB2 Analytics Accelerator Loader then reads the data from the UNIX System Services named pipe, and loads the records directly into the Accelerator, either an accelerator-shadow or accelerator-only table. This avoids input/output (I/O) activities of accessing records in a data set.

Figure 9-5 shows the combined workflow using DataStage and the Analytics Loaders.

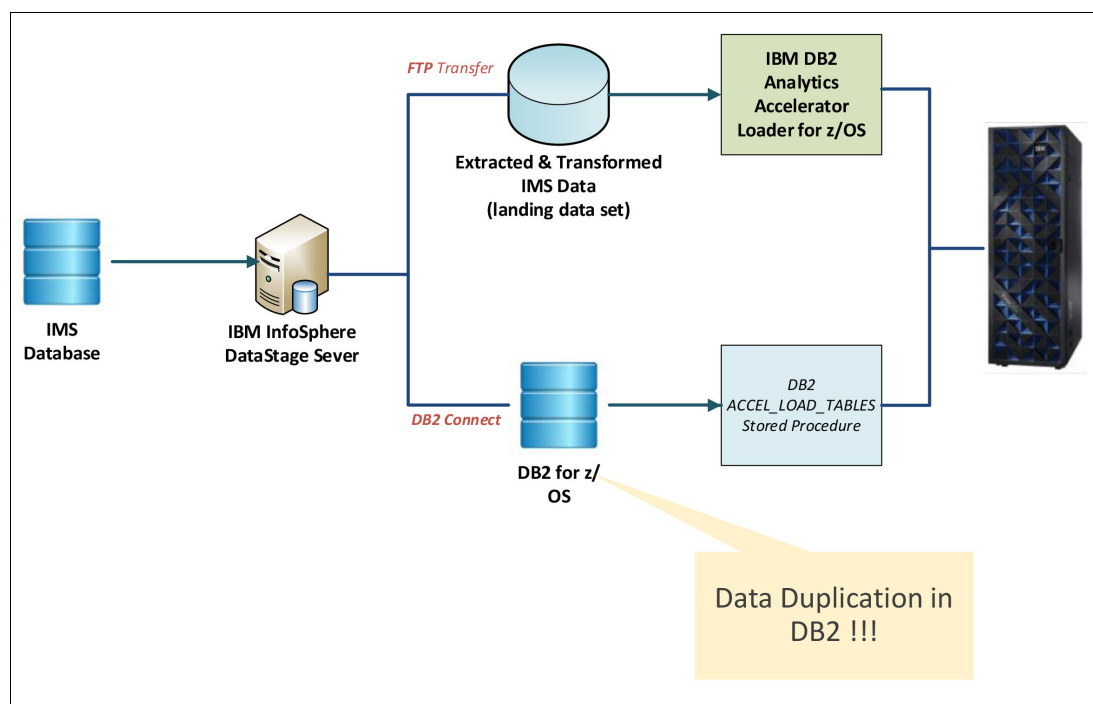


Figure 9-5 Data extraction workflow using IBM InfoSphere DataStage and the Accelerator Loader

9.5 Architecture pattern: Incremental archiving with accelerator-only tables

In addition to data integration and transformation requirements, accelerator-only tables can also be used to address more information technology (IT)-specific requirements.

This section shows how to implement incremental archiving, and how to run incremental data updates with the help of accelerator-only tables.

The High Performance Storage Saver (HPSS) feature in DB2 Analytics Accelerator enables you to archive historical data in DB2 table partitions to reclaim space for data that is mostly read-only.

After selecting a partition to be archived, an image copy is taken, native storage for data in the corresponding DB2 partition storage is freed up, and the partition is set to a Partition Read-Only (PRO) state, protecting it from undesired further updates. This concept is well-suited if a DB2 table is partitioned, particularly if data is partitioned by date and time, and when new data arrives in the current partition only. In that case, the DB2 *rotate partition* function can be used to define and continuously manage a fixed time frame of history data.

HPSS is the premier feature to address archiving requirements with the Accelerator, because it includes many automated checks, recovery, and even high availability (HA) capabilities.

There are cases however, where a more flexible approach is wanted. For example, if the source table is not partitioned, or not partitioned by range, but still needs to have a limited amount of active data with older data stored somewhere else.

Figure 9-6 shows an architecture pattern with AOTs to satisfy this requirement. A DB2 table keeps operational data for (in this example) 60 days plus the current day. We assume that transactions continuously I/U/D rows for the current day (day 61) only. We call this the *operational* table, because it is used and changed by operational applications.

This operational table has been added to the Accelerator and its content is maintained there as well, using either incremental update using Change Data Capture (CDC) or regular table/partition unload.

To implement the pattern, a history table with the same table layout as the operational table is created as an accelerator-only table (AOT). This history table also contains data older than 60 days, and is used for reporting or analytics.

At the beginning of a new day (Day 61), there are no more changes for the last day (Day 60), and its content can be copied from the operational table to the history table. At the same time, the oldest data in the operational table is removed to keep it at the intended size. This procedure is similar to a *rotate partition* operation, and happens in three steps:

1. An insert-select statement copies data from the last day (Day 60) into the *History AOT*.
2. A backup of the oldest data (*Day 1*) data is taken from the operational table in DB2. This can be an unload job, selecting data by date.
3. Day 1 data is deleted from the operational table in DB2.

Figure 9-6 illustrates the procedure for incrementally building data history in an accelerator-only table.

