Redbooks

ibm.com/redbooks

Enabling Real-time Analytics on IBM z Systems Platform

Lydia Parziale

Oliver Benke

Willie Favero

Ravi Kumar

Steven LaFalce

Cedrine Madera

Sebastian Muszytowski

Analytics

z Systems

IBM

Redbooks

International Technical Support Organization

Enabling Real-time Analytics on IBM z Systems Platform

August 2016

SG24-8272-00

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

First Edition (August 2016)

This edition applies to IBM DB2 Analytics Accelerator for z/OS v5.1.

© Copyright International Business Machines Corporation 2016. 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
Now you can become a published author, too! xiii
Comments welcome xiii
Stay connected to IBM Redbooks xiv

Chapter 1. Executive overview 1
1.1 Introduction 2
1.2 Real-time analytics 4
1.2.1 Business advantages 5
1.2.2 IT advantages 5
1.3 In-database analytics 5
1.3.1 Accelerated in-database transformation 6
1.3.2 Accelerated in-database predictive modeling 6
1.4 Enabling applications with machine learning capability 6
1.5 Value propositions 7
1.6 Related products 7
1.7 Use cases 8
1.7.1 Countering payment fraud and financial crimes 8
1.7.2 Insurance claims in-process payment analytics 9
1.7.3 Predictive customer intelligence 10

Chapter 2. Analytics implementation on z Systems platform 11
2.1 Adding analytics to a mainframe data sharing environment 12
2.1.1 SPSS Modeler 15
2.1.2 DB2 Analytics Accelerator wrapper stored procedure 16
2.1.3 Using accelerator-only tables 18
2.2 Installation and customization 19
2.2.1 DB2 and DB2 Analytics Accelerator setup and installation 20
2.2.2 Required DB2 privileges for SPSS users 24
2.2.3 SPSS Modeler client 25
2.2.4 SPSS Modeler server 32
2.2.5 Data sources in SPSS 33
2.2.6 AQT_ANALYTICS_DATABASE variable 33
2.2.7 User management in the SPSS Modeler server 33
2.2.8 Installing SPSS Modeler scoring adapter for DB2 z/OS 34
2.3 Real-time analytics lifecycle 34
2.3.1 Swim lane diagram of in-database analytics lifecycle 37
2.3.2 Interaction between a DB2 DBA and a data scientist 37
2.3.3 Key strengths of various components 38

Chapter 3. Data integration using IBM DB2 Analytics Accelerator Loader for z/OS . 41
3.1 Functional overview 42
3.1.1 Loader v2.1 enhancements 43

© Copyright IBM Corp. 2016. All rights reserved.

3.1.2 Loader methods to move data 44
3.1.3 Components and interfaces 44
3.2 Getting started 46
3.2.1 Installation 46
3.2.2 Customization 46
3.2.3 Workload Management (WLM) performance goals 47
3.2.4 IBM z Systems Integrated Information Processor (zIIP) 53
3.2.5 z Systems advantage 56
3.2.6 Parallelism 57
3.3 Scenarios 57
3.3.1 ACCEL_LOAD_TASKS 58
3.3.2 Sequential Input IDAA_ONLY and IDAA_DUAL 58
3.3.3 Load RESUME 64
3.3.4 IBM DB2 Analytics Accelerator Loader image copy input 67
3.3.5 VSAM 68
3.4 System Management Facility (SMF) 77

Chapter 4. Data transformation 83
4.1 Introduction 84
4.1.1 Accelerator-only table (AOT) 85
4.1.2 Enabling in-database processing on SPSS Modeler client 86
4.2 SQL pushback in SPSS Modeler 86
4.2.1 How SQL generation works 86
4.2.2 Where improvements can occur with IDT using Accelerator 87
4.3 Nodes supporting SQL generation for DB2 Accelerator 88
4.3.1 Source palette tab 88
4.3.2 Record Ops palette tab 89
4.3.3 Field Ops palette tab 90
4.3.4 Graphs palette tab 92
4.3.5 Database Modeling (Nuggets) palette tab 93
4.3.6 Output palette tab 94
4.3.7 Export palette tab 94
4.4 In-database analytics Processing effort by components 95
4.4.1 SPSS Modeler client (running on Windows) 95
4.4.2 SPSS Modeler server 95
4.4.3 DB2 for z/OS with Analytics Accelerator 96
4.4.4 Netezza based DB2 Analytics Accelerator for z/OS 96
4.5 SPSS data transformation stream 96
4.5.1 SQL Preview option to verify in-database processing 100
4.5.2 Why are my nodes purple 100
4.5.3 Enable cache at node level 100
4.6 Stream messages 101
4.7 Data transformation using DataStage 105

Chapter 5. Data modeling using SPSS and DB2 Analytics Accelerator 107
5.1 Introduction 108
5.2 SPSS data modeling stream design 108
5.2.1 SPSS data modeling stream 108
5.2.2 Example of how to optimize pruning a decision tree 112
5.3 Adding record IDs to input tables 115
5.4 Combining in-database transformation on Accelerator with traditional model creation 116

Chapter 6. Model scoring 117
6.1 In-transactional scoring (single-record scoring) 118
6.2 In-database scoring and scoring adapter 119
6.2.1 Real-time scoring and near real-time scoring 119
6.2.2 Export PMML from SPSS model apply node 122
6.3 Batch scoring 127
6.3.1 Complete batch scoring example with accelerated in-database predictive modeling 127
6.3.2 Batch scoring using Scoring Adapter into accelerator-only tables (less efficient method) 128
6.3.3 Batch scoring using Scoring Adapter into z/OS DB2 base tables (more efficient method) 132
6.4 DMG and PMML 135
6.4.1 Zementis and in-application scoring 136
6.5 Scoring and smart decisions 136
6.5.1 Solution architecture 136
6.5.2 Operational Decision Manager overview 138
6.5.3 SPSS Collaboration and Deployment Services 139
6.5.4 SPSS Scoring Adaptor for DB2 z/OS 139
6.6 Examples of integration for scoring solutions using ODM on z/OS 140
6.6.1 Example: Calling the SPSS Scoring Adapter for IBM DB2 for z/OS 140
6.6.2 Example: Calling SPSS Collaboration and Deployment Services 141