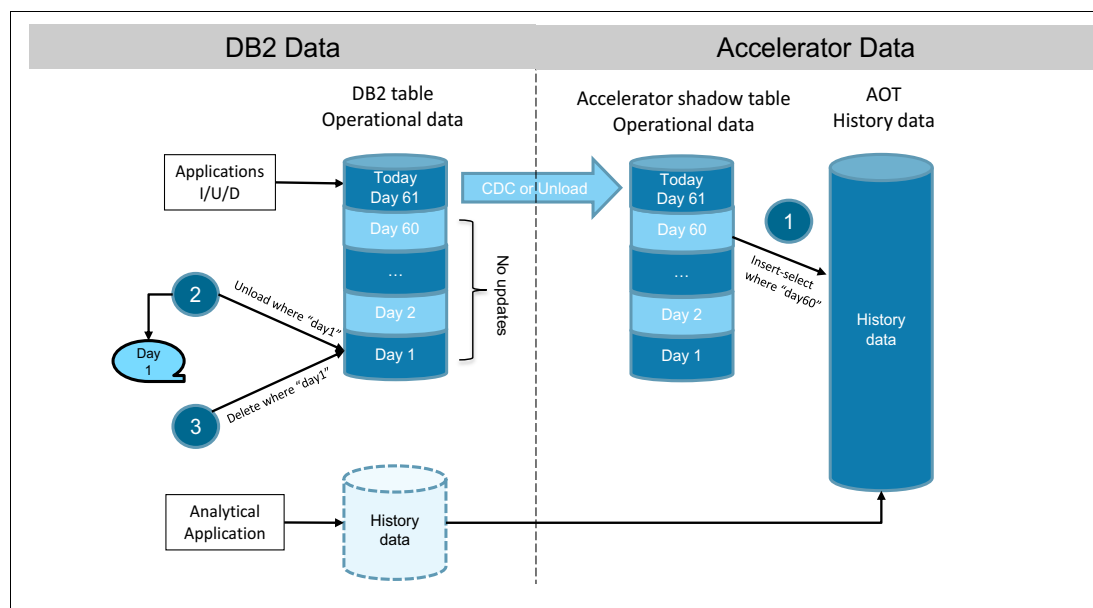


Figure 9-6 Incrementally building data history in an accelerator-only table

There are variations of this pattern:

- The distinction between active and read-only data in the operational data does not have to be defined by date, but also can be by any other criteria.
- If changes happen all over the operational table, the daily processing (step 1 in Figure 9-6) can also replace (first delete, then insert) all of the operational table data in the history table. The high performance of insert-select operations on the Accelerator enables you to handle larger operational tables that way as well.

Figure 9-7 illustrates how to handle the case when the history AOT is lost and needs to be rebuilt using the daily backup data.

An empty recovery table for a single day is created with the same layout as the operational and history table. This is a standard DB2 table, which is also defined with an Accelerator shadow. Then DB2 Analytics Accelerator Loader can load (load or replace) the backup data directly into the recovery shadow table on the Accelerator (step 1). Then, an insert-select statement moves this data to the history AOT (step 2) where it is cumulated.

These steps are repeated for all daily backup files that are recovered.

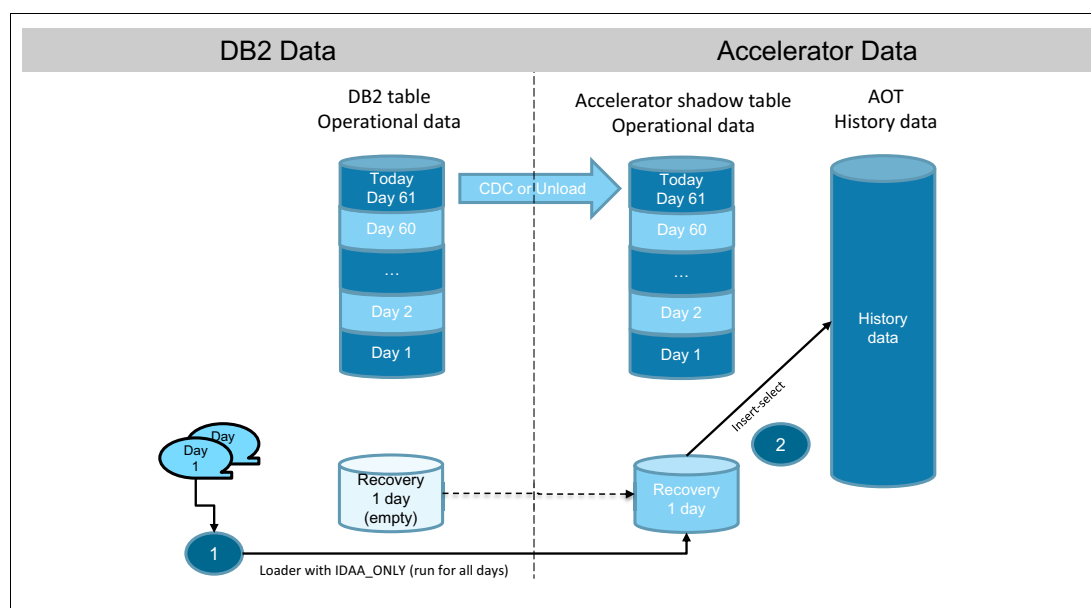


Figure 9-7 Recovery of data in a history AOT

Again, variations to this pattern can be implemented if backups are not taken daily, or if they are already cumulated on the mainframe.

The way we describe building the history table in this paragraph leads to an ever-growing history table. Usually, disk size on the Accelerator is not an issue, and the fast scan speed enables quick access to large data. Some use cases might require more disk space. Nevertheless, that data in the history table is managed as well, so that only data for a certain time frame is kept there. Because accelerator-only tables also support delete operations, an additional step can be introduced to regularly drop old data from the history table as well.



In-database analytics

IBM DB2 Analytics Accelerator V5.1 for z/OS now supports a subset of existing analytical capabilities that are well-known from IBM Pure Data for Analytics appliances. These capabilities are called *IBM Netezza Analytics*, which is an embedded, purpose-built, and advanced analytical platform.

This functionality is now partially available to users of IBM DB2 Analytics Accelerator V5.1, which uses the same platform. This chapter describes the newly introduced analytical capabilities for a DB2 Analytics Accelerator, and shows how these can be used with IBM SPSS Modeler.

This chapter covers the following topics:

- ▶ Reasons to use in-database analytics with IBM DB2 Analytics Accelerator
- ▶ Executing analytical functions using an Accelerator
- ▶ Enable IBM SPSS modeling for acceleration
- ▶ Technical implementation

10.1 Reasons to use in-database analytics with IBM DB2 Analytics Accelerator

Advanced analytics enables you to unveil hidden or not obvious information in the warehouse data that cannot be retrieved easily with queries or reporting. In that way, it can help to improve the business in all industries. For instance, mailing campaigns can be optimized by analyzing sales transaction data and sending offers to those customers with the highest potential interest. Another example is fraud detection, where patterns of fraudulent behavior can be detected more easily, making it possible to prevent fraud more effectively.

In general, finding such insights is an iterative process. Starting with a concrete business question, you must determine the data sources that might help to answer the question. The selected data usually must be preprocessed (combined, aggregated, and transformed), until it can be fed into an appropriate data mining algorithm, which computes a data mining model representing the found insights.