Chapter 7. Acceleration for descriptive analytics: IBM DB2 Analytics Accelerator value with IBM Cognos BI 143
7.1 Descriptive analytics 144
7.2 IBM Cognos Business Intelligence 144
7.3 IBM Cognos BI 10 and DB2 Analytics Accelerator and z13 145

Chapter 8. Using R with the IBM DB2 Analytics Accelerator 149
8.1 Introduction to the R programming language 150
8.2 Prerequisites 150
8.3 Using RJDBC with the IBM DB2 Analytics Accelerator 151
8.3.1 RJDBC functions 153
8.4 Using RODBC with the IBM DB2 Analytics Accelerator 154
8.4.1 RODBC functions 156
8.5 Using accelerator-only tables (AOTs) with R 157
8.6 Using Netezza Analytics stored procedures 159
8.6.1 Basic concept 160
8.6.2 Helper stored procedures 161
8.6.3 Decision trees 162
8.6.4 Regression trees 162
8.6.5 K-means clustering 163
8.6.6 TwoStep clustering 163
8.6.7 Naive Bayes 163
8.7 Performance considerations 164

Appendix A. ODBC connectivity between SPSS Modeler and DB2 for z/OS 165
ODBC related terms and concepts 166
ODBC Driver Manager 166
ODBC gateway mode 167
ODBC driver used by SPSS Modeler server 170
Installation and configuration of IBM DB2 ODBC driver 171
Base installation and DB2 instance create 171

Starting DB2 171
DB2 communication database and BIND 171
DB2 Connect license 173
SPSS Modeler configuration for use of DB2 ODBC driver. 173
Installation and configuration of SPSS SDAP driver 174

Appendix B. Job to install wrapper stored procedures. 179
JCL job sample 180
IBM Netezza Analytics stored procedures 190
K-means 190
Naive Bayes 191
Decision Tree 192
Regression Tree 193
TwoStep 194

Related publications 197
IBM Redbooks 197
Other publications 197
Online resources 197
Help from IBM 198

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US

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 jurisdictions 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 provide in any way it believes appropriate without incurring any obligation to you.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

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.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

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 actual people or business enterprises 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. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

© Copyright IBM Corp. 2016. All rights reserved.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

BigInsights®
CICS®
Cognos®
DataStage®
DB2®
DB2 Connect™
FlashCopy®
GDPS®
HiperSockets™
IBM®
IBM PureData®
IBM z Systems™
IBM z13™
IMS™
InfoSphere®
MVS™
PureData®
RACF®
Redbooks®
Redbooks (logo) ®
RMF™
SPSS®
System z®
WebSphere®
z Systems™
z/OS®
z13™
zEnterprise®

The following terms are trademarks of other companies:

Netezza, Netezza Performance Server, NPS, 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 personalized notifications of new content
- Link to the latest Redbooks blogs and videos

Get the latest version of the Redbooks Mobile App

iOS **Download Now** Android

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

Regarding online transaction processing (OLTP) workloads, IBM® z Systems™ platform, with IBM DB2®, data sharing, Workload Manager (WLM), geoplex, and other high-end features, is the widely acknowledged leader. Most customers now integrate business analytics with OLTP by running, for example, scoring functions from transactional context for real-time analytics or by applying machine-learning algorithms on enterprise data that is kept on the mainframe. As a result, IBM adds investment so clients can keep the complete lifecycle for data analysis, modeling, and scoring on z Systems control in a cost-efficient way, keeping the qualities of services in availability, security, reliability that z Systems solutions offer. Because of the changed architecture and tighter integration, IBM has shown, in a customer proof-of-concept, that a particular client was able to achieve an orders-of-magnitude improvement in performance, allowing that client's data scientist to investigate the data in a more interactive process.

Open technologies, such as Predictive Model Markup Language (PMML) can help customers update single components instead of being forced to replace everything at once. As a result, you have the possibility to combine your preferred tool for model generation (such as SAS Enterprise Miner or IBM SPSS® Modeler) with a different technology for model scoring (such as Zementis, a company focused on PMML scoring). IBM SPSS Modeler is a leading data mining workbench that can apply various algorithms in data preparation, cleansing, statistics, visualization, machine learning, and predictive analytics. It has over 20 years of experience and continued development, and is integrated with z Systems. With IBM DB2 Analytics Accelerator 5.1 and SPSS Modeler 17.1, the possibility exists to do the complete predictive model creation including data transformation within DB2 Analytics Accelerator. So, instead of moving the data to a distributed environment, algorithms can be pushed to the data, using cost-efficient DB2 Accelerator for the required resource-intensive operations.

This IBM Redbooks® publication explains the overall z Systems architecture, how the components can be installed and customized, how the new IBM DB2 Analytics Accelerator loader can help efficient data loading for z Systems data and external data, how in-database transformation, in-database modeling, and in-transactional real-time scoring can be used, and what other related technologies are available.

This book is intended for technical specialists and architects, and data scientists who want to use the technology on the z Systems platform. Most of the technologies described in this book require IBM DB2 for z/OS®. For acceleration of the data investigation, data transformation, and data modeling process, DB2 Analytics Accelerator is required. Most value can be archived if most of the data already resides on z Systems platforms, although adding external data (like from social sources) poses no problem at all.

If you want only an overview of enabling real-time analytics on IBM z Systems platforms and how it might fit into your organization, read Chapter 1, "Executive overview" on page 1.

© Copyright IBM Corp. 2016. All rights reserved.

Authors

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

Lydia Parziale is a Project Leader for the ITSO team in Poughkeepsie, New York, with domestic and international experience in technology management including software development, project leadership, and strategic planning. Her areas of expertise include business development and database management technologies. Lydia is a certified PMP and an IBM Certified IT Specialist with an MBA in Technology Management and has been employed by IBM for over 25 years in various technology areas.

Oliver Benke is a Software Engineer at the IBM Laboratory in Boeblingen, Germany where for 17 years he has worked on various projects. In the last several years, he has developed prototypes for SPSS Modeler Scoring Adapter for z Systems, DB2 Analytics Accelerator accelerator-only tables and IBM Netezza® Analytics integration for in-database modeling, including related performance work, evaluations, and early customer validation.