Usually, the results of the initial data mining runs are not sufficient for solving the business questions. Therefore, additional data sources might be selected, other data transformation steps might be applied, or data mining runs with different parameter settings or different algorithms might be run, until a data mining model with a satisfactory quality is computed.

When such a model is found, it can be deployed in an operational context in which a particular business operation can be performed. For instance, customers are selected for a mailing campaign or product recommendations are computed. This step is called scoring.

In the past, data was offloaded from an IBM z/OS data warehouse to perform these types of analytics on a different platform. In most cases, this was to avoid additional load on the IBM z Systems mainframe.

However, this is a time-consuming, manual process that must be repeated multiple times, so this offloading process could take days or even weeks until a model with a reasonable quality is built. This makes it difficult to react to changed circumstances in a timely fashion. In addition, it was difficult to ensure that only those people with the appropriate authorizations had access to this data.

This is different with IBM DB2 Analytics Accelerator (Accelerator) and its new advanced analytics functionality. All steps of a data mining process can be run on the same system. Off loading and uploading data is not necessary. The data remains in a secure environment. No extra security measures need to be implemented, because access control is handled by DB2 for z/OS.

This makes it easy and efficient to find new insights in the warehouse data. Data analysts can perform all steps of the data mining process. They can quickly react to the results of one iteration without needing the intervention of database or system administrators. In this way, better data mining models can be built in only a fraction of the time needed when performing advanced analytics on a different platform.

10.2 Executing analytical functions using an Accelerator

Consider the example of credit card fraud detection. The task of data modeling is likely to involve large amounts of data to finally come up with a comprehensive fraud detection data model, which is used to analyze and identify fraudulent transactions. Data mining algorithms analyze this data and compute data mining models that can be used in the *scoring* process to identify fraudulent transactions both in online and batch systems.

Applying the same model over a period of time to identify fraudulent transactions without adjusting it might not provide the expected results infinitely, especially because fraud patterns can quickly change. Patterns change quickly because advanced technology is widespread and almost universally available. It might not be feasible to re-create a complex model that is applied within the scoring process multiple times per day because of simple elapsed time considerations.

Calculating a data mining model can be performed in different ways. One way to build a data mining model is to leave the processing to a specialized data mining tool, such as SPSS Modeler, which does not necessarily rely on Structured Query Language (SQL). Another method is to use SQL to perform the modeling task to create a data mining model.

If your modeling application allows you to use SQL to calculate your data mining model, you can now use an Accelerator to perform this task. Given the speed with which an Accelerator can process SQL workload, it is likely that those models can be calculated more often, probably multiple times a day. They can then be used on the Accelerator to score transactions in batch mode. It is also possible to transfer them to the z/OS side, where they can be applied in real time to online transactions with SPSS Modeler Scoring Adapter for DB2 for z/OS.

Being able to update fraud-detection models on time, and to integrate scoring tightly into online transaction processing (OLTP) environments, leads to more effective detection of fraudulent transactions, which in turn results in decreased financial losses and staying ahead of suspicious activity.

Figure 10-1 depicts a graphical representation of how a modeling task can be performed with an Accelerator.

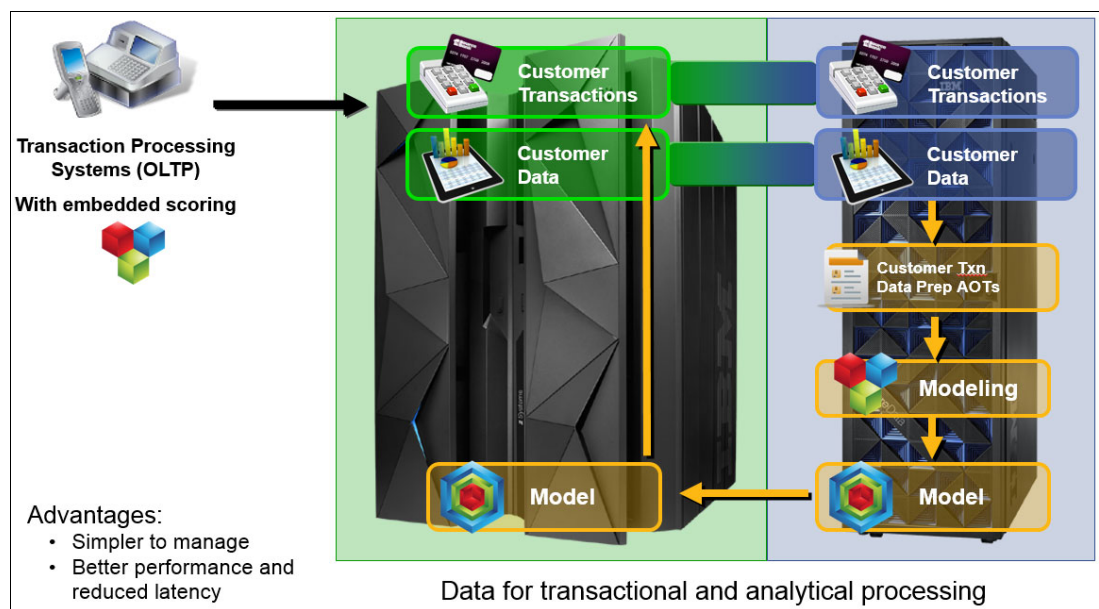


Figure 10-1 Using Accelerator-only tables and ELT logic in the Accelerator for SPSS modeling

To begin with, DB2 for z/OS contains customer transactions, such as credit card transactions. Either an extract or the full set of transactions is used to create the required model. Typically, because of the sheer amount of data, a representative sample of customer transactions is used to create the required data mining model.

After you have identified the data source for creating a data mining model, you copy corresponding tables to an Accelerator and enable them for acceleration. If required, data can be enriched or transformed as input for the modeling process.

The algorithm to build your data mining model then uses SQL to access data stored on an Accelerator, and the resulting model can then be copied back to DB2 for z/OS to finally use it for real-time scoring. In Figure 10-1 on page 177, we marked in orange those steps of the entire process that can actually run on an Accelerator.

With DB2 Analytics Accelerator V5.1, you can now create data mining models with the help of an Accelerator. The analytical functions running on an Accelerator are made available by DB2 for z/OS stored procedures that are routed to the Accelerator. The code containing the mining algorithms runs on the Accelerator only.