Willie Favero was an IBM Senior Certified IT Software Specialist and subject matter expert (SME) on DB2 for z/OS with the IBM Silicon Valley Lab z Systems Analytics SWAT Team. He has 40 years of experience working with databases, including 31 years working with DB2. Willie is a highly sought-after speaker for international conferences and user groups. He also publishes articles and white papers, contributes to numerous IBM Redbooks publications, and has a top technical (DB2) blog on the Internet.

Ravi Kumar is a Senior Managing Consultant at IBM (Analytics Platform, North American Lab Services). Ravi is a Distinguished IT Specialist (Open Group certified) with more than 23 years of IT experience. He has a Master's in Business Administration (MBA) degree from University of Nebraska, Lincoln. He has contributed to seven other Redbooks publications, in the areas of Database, Analytics Accelerator, and Information Management tools.

Steven LaFalce started his career at IBM in 2002 working on design, testing, and qualification of z Systems memory subsystems. He later joined z Systems Hardware Product Engineering working on the central electronic complex subsystem. In his current role, he is working in z Systems Field Technical Enablement on business analytics projects.

Cedrine Madera is a Senior Information Architect at the IBM Montpellier client center, engaged as a subject matter expert on worldwide projects for analytics, big data, data lake, and data governance. She has held various client-facing analytics consulting practices and projects before becoming a lead in the world Analytics zChampion group. She has 18 years of experience in information technology and analytics methods.

Sebastian Muszytowski is a Software Engineer with the IBM DB2 Analytics Accelerator team in the IBM Research and Development Lab in Boeblingen, Germany. His work centers around enabling and supporting local and international customers in building and maintaining enterprise data warehousing solutions with the IBM DB2 Analytics Accelerator. Sebastian has been employed by IBM for five years.

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

Paul T. DiMarzio, William F. Henderson, Abuchi Obiegbu, James Perlik, Bruce Smilowitz, James D. Thomasson, Dino Tonelli, Mythili K. Venkatakrishnan
IBM USA

Guillaume Arnould, Marie-Laure Pessoa Caucat, Stephane Faure, Fabrice Jarassat, Yann Kindelberger
IBM France

Dr. Toni Bollinger, Jianguo Chen, Reinhold Geiselhart, Michael Haide, Erik Hassold, Claus Kempfert, Steffen Knoll, Erik Seiffert, Roland Seiffert, Dr. Knut Stolze, Dr. Otto Wohlmuth
IBM Germany

Dr. Niall Mccarroll
IBM UK

Yi YS Shao, Fu Ju A An
IBM China

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.

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

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 found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

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

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

1

Executive overview

This overview describes how IBM DB2 Analytics Accelerator for z/OS opens a new dimension of analytical processing when integrated with IBM SPSS Modeler and highlights the power of the IBM DB2 Analytics Accelerator Loader for z/OS v2.1 tool in integrating data from sources other than z/OS without increasing your MIPS consumption on z Systems. The overview also discusses the business and IT benefits of enabling predictive analytics on your most reliable mainframe platform without compromising security.

This chapter contains the following topics:

- Introduction
- Real-time analytics on z Systems operational data
- In-database analytics using SPSS and DB2 Analytics Accelerator
- Enabling applications with machine learning capability
- Value propositions
- Related products
- Use cases

© Copyright IBM Corp. 2016. All rights reserved.

1.1 Introduction

Businesses that rely on data as part of their daily operations are deploying solutions that capture data, process them, and calculate predictions for intelligent decision making to provide customized service to their customers. An essential component to accomplishing this service is understanding the opportunities and risks at the granular level of each individual customer interaction while the interaction is happening. Accordingly, key considerations for these solutions are the richness of analysis and the ability to maintain service level agreements (SLAs) without data breach.

The IBM z Systems™ platform addresses these considerations and provides solutions that produce actionable results in real time. Figure 1-1 shows the technologies and tools presented in this book. The top shows a highly scalable online transaction processing (OLTP) system; the middle shows IBM SPSS Modeler with Scoring Adapter for DB2 z/OS; the bottom shows IBM DB2 for z/OS database engine with DB2 Analytics Accelerator for z/OS appliance to demonstrate a real-time analytics use case that can be applied to business scenarios across industries.

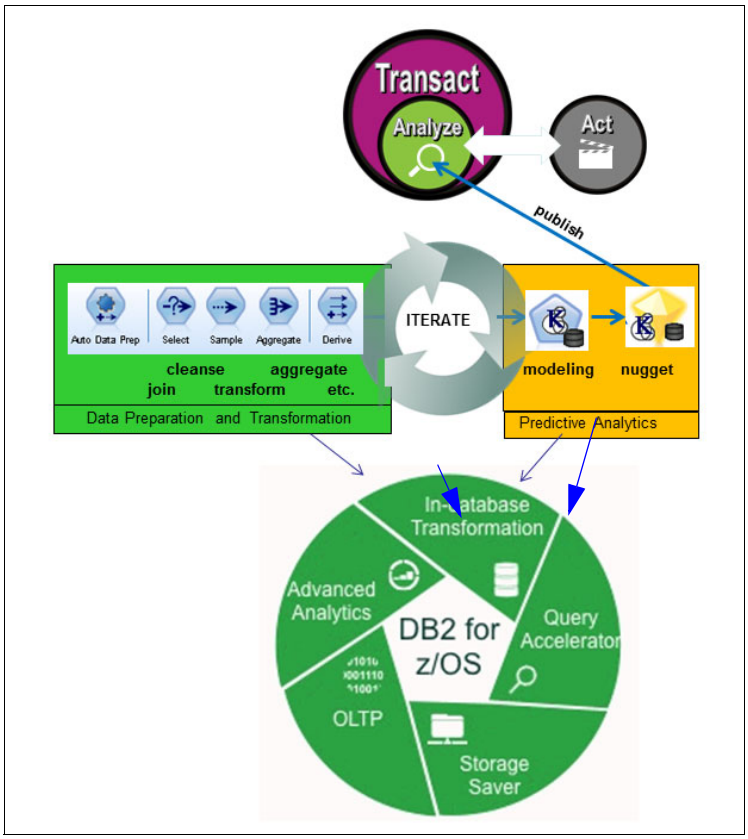


Figure 1-1 Predictive modeling for OLTP applications with DB2 Analytics Accelerator for z/OS

In this case, the platform centers around DB2 for z/OS, with best-in-class data lifecycle management support for fighting fraud, preventing financial crimes, and generating customer insights. Although most of the valuable data already resides on z Systems mainframe, another possibility is to consume additional data that does not originate on the mainframe, such as social data (using the IBM DB2 Analytics Accelerator Loader for z/OS tool, which is described in Chapter 3, “Data integration using IBM DB2 Analytics Accelerator Loader for z/OS” on page 41).