Because of this encapsulation, you can call these stored procedures either directly or through specific tooling in DB2 for z/OS. One prominent example of such tooling to support these additional analytical capabilities is IBM SPSS Modeler. The following section provides more information about SPSS Modeler supporting DB2 Analytics Accelerator.

10.3 Enable IBM SPSS modeling for acceleration

With SPSS Modeler 17, you can perform all modeling activities on data within the Accelerator by using SQL or inside SPSS Server. For this purpose, we enabled these procedures that are required to run SPSS Modeler with an Accelerator in DB2 for z/OS. For further reference, we list all new stored procedures used by SPSS in 10.4, “Technical implementation” on page 180.

Modeling without enhanced Accelerator support

Before Accelerator-only tables and additional analytical functionality were available on an Accelerator, support for data modeling was limited to the initial extraction of source data that was then passed as input to the modeling process. Figure 10-2 depicts this situation.

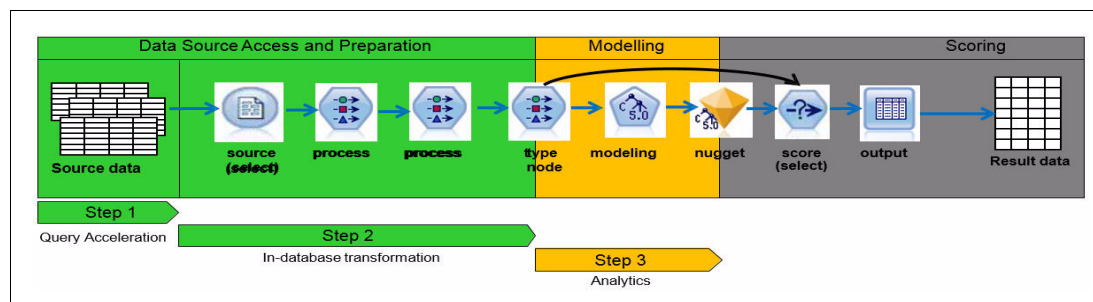


Figure 10-2 SPSS Modeler without running analytical functions on an Accelerator

Step 1 refers to the extraction of source data that can be performed by an Accelerator. Steps 2 and 3 are performed outside of an Accelerator, either in SPSS Server or DB2 for z/OS, resulting in elongated elapsed times.

The SPSS source node selects the input data from DB2 z/OS from an Accelerator in step 1. In steps 2 and 3, all data preparation is done in SPSS Modeler, followed by the modeling algorithm in SPSS Modeler. The process is completed by scoring the model and inserting scoring results into a DB2 z/OS result table.

Modeling with an Accelerator supporting the entire process

The full support of SPSS Modeler is provided with DB2 Analytics Accelerator Version 5.1, which contains support for both Accelerator-only table and analytical functionality. The capability of running analytical functionality now enables the modeling process to be entirely performed on an Accelerator, as shown in Figure 10-3.

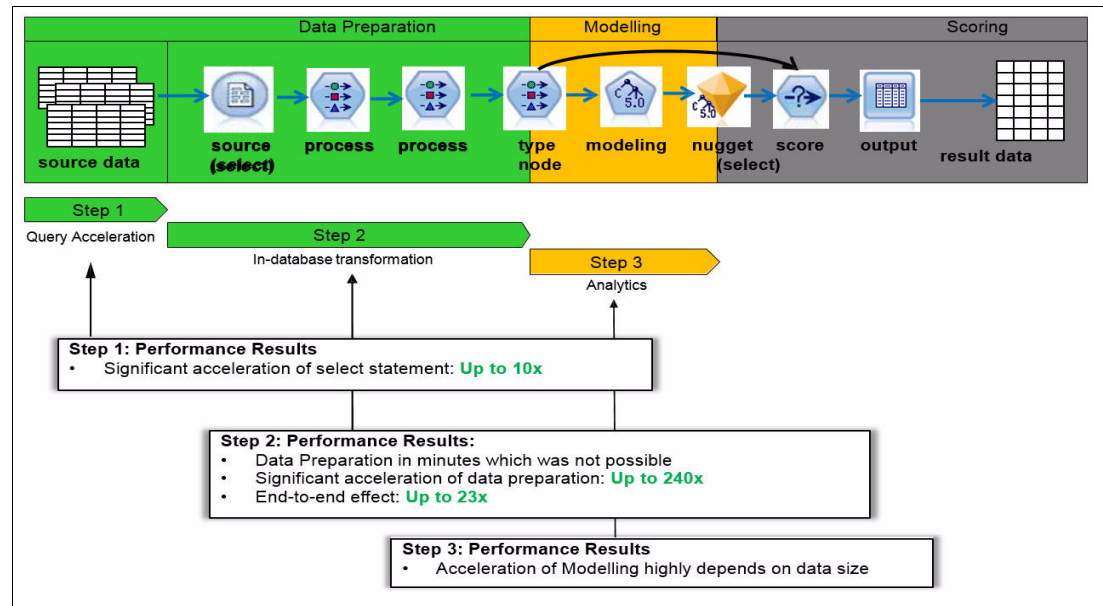


Figure 10-3 SPSS Modeler running analytical functions on an Accelerator

In step 1, we extract the original source data. In step 2, the data preparation phases are completed. Data preparation phases can, for example, include extra data aggregations, calculations, or enrichments of data, which can then be stored as an intermediate result. This transformed or enriched data can then finally be passed on to the modeling process.

The result of the modeling process, which is the data mining model, is then stored in step 3 in a relational table again. Having this information available in DB2 for z/OS enables the data mining tool to score incoming transactions in real-time.