For critical data, businesses want to retain control of the data so it cannot be compromised, another area where the z Systems platform is in a clear lead position. To get valuable insights from that data, data scientists analyze vast amounts of data in an iterative process, experimenting and looking for an optimal solution. After a predictive model is created and validated by the data scientist, it is deployed to the operational system that can use the model for in-database or in-application real-time scoring. Many installations evaluate the scored results against their business rules, although in some simple cases, reacting on the score values directly in the transactional application code might be appropriate.

The z Systems hardware and software stack is designed for mission-critical systems that are highly available and can meet stringent SLAs. It is highly optimized for efficiency and extreme scalability. The traditional z Systems values greatly help the OLTP workload, where real-time scoring of predictive models is tightly integrated. Although building new predictive models and predictive or prescriptive analytics work of data scientists to mine the data for new insights does not have the same requirements for scalability and is not something guarded by SLAs, it might soon become mission-critical. However, for this analytical work, which usually consumes vast amounts of data, the requirement is usually that the data be kept under z Systems governance, with the high level of security that the platform provides.

One of the key innovations in IBM DB2 Analytics Accelerator version 4.6 (with in-database transformation) and version 5.1 (with in-database modeling) is that it allows for pushing the processing to where the (accelerated) data resides—that is, in to the Accelerator. As for reporting, a more appropriate way is to do reporting on a consistent snapshot of the data. If the workload requires near real-time data also for model creation and reporting in order to rapidly discover new trends, then change data capture (CDC) replication technology that is part of the DB2 Analytics Accelerator solution can be used.

A customer study, using an early prototype of the technology now productized in DB2 Analytics Accelerator 5.1, revealed that with the SPSS streams that this retail customer had in place, achieving performance improvements by a factor of 200 was possible just for the data transformation, pushing stream execution times down from hours to seconds (see Figure 1-2 on page 4). Together with modeling and batch scoring, this customer was able to realize a 20x overall performance improvement, allowing the data scientists to more interactively experiment with the data as the analytical model, which previously took hours to compute was made available for validation in minutes. With in-database processing, a slight change in data preparation can show immediate results.

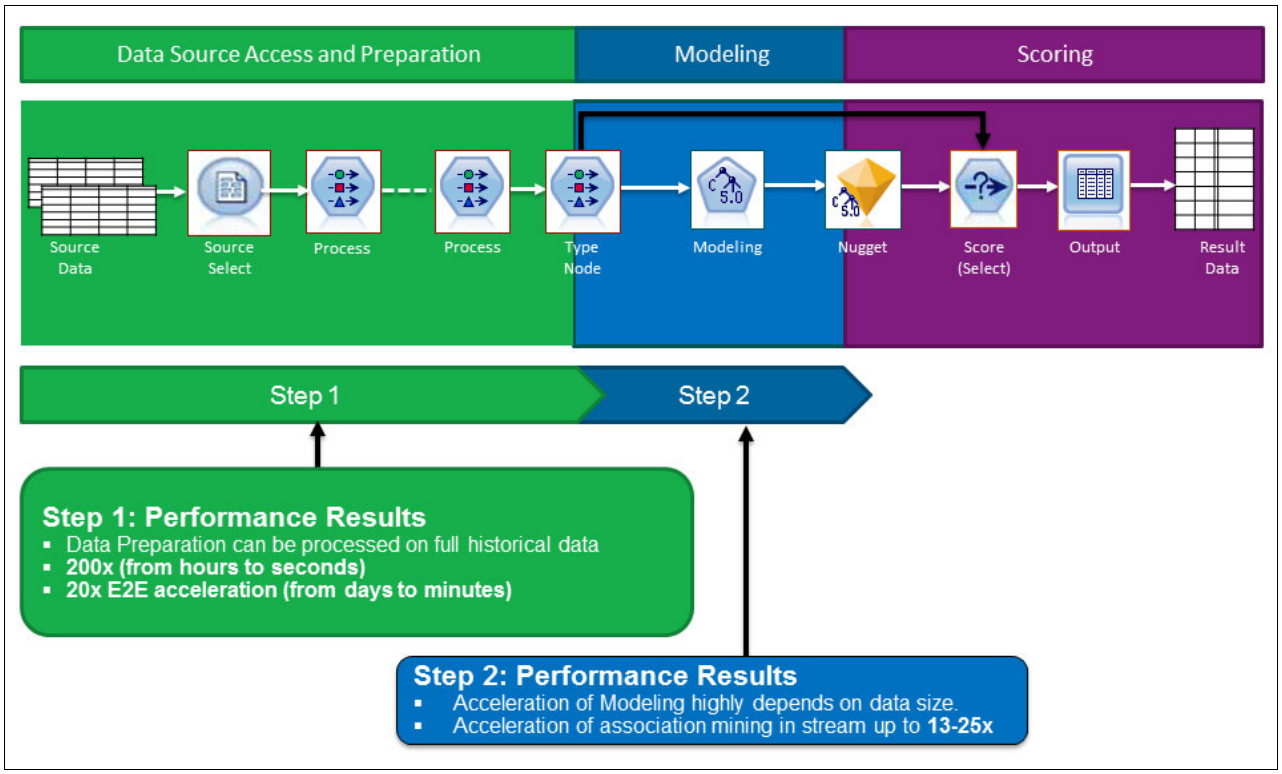


Figure 1-2 Performance improvements by pushing the processing to the accelerator

1.2 Real-time analytics

Innovations in our modern information technology world allow data to be collected, processed, and used at a faster rate than ever before. This speed is fostering the development of more sophisticated analytics that incorporate more data for better decision-making. Such solutions provide faster identification, correlation, and responses to new customer behaviors.

To meet SLAs, avoid penalties, and maintain customer expectations for real-time service, businesses traditionally deploy sub-optimized solutions based on perceived infrastructure constraints. For example, they rely on simpler solutions, such as analyzing a subset of transactions, or post-transactions, and use business rules to make decisions based on aged data. However, these solutions do not closely link transactional systems with database systems so that decisions can be automated, during the transaction, in real-time. Consequently, businesses will miss revenue-generating opportunities, overpay claims, or not uncover criminal activity in a timely fashion.

IBM z Systems platform can score transactions in real time without breaking SLAs and can build richer, more accurate models that are orders of magnitude faster.

1.2.1 Business advantages

The following list highlights several perceived business advantages of real-time analytics:

- Integrate advanced analytics as part of each transaction with negligible impact to transaction SLAs
- Access to most current data for best analytic outcome, and reduced false positives
- Actionable insight on every transaction, real-time or batch
- Analytics in the flow of business to stop fraud, increase customer loyalty, increase revenue and reduce risk

1.2.2 IT advantages

The following list highlights several technical advantages of real-time analytics:

- Avoiding costly extract, transfer, and load (ETL) processes for analytics with fewer copies of data to manage, secure, and make highly available
- Reduced network costs through avoiding off-platform calls during transaction interaction
- z Systems governance for integrating new analytic models
- Leveraging investments in z Systems data infrastructure, particularly data sharing and DB2 Analytics Accelerator
- Extremely efficient scoring in OLTP applications within DB2 for z/OS (using SPSS Scoring Adapter for DB2) or Java (using Zementis), minimizing IT resource consumption

1.3 In-database analytics

In-database analytics refers to processing of analytics algorithms and associated data transformation in the database engine itself. If you perform predictive analytics processing outside your operational database environment, you will be dealing with significant data movement and processing consumption, which can limit your modeling options.

Data preparation, data transformation, and modeling algorithms in any predictive modeling tool, including IBM SPSS Modeler typically involves copying input data from DB2 for z/OS database to the SPSS Modeler server, which in turn performs all data preparation and modeling algorithms. However, the SPSS Modeler server supports integration with data mining and modeling stored procedures (also called as IBM Netezza Analytics stored procedures) available from the DB2 Analytics Accelerator appliance, which would allow you to build, score, and store predictive models within the control of your existing DB2 for z/OS database.

The database engine in the IBM DB2 Analytics Accelerator for z/OS appliance can perform in-database transformation and in-database modeling by keeping the data in its secure place, and at the same time enabling your data scientists to build richer, more accurate models that are orders of magnitude faster using its massively parallel processing (MPP) architecture.

Note: In-database transformation in combination with in-database modeling can result in streams that can be run from start to finish in the database appliance (IBM DB2 Analytics Accelerator for z/OS), resulting in significant performance gains over streams that are run in the SPSS Modeler server.

1.3.1 Accelerated in-database transformation

In-database transformation allows you to combine the analytical capabilities and ease of use of IBM SPSS Modeler with the reliability, availability, security and stability of the DB2 for z/OS environment, while taking advantage of the IBM DB2 Analytics Accelerator for z/OS appliance capability to accelerate SQL execution. The data preparation and data transformation operations will basically generate SQL statements using an SPSS Modeler feature called SQL pushback. The generated SQL is executed and accelerated in the DB2 for z/OS appliance database. The DB2 Analytics Accelerator appliance can perform data sampling, cleansing, filtering, aggregating, joining, and so on by executing the generated SQL statement. DB2 Analytics Accelerator can also store the transformed data temporarily in an accelerated table, which in turn can be used by the modeling algorithms.

1.3.2 Accelerated in-database predictive modeling

In-database predictive modeling take advantage of the IBM Netezza Analytics stored procedure algorithms. The predictive models are entirely built inside the database. The IBM Netezza Analytics stored procedures can efficiently scan the input data, which is already available in the database to build the model. The power and performance of IBM Netezza Analytics stored procedures are realized by fully using the MPP architecture of the DB2 Analytics Accelerator appliance. In this way, you can build your predictive models order of magnitude faster than what is possible by other means and you can frequently refresh your models as the operational data changes.

The predictive models built inside the appliance database can then be published to DB2 for z/OS database. Then the models can be browsed and scored through the SPSS Scoring Adapter interface.

With DB2 Analytics Accelerator, large amounts of data can be ingested and analyzed quickly. This can help with building certain models such as naive Bayes, which does well with large amounts of data.

1.4 Enabling applications with machine learning capability

IBM SPSS Modeler is a powerful analytic tool that supports all phases of data analytics process, including data preparation, model building, deployment, and model maintenance. The solution discussed in this book uses SPSS Modeler to build analytical models, which can be used in statistical analysis, data mining and machine learning. The data scientists can work with the SPSS Modeler client interface to access mainframe data with the same level of ease as from any other platform they are accustomed to.

Machine learning includes supervised learning, using complex mathematical algorithms. Until recently, z Systems platform did not offer an efficient solution in the area of complex mathematical processing. So, in the past, maybe you resorted to offloading operational data (a snapshot from a prior point in time) from z Systems platform to a distributed platform in order to implement machine learning. That way often resulted in obsolete and unreliable results in addition to the unwanted security exposures.

The DB2 Analytics Accelerator for z/OS enables implementing your analytical processing on z Systems platform by accelerating the execution of data transformation and analytical modeling algorithms with the power and performance of IBM Netezza technology. You can use SPSS Scoring Adapter for DB2 for z/OS to perform real-time scoring on your predictive models, which can quickly reveal what is interesting in your data. You can combine your business rules with the predicted information from the analytics models to make real-time

decisions on your DB2 for z/OS data from within your mainframe applications. This approach basically enables your OLTP and batch applications accessing mainframe data with early machine learning capability to learn hidden patterns in your operational data by using clever mathematical models that are readily available with SPSS Modeler. With this approach, you no longer need to offload data from z Systems platform to distributed environments in order to implement machine learning capability, thereby, completely eliminating a potential data breach.