- ▶ **prune_dectree**
Prunes a previously built Decision Tree model.
- ▶ **regtree**
Builds a Regression Tree model by growing and pruning a tree.
- ▶ **grow_regtree**
Builds a Regression Tree model.
- ▶ **prune_regtree**
Prunes a previously built Regression Tree model.
- ▶ **predict_regtree**
Applies a Regression Tree model to generate regression predictions for a dataset.
- ▶ **naivebayes**
Builds a Naive Bayes model.
- ▶ **predict_naivebayes**
Applies a Naive Bayes model to generate classification predictions for a dataset.
- ▶ **kmeans**
Builds a Clustering model that clusters the input data into K centers. The centers are calculated as the mean value of the nearest input data records.
- ▶ **predict_kmeans**
Applies a K-means Clustering model to cluster records of a data set.
- ▶ **two_step**
Builds a TwoStep Clustering model that distributes the input data into a hierarchical tree structure according to the distance between the data records, then reduces the tree into K clusters. A second pass over the data associates the input data records to the next cluster.
- ▶ **predict_twostep**
Applies a TwoStep Clustering model to score records of a data set.
- ▶ **split_data**
Randomly splits the input data into two separated subsets.
- ▶ **pmm1_model** (Predictive Model Markup Language (PMML) model)
Stores the given analytics model as a PMML document to a table.
- ▶ **export_pmm1**
Exports the given analytics model as a PMML document to a file, or it exports a model from a PMML table to a file. If no PMML table exists containing the PMML document for this model, one can be created automatically when requested. Optionally, rather than writing to a file, the result can be returned by the procedure.
- ▶ **model_exists**
Determines if the given model exists. The model can be searched in the current or in another given database.
- ▶ **drop_model**
Drops the given model. All managed tables of this model are also dropped.

Note that these stored procedures are not limited to SPSS usage. You can also call the previously mentioned stored procedures from other applications running in DB2 for z/OS.



Description of IBM z Systems environment used for this publication

This chapter describes the IBM z Systems environment we used for tests and validation of scenarios used in this book.

All systems are based in the IBM Research and Development Lab facility in Germany.

This chapter covers the following topics:

- ▶ IBM DB2 Analytics Accelerator
- ▶ IBM DB2 for z/OS
- ▶ IBM System z system environment
- ▶ IBM InfoSphere Information Server DataStage

A.1 IBM DB2 Analytics Accelerator

We run a N2002-005 (quarter rack) system for the IBM DB2 Analytics Accelerator (Accelerator) with the following characteristics:

- ▶ Accelerator machine version
 - Accelerator name: STRIPER (defined name during pairing with our DB2 for z/OS subsystem)
 - Accelerator server version: 4.1.5.201504221027
 - IBM Netezza Version: 7.2.0.3-P2 [Build 43166] (20150422-1027)
 - Field definition table (FDT) Version: 3.0.5.1
 - Netezza Host Platform (HPF) Version: 5.3.2
 - Incremental Update - Access Server Version: 11.3.3.4289 (20150422-1027)
 - Incremental Update - Replication Engine Version: 11.3.3 [Build BR_LSTYTHIK_96_9] (20150422-1027)
- ▶ IBM DB2 Analytics Accelerator Stored Procedures Version
 - Product version: 4.1.5
 - Stored procedure interface version: 9
 - Build label: 20150323-1646
 - Build time stamp: 2015-03-23 16:50:06
 - Build platform: IBM OS/390® SCLM3 22.00 03 2097
 - Support level: 4
 - DRDA protocol level:
AQT_0000000000040004_0000000000000013_e000000000000000; (AQT10001I)
- ▶ IBM Data Studio and DB2 Analytics Accelerator plug-in
 - Plug-in Version: com.ibm.datatools.aqt 5.1.0.201507020850
 - Connection profile properties:
 - IBM DB2 Universal Database™ (UDB) zSeries
 - V10 (New-Function Mode)
 - com.ibm.db2.jcc.DB2Driver
 - jdbc:db2://tmcc-123-25:446/CC825:retrieveMessagesFromServerOnGetMessage=true;emulateParameterMetaDataForZCalls=1

A.2 IBM DB2 for z/OS

We use DB2 10 for z/OS, DB2 DSN10015.

ZPARM setting

Example A-1 shows the output from SYSPROC.ADMIN_INFO_SYSPARM.

Example A-1 ZPARM output

```
AUDITST=00000000000000000000000000000000
CONDBAT=0000010000
CTHREAD=00200
DLDFREQ=ON
PCLOSEN=00010
IDBACK=00120
IDFORE=00240
CHKTYPE=SINGLE
CHKFREQ=0000500000
CHKLOGR=NOTUSED
CHKMINS=NOTUSED
MON=00000000000000000000000000000000
MONSIZE=0001048576
SYNCVAL=NO
RLFAUTH=SYSIBM
RLF=NO
RLFERR=NOLIMIT
RLFTBL=01
MAXDBAT=00200
DSSTIME=00005
EXTSEC=YES
SMFACCT=10000000000000000000000000000000
SMFSTAT=10111100000000000000000000000000
SMFCOMP=OFF
DEL_CFSTRUCTS_ON_RESTART=NO
ROUTCDE=1000000000000000
STORMXAB=00000
STORTIME=00180
STATIME=00001
TRACLOC=00016
PCLOSET=00010
TRACSTR=00000000000000000000000000000000
TRACTBL=00016
URCHKTH=000
WLMENV=DSNAWLM
LOBVALA=0000010240
LOBVALS=0000002048
XMLVALA=0000204800
XMLVALS=0000010240
PTASKROL=YES
EXTRAREQ=00100
EXTRASRV=00100
TBSBP00L=BPO
IDXBPO0L=BPO
LBACKOUT=AUTO
BACKODUR=005
```

URLGTH=0000000000
UIFCIDS=NO
ACCUMACC=00010
ACCUMUID=00000
TSQTY=0000000000
IXQTY=0000000000
DSVCI=YES
MGEXTSZ=YES
SMF89=NO
OTC_LICENSE=NOT_USED
MAXOFILR=0000000100
TBSBP8K=BP8K0
TBSBP16K=BP16K0
TBSBP32K=BP32K
TBSBPLOB=BP0
TBSBPXML=BP16K0
IMPDSDEF=YES
IMPTSCMP=NO
IMPTKMOD=YES
IMPDSIZE=00004
DPSEGSZ=00032
LOB_INLINE_LENGTH=00000
DSNZPARM=DSNZPARM
ACCESS_CNTL_MODULE=DSNX@XAC
IDAUTH_MODULE=DSN3@ATH
SIGNON_MODULE=DSN3@SGN
XML_RANDOMIZE_DOCID=NO
TWOACTV=1
OFFLOAD=YES
TWOBSDS=2
TWOARCH=1
MAXARCH=0000010000
DEALLCT=(0000,00)
MAXRTU=00002
OUTBUFF=0000004000
ARC2FRST=NO
REMOTE_COPY_SW_ACCEL=DISABLE
BLKSIZE=0000024576
CATALOG=NO
ALCUNIT=BLK
PROTECT=NO
ARCWTOR=YES
COMPACT=NO
TSTAMP=NO
QUIESCE=00005
ARCRETN=09999
ARCPFX1=DSNA.ARCHLOG1
ARCPFX2=DSNA.ARCHLOG2
PRIQTY=0000001440
SECQTY=0000000180
UNIT=TAPE
UNIT2=
SVOLARC=NO
ARCWRTC=1011000000000000
MSVGP=