1.5 Value propositions

The integrated solution on z Systems platform that is shown is in Figure 1-1 on page 2 offers the following value propositions:

- Improves productivity of your data scientists. Accelerates turnaround time of predictive analytics lifecycle.
- Unveils hidden information in your warehouse data that cannot be derived with single pass of your data using SQL queries or application reports.
- Enables your applications accessing mainframe data with machine learning capability.
- Eliminates security breach by keeping the hackers away from your sensitive data used in analytics. Reduces IT sprawl for analytics by eliminating data marts in other platforms.
- Allows you to keep and reuse the transformed data that is frequently used in extract, load, and transform (ELT) or extract, transform, and load (ETL) or data preparation completely within the Analytics Accelerator, significantly reducing the cost of loading and unloading on DB2 for z/OS. Reduces complexity, latency, and cost of data transformations for analytics.
- Allows you to perform analytics on historical data while lowering your storage costs on z/OS by using the High Performance Storage Saver feature of DB2 Analytics Accelerator.
- Allows improved access to historical data combined with z/OS transactional data, social media data, and other external data to gain further business insight with the help of IBM DB2 Analytics Accelerator Loader for z/OS tool.

1.6 Related products

Several related product offerings for doing real-time predictive analytics on z Systems platform are as follows:

- IBM SPSS Modeler

  This data-mining workbench is capable to push data transformation down to the database (with accelerator), and do predictive model creation within the accelerator. Learn more by reading Chapter 5, “Data modeling using SPSS and DB2 Analytics Accelerator” on page 107.

- SAS Enterprise Miner

  This vendor data-mining workbench can also create predictive models and more.

- IBM Netezza Analytics

  This set of stored procedures on the accelerator can create predictive models or do statistical or transformational operations on the loaded data, bundled in DB2 Analytics Accelerator 5.1. Learn more by reading 4.4.4, “Netezza based DB2 Analytics Accelerator for z/OS” on page 96.

1.7 Use cases

The real-time analytics solution discussed in this book merges the predictive power of SPSS Modeler with the performance, security, and scalability of the a Systems platform. This combination enables organizations to extract insights from their high-volume transactional applications running on DB2 for z/OS and to use those insights to make smart, proactive decisions. With SPSS Modeler for Linux on z Systems platform, businesses can create models and score new transactional data in real time, reducing the cost and complexity of operational decision-making.

SPSS Modeler can provide broad and deep descriptive and predictive analytics, data preparation and automation, and analytics of structured and unstructured data from virtually any source. With this single solution, you can apply statistical analysis, data mining, real-time scoring, and decision management to human capital management, evidence-based medicine, crime prediction and prevention, supply-chain analysis, and much more.

Explore the following scenarios might stimulate your imagination in arriving at your own solution to meet your business requirements by using technologies discussed in this book.

1.7.1 Countering payment fraud and financial crimes

In this scenario, you find a set of possible sequences of interactions that are typically applied in the banking industry for countering payment fraud, waste, abuse, and financial crimes.

The business goals of this use case are to reduce loss due to card fraud, reduce card deactivation, grow revenue associated with card purchases, reduce call center costs, and improve service yielding preferred card usage. The approach is to incorporate aggregate data from geographic location, merchant, issuer, and card history into the existing card authorization business flow to reduce fraud while preserving transactional SLAs. One feature associated with this use case is that integrated high-performance query optimizations can enable data aggregation several times a day and use this complex data as part of a real-time fraud detection process. A second feature associated with this use case is that enhancements with predictive scoring can be integrated with fraud detection transaction for even more preventive capabilities.

Figure 1-3 shows a business-view diagram for this use case.

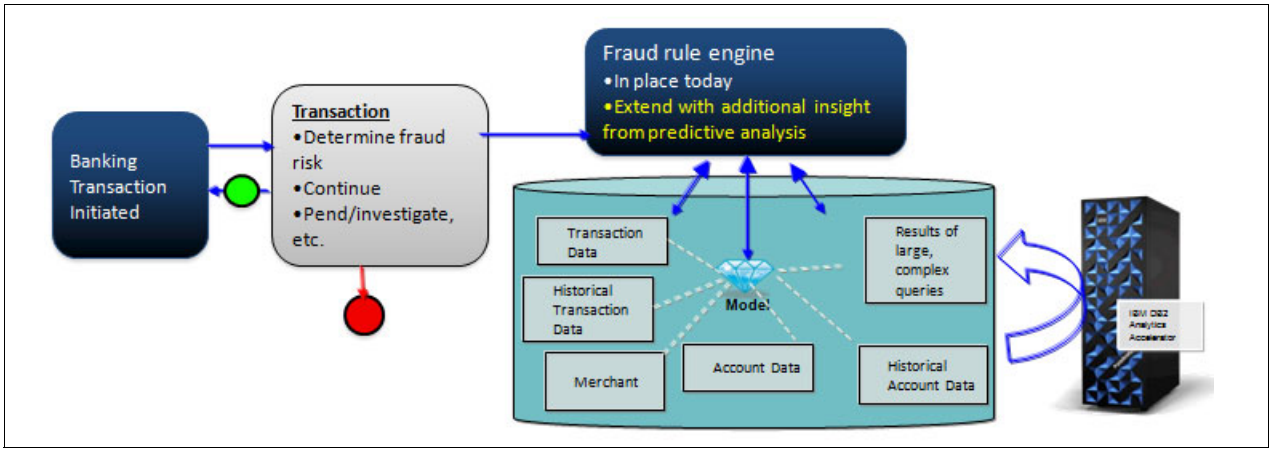


Figure 1-3 Countering payment fraud, waste, abuse, and financial crimes

1.7.2 Insurance claims in-process payment analytics

In this scenario, you find a set of possible sequences of interactions that are typically applied in insurance claims processing systems to perform additional in-process payment analytics.

The business goals for this use case are to quickly and efficiently tag each claim with extra business insight. By extending in-process claims scoring across domains, predictive analytics can be transformed from a specialized function to one that best uses transactional systems at a repeatable enterprise scale.

One challenge this scenario addresses is that complex reports for overpaid claims are not completing on time, resulting in monetary losses. The solution is to integrate optimized analytics of the DB2 Analytics Accelerator with overpayment reporting of transactions. Benefits of this solution include up to 2000x improvement in speed of overpayment reports, line-of-business users are enabled to respond with more agility to overpayment trends, and informed decisions can be made at the right time.

Another challenge this scenario addresses is to stop improper payments prior to payment, avoid “pay and chase,” and meet SLAs. The solution is to integrate predictive analytics into claims adjudication. Benefits of this solution include an efficient scale for analytics, scale requirements are only achievable with analytics as part of the transaction flow, and the expected results of efficient in-transaction analytics can be multimillion US dollars per year. Figure 1-4 shows a business-view diagram for this use case.

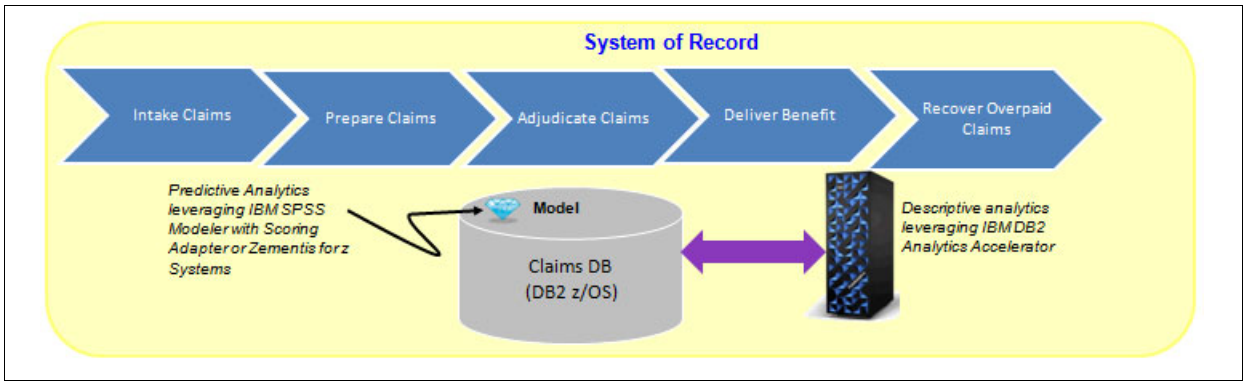


Figure 1-4 Insurance claims processing: in-process payment analytics

1.7.3 Predictive customer intelligence

In this scenario, you find a set of possible sequences of interactions that can be applied to predict customer intelligence in a banking industry for up-selling purposes.

The business goals of this use case are banks that want to integrate information across all product lines in order to make real-time, targeted decisions. Real-time decisions are necessary because banks can risk losing customer business or loyalty for other products. Therefore, there is a need to incorporate high value, predictive advanced analytics as part of transactional systems. For example, all of the data is aggregated to determine a score that can be used to reduce a bank's risk in approving a loan application. Real-time predictive customer intelligence leads to smart business decisions at the point of impact. Benefits of this solution include building long-term customer relationships, driving one decision one interaction at a time, and maximizing customer life-time value. Figure 1-5 shows a business-view diagram for this use case.

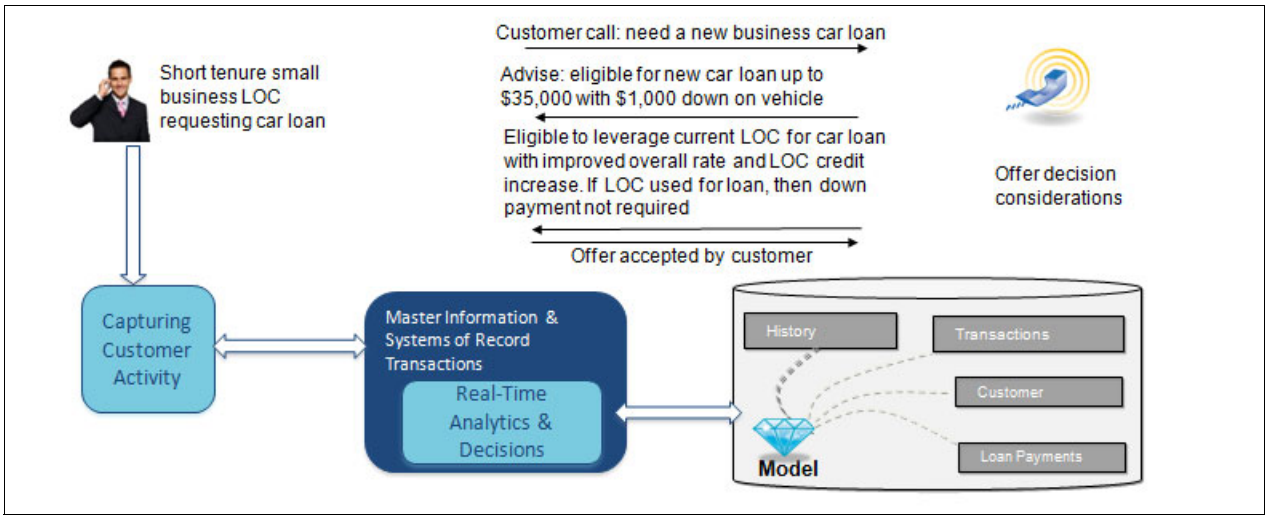


Figure 1-5 Predictive customer intelligence

2

Analytics implementation on z Systems platform

The architecture described in this book combines technologies optimized for online transaction processing (OLTP) with technologies optimized for online analytics processing (OLAP) workloads in one system, combining the strengths of both and allowing the overall system to have leading capabilities for hybrid workloads that are relevant for typical enterprise environments. This type of architecture combines high volume real-time transaction processing environment doing a large number of update operations, following strict service-level agreements (SLAs) with the needs of data scientists analyzing the same data with machine learning algorithms or OLAP reporting tools and potentially deploying new models to the transactional system. The analytical workload can be processed without impacting the transactional workload in a DB2 data sharing environment on z Systems, all without the need to offload the critical data to less secure environments.