MSVGP2=
ABIND=YES
SYSADM2=PURIT
AUTHCACH=03072
AUTH=YES
BMPTOUT=00004
LEMAX=00020
BINDNV=BINDADD
CDSSRDEF=1
DBCHK=NO
DEFLTID=IBMUSER
CHGDC AND EDPROP=1
DECDIV3=NO
DLITOUT=00006
DSMAX=0000020000
EDMPOOL=0000000000
RECALLD=00120
RECALL=YES
IRLMAUT=YES
ABEXP=YES
IRLMPC=DSNAIRLM
IRLMSID=IRLM
IRLMSWT=0000000120
NUMLKTS=0000001000
NUMLKUS=0000010000
SEQCACH=BYPASS
RRULOCK=NO
DESCSTAT=YES
SEQPRES=NO
CACHEDYN=YES
RETLWAIT=00000
CACHERAC=0005242880
CONTSTOR=YES
MAXKEEPD=0000005000
PROTOFF=NO
AEXITLIM=00010
RETVLCFK=NO
OJPERFEH=YES
SYSOPR1=BUWD
SYSOPR2=PURIT
CACHEPAC=0005242880
PARA_EFF=00050
PARAMDEG=0000000000
STATHIST=NONE
RGFNMPT=DSN_REGISTER_APPL
RGFCOLID=DSNRGCOL
RGFESCP=
RGFINSTL=NO
RGFDEDPL=NO
RGFFULLQ=YES
RGFDEFLT=ACCEPT
RGFDBNAM=DSNRGFDB
RGFNMORT=DSN_REGISTER_OBJT
MAXRBLK=0000400000
MINRBLK=0000000001

SYSADM=BUWD
SRTPOOL=0000010000
IRLMRWT=0000000060
SITETYP=LOCALSITE
UTIMOUT=00006
XLKUPDLT=NO
OPTHINTS=NO
TRKRSITE=NO
NPGTHRSH=0000000000
SJTABLES=00010
STARJOIN=DISABLE
SMSDCF=NO
SMSDCIX=NO
DBACRVW=NO
SUPERRS=YES
STATROLL=YES
MINSTOR=NO
EVALUNC=NO
STATSINT=00030
MINDVSCL=NONE
MXTBJOIN=00225
EDMDBDC=0000023400
EDMSTMTC=0000115836
EDM_SKELETON_POOL=0000009028
REFSHAGE=0
MAINTYPE=SYSTEM
MAX_OPT_STOR=00040
MAX_OPT_CPU=00100
LRDRTHLD=00010
VOLTDEVT=SYSDA
DISABSCL=NO
SKIPUNCI=NO
MXQBCE=0000001023
PADIX=NO
SMGE=NO
SECLCACH=00255
MXDTCACH=00020
MAX_NUM_CUR=0000000500
MAX_ST_PROC=0000002000
INLISTP=0000000050
MAXTEMPS=0000000000
WFDBSEP=NO
SYSTEM_LEVEL_BACKUPS=NO
RESTORE_RECOVER_FROMDUMP=NO
UTILS_DUMP_CLASS_NAME=NO
RESTORE_TAPEUNITS=NOLIMIT
UTIL_TEMP_STORCLAS=NO
OPTXQB=ON
RPITWC=YES
UNION_COLNAME_7=NO
PTCDIO=OFF
OPTIXIO=ON
STATCLUS=ENHANCED
UTSORTAL=YES
IGNSORTN=NO

DB2SORT=DISABLE
 CACHEDYN_FREELocal=00001
 PLANMGMT=EXTENDED
 PLANMGMTSCOPE=STATIC
 MORE_UNION_DISTRIBUTION=OFF
 MAX_CONCURRENT_PKG_OPS=00010
 ADMTPROC=DSNAADMT
 HONOR_KEEPDICtIONARY=NO
 INDEX_IO_PARALLELISM=YES
 CATDDACL=
 CATDMGCL=
 CATDSTCL=
 CATXDACL=
 CATXMGCL=
 CATXSTCL=
 ZOSMETRICS=NO
 FCCOPYDDN=HLQ.&DB..&SN..N&DSNUM..&UQ.
 FLASHCOPY_COPY=NO
 FLASHCOPY_LOAD=NO
 FLASHCOPY_REORG_TS=NO
 FLASHCOPY_REBUILD_INDEX=NO
 FLASHCOPY_REORG_INDEX=NO
 SECADM1=BUWD
 SECADM1_TYPE=AUTHID
 SECADM2=PURIT
 SECADM2_TYPE=AUTHID
 SEPARATE_SECURITY=NO
 REVOKE_DEP_PRIVILEGES=SQLSTMT
 OPTIOWGT=ENABLE
 COMPRESS_SPT01=NO
 EN_PJSJ=OFF
 RRF=ENABLE
 CHECK_SEtCHKP=NO
 MAXTEMPS_RID=NOLIMIT
 CHECK_FASTREPLICATION=PREFERRED
 REC_FASTREPLICATION=PREFERRED
 FLASHCOPY_PPRC=
 BIF_COMPATIBILITY=V9_DECIMAL_VARCHAR
 SPT01_INLINE_LENGTH=32138
 SIMULATED_CPU_COUNT=OFF
 SIMULATED_CPU_SPEED=OFF
 REALSTORAGE_MAX=NOLIMIT
 REALSTORAGE_MANAGEMENT=AUTO
 DDLTOX=00001
 DISABLE_EDMRTS=NO
 DISALLOW_DEFAULT_COLLID=NO
 IX_TB_PART_CONV_EXCLUDE=NO
 RESTRICT_ALT_COL_FOR_DCC=NO
 REORG_IGNORE_FREESPACE=NO
 ACCEL=COMMAND
 QUERY_ACCELERATION=NONE
 REORG_LIST_PROCESSING=PARALLEL
 OPT1ROWBLOCKSORT=DISABLE
 REORG_PART_SORT_NPSI=NO
 QUERY_ACCEL_OPTIONS=1,2,3

GET_ACCEL_ARCHIVE=NO
SUBQ_MIDX=DISABLE
OPTIOPIN=ENABLE
PGRNGSCR=NO
CACHE_DEP_TRACK_STOR_LIM=0000000002
XML_RESTRICT_EMPTY_TAG=NO
ACCELMODEL=YES
PREVENT_NEW_ICTRL_PART=NO
SQWIDSC=NO
SUPPRESS_HINT_SQLCODE_DYN=NO
TEMPLATE_TIME=UTC
RETRY_STOPPED_OBJECTS=NO
CATALOG=DSNA
RESTART_DEF=RESTART
ALL-DBNAME=ALL
CMTSTAT=INACTIVE
TCPALVER=NO
RESYNC=00002
RLFERRD=NOLIMIT
DDF=AUTO
IDHTOIN=00120
MAXTYPE1=0000000000
TCPKPALV=00120
POOLINAC=00120
SQLINTRP=ENABLE
PRIVATE_PROTOCOL=AUTH
MAXCONQN=OFF
MAXCONQW=OFF
DDF_COMPATIBILITY=NULL
ASSIST=NO
COORDNTR=NO
IMMEDWRI=NO
DSHARE=NO
MEMBNAME=DSN1
GRPNAME=DSNCAT
RANDOMATT=YES
DB2SUPLD=NO
DEFLANG=IBMCOB
DECIMAL=PERIOD
DELIM=DEFAULT
SQLDELI=DEFAULT
DSQLDELI=APOST
MIXED=NO
SCCSID=00037
MCCSID=65534
GCCSID=65534
ASCCSID=00819
AMCCSID=65534
AGCCSID=65534
USCCSID=00367
UMCCSID=01208
UGCCSID=01200
ENSCHHEME=EBCDIC
APPENSCH=EBCDIC
NEWFUN=V10

DECARTH=DEC15
DYNRULS=YES
DATE=ISO
TIME=ISO
DATELEN=000
TIMELEN=000
IMPLICIT_TIMEZONE=CURRENT
STDSQL=NO
PADNTSTR=NO
CHARSET=ALPHANUM
SSID=DSNA
LC_CTYPE=
DEF_DECFLOAT_ROUND_MODE=ROUND_HALF_EVEN
PC SPECIFIED=YES
WAIT TIME FOR LOCAL DEADLOCK=00001
LOCAL CYCLES PER GLOBAL CYCLE=00001
TIMEOUT INTERVAL=0000000060
IRLM MAXIMUM CSA USAGE ALLOWED=0000000000
SYSTEM LOCK TABLE HASH ENTRIES=0000000000
PENDING NUMBER OF HASH ENTRIES=0000000000
SYSTEM LOCK TABLE LIST ENTRIES=0000000000

A.3 IBM System z system environment

The IBM z/OS System was set up in an IBM z/VM environment, to have more flexibility in assign and secure the IBM System z system resources. In cases of scalability and proof of concept (PoC) tests, the environment is copied or switched into a dedicated performance sphere.

Figure A-1 shows the lab System z system environment.

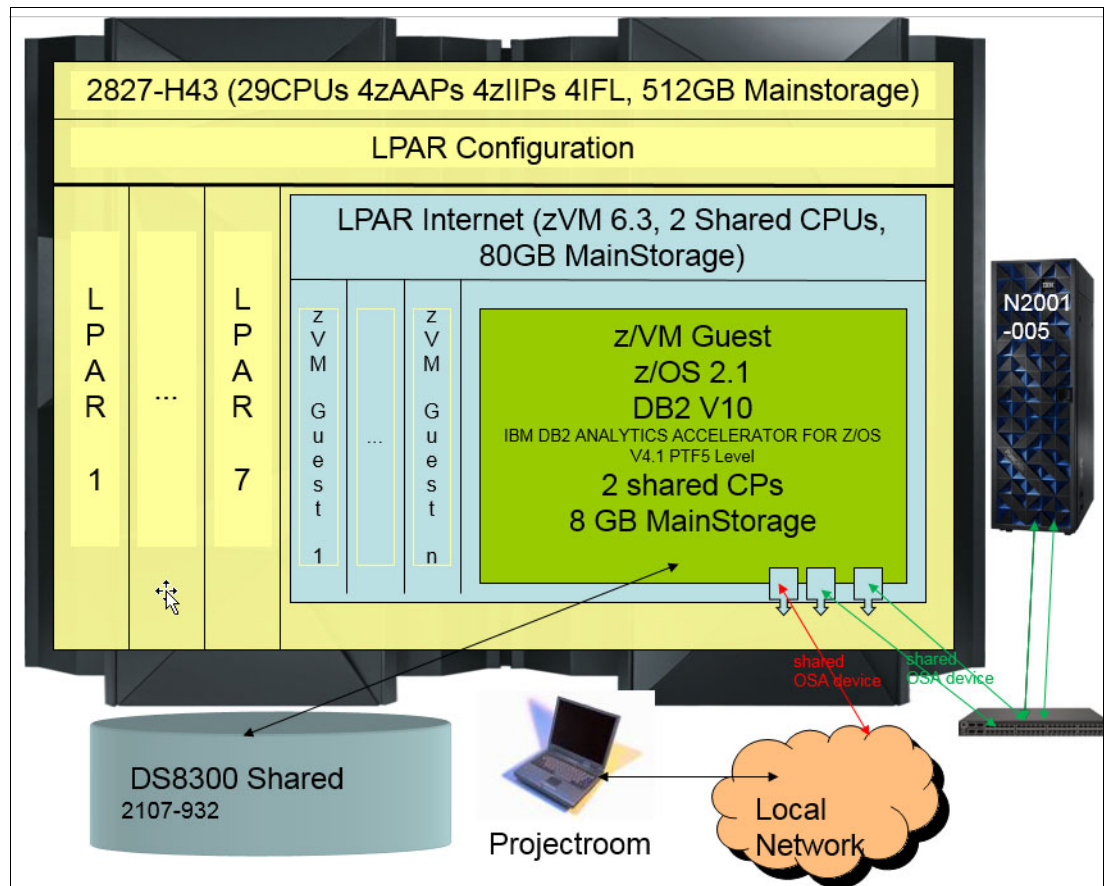


Figure A-1 System z system environment

The following list describes the configuration of the IBM z/OS system in more details.

- ▶ System z server: z/VM z/OS guest on an IBM zEnterprise system, a single two-book IBM zEnterprise EC12 (zEC12) H43 with 512 gigabytes (GB) memory and 43 processors:
 - 29 processors
 - 4 System z Application Assist Processors (ZAAPs)
 - 4 System z Integrated Information Processors (ZIIPs)
 - 4 Integrated Facilities for Linux (IFLs)
 - 2 integrated catalog facilities (ICFs)

The z/OS Transmission Control Protocol/Internet Protocol (TCP/IP) used three Open Systems Adapter (OSA) adapters, two for the Netezza connection (OSA-Express4S 10 gigabits per second (Gbps) Ethernet SR) and one for the local network (OSA-Express4S 1000BASE-T Ethernet).

For the DS8K connection, we have four Fibre Channel connection (IBM FICON®) channels on different IBM FICON Express.

- ▶ DS8300-932 was used for the shared environment with a total Space of 14 terabytes (TB).
The z/OS System used about 150 GB IBM ECKD™ Space. Four x FICON 4 Gbps input/output (I/O) ports are used to connect the zEnterprise system.
- ▶ IBM DB2 Analytics Accelerator runs on the IBM PureData System for Analytics N2001-005. The current software level shows in Figure A-2.
The z/OS system is connected to two 10 gigabit Ethernet (GbE) dual port adapters.
- ▶ z/OS (z/VM guest) software components:
 - z/OS V2.1 Service level: program update tape (PUT)1505 with some additional program temporary fixes (PTFs)
 - DB2 10 Service level: PUT1503 with some additional PTFs
 - IBM DB2 Analytics Accelerator V4 Service level: PTF-5 (UI27257,UI27269,UI27301-3, UI30667, UI27386-7)
 - IBM DB2 Query Management Facility (QMF): HSQBB20

▼ About			
z/OS			
Stored Procedures:	4.1.5.20150323-1646		
Server	Transfer updates	Remove	Apply other software version
Accelerator server:	4.1.5.201504221027	Netezza Performance Server (NPS):	7.2.0.3-P2 [Build 43166] (20150422-1027)
Netezza Firmware (FDT):	3.0.5.1	Netezza Host Platform (HPF):	5.3.2
Access server:	11.3.3.4289 (20150422-1027)	Replication Engine:	11.3.3 [Build BR_LSTYTHIK_96_9] (20150422-1027)
Client			
Studio:	4.1.5.201411182134	Check for Updates	

Figure A-2 Accelerator Software levels listed on Data Studio view

A.4 IBM InfoSphere Information Server DataStage

The following items are on the InfoSphere Information Server DataStage server:

- ▶ IBM InfoSphere Information Server Version 11.3.1.1
- ▶ InfoSphere Information Server 11.3/11.3.1 Fixpack 1 (fixpack.is_11_3_1_1.windows.RC2.2.150306)
- ▶ IBM WebSphere 8.5.5.4
- ▶ Metadata repository: "DB2 v10.5.0.5", "s141128", "IP23633", and Fix Pack 5 ("64" bits and DB2 code release "SQL10055" with level identifier "0606010E")
- ▶ Data Direct Open Database Connectivity (ODBC) driver version: 07.10.0022 (B0028, U0020)

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only:

- ▶ *Complete Analytics with IBM DB2 Query Management Facility: Accelerating Well-Informed Decisions Across the Enterprise*, SG24-8012
- ▶ *Reliability and Performance with IBM DB2 Analytics Accelerator V4.1*, SG24-8213
- ▶ *Reliability and Performance with IBM DB2 Analytics Accelerator V4.1*, SG24-8213
- ▶ *Hybrid Analytics Solution using IBM DB2 Analytics Accelerator for z/OS V3.1*, SG24-8151

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, drafts, and additional materials, at the following website:

ibm.com/redbooks

Other publications

These publications are also relevant as further information sources:

- ▶ Wes McKinney, *Python for Data Analysis*, O'Reilly, 2013, ISBN 978-1-449-31979-3
- ▶ Fernando Pérez, Brian E. Granger, *IPython: A System for Interactive Scientific Computing*, Computing in Science and Engineering, vol. 9, no. 3, pp. 21-29, May/June 2007, doi:10.1109/MCSE.2007.53. URL: <http://ipython.org>
- ▶ *IBM DB2 Analytics Accelerator for z/OS V4.1.0 User's Guide*, SH12-7040

Online resources

These websites are also relevant as further information sources:

- ▶ IBM DB2 Analytics Accelerator for z/OS
<http://www.ibm.com/software/products/en/db2analacceforzos>
- ▶ IBM DB2 for z/OS
<http://www.ibm.com/software/data/db2/zos/db2-10-zos/>
- ▶ "IBM InfoSphere Information Server architecture and concepts"
http://www.ibm.com/support/knowledgecenter/SSZJPZ_11.3.0/com.ibm.swg.im.iis.production.iisinfsv.overview.doc/topics/cisoarchoverview.html

- ▶ “IBM InfoSphere Information Server, Best Practices Topology”
https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/Wc7a4f8c1c8ac_4813_9e13_1f2836f1287d/page/Designing+a+topology+for+InfoSphere+Information+Server
- ▶ “Processing stage types” (for InfoSphere Balance Optimization)
https://www.ibm.com/support/knowledgecenter/SSZJPZ_11.3.0/com.ibm.swg.im.iis.ds.parjob.dev.doc/topics/limitationsstagetypes.html
- ▶ “Supported functions in SQL”
https://www.ibm.com/support/knowledgecenter/SSZJPZ_11.3.0/com.ibm.swg.im.iis.ds.parjob.dev.doc/topics/limitationsfunctions.html
- ▶ “TPC Benchmark H, Standard Specification”
<http://www.tpc.org/tpch/spec/tpch2.1.0.pdf>

Help from IBM

IBM Support and downloads

[ibm.com/support](https://www.ibm.com/support)

IBM Global Services

[ibm.com/services](https://www.ibm.com/services)



SG24-8314-00

ISBN 0738441198

Printed in U.S.A.

Get connected