This chapter describes the architecture that was used as the basis in writing this book, starting first with an overview of how to add an analytics environment to a preexisting data-sharing environment and continuing with a guide to installation and customization of the environment. The chapter also discusses the phases of analytics lifecycle with reference to the architecture used.

This chapter contains the following topics:

- Adding analytics to a mainframe data sharing environment
- Installation and customization
- Real-time analytics lifecycle

© Copyright IBM Corp. 2016. All rights reserved.

2.1 Adding analytics to a mainframe data sharing environment

The offloading of data-intense workloads, often involving the need for table scan operations on complete databases, to the accelerator with its asymmetric massively parallel processing (AMPP) architecture, enables the delivery of the complete solution at a competitive process. Although the accelerator box does not have the same reliability or quality as the IBM z13™ mainframe, it is used only for workloads that are not business-critical. If a transactional system goes down, the result can be direct revenue loss, significant damage to the company, or both loss and damage; however, if an analytical model cannot be refreshed or a report generation is delayed, the impact is usually less significant. Nevertheless, the data processed is often critical data, data which is simply to valuable to risk that attackers might steal it.

The architectural overview (Figure 2-1) shows the concepts involved in the data sharing function of DB2 for z/OS. With this function, you can completely isolate analytic workloads from business-critical transactional workloads.

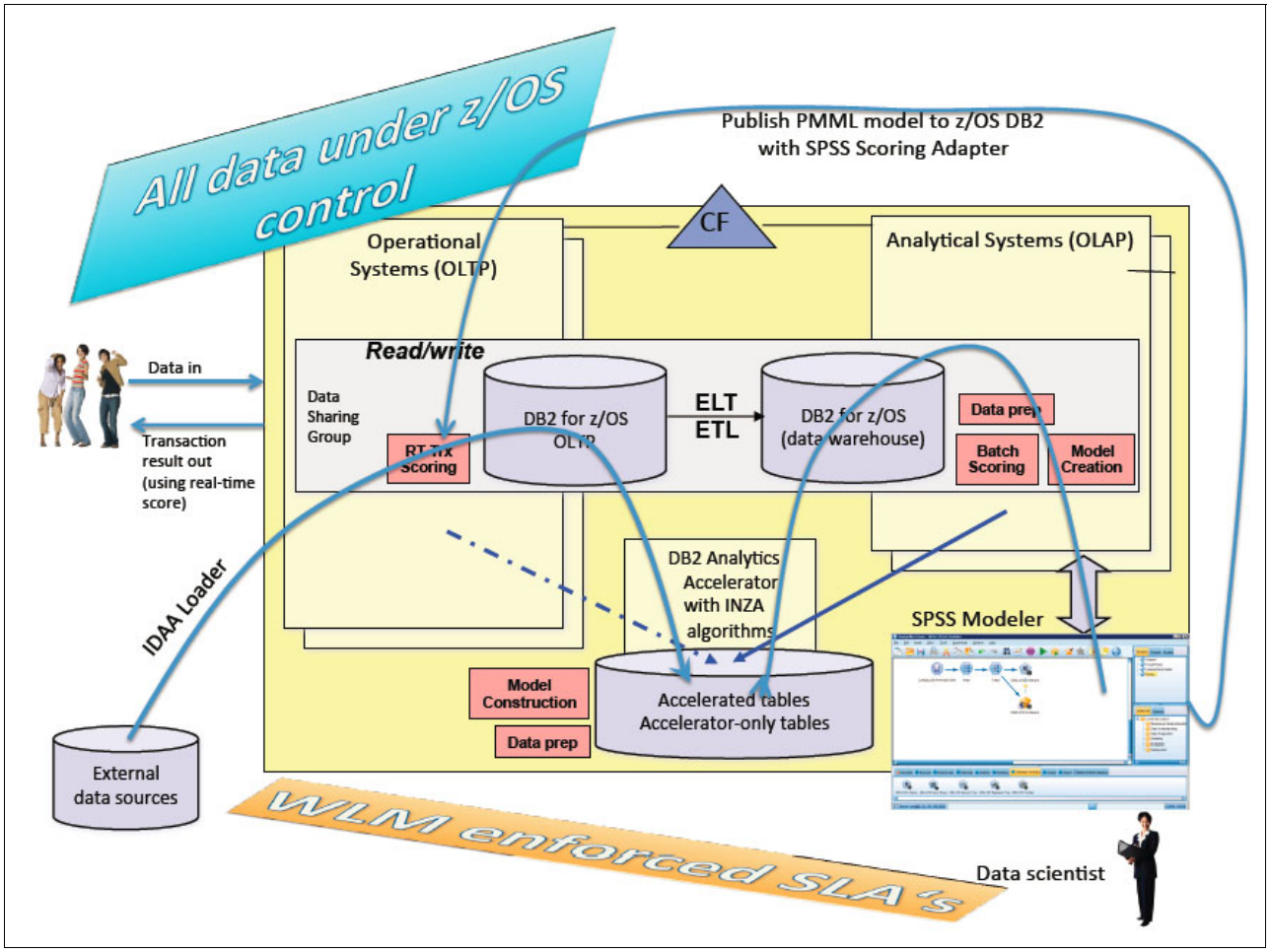


Figure 2-1 Architectural overview

The DB2 Accelerator will not do anything unless it is first routed through DB2 for z/OS with the IBM RACF® component or similar security component. This is also true for the in-accelerator stored procedures (INZA) work on all the accelerator tables (including accelerator-only tables, or AOTs). Before the request is sent from DB2 for z/OS to the accelerator, z/OS validates access privileges on all involved tables, and it creates a sandbox environment on the accelerator, so the accelerator stored procedure (more information is in 2.1.2, “DB2 Analytics Accelerator wrapper stored procedure” on page 16) can see only those tables required for the processing, and only if the invoking user has the correct access rights.

Sample scenario explaining the architecture of the solution

In our sample scenario, the corporation has core operational data on the mainframe, managed with DB2 for z/OS in a data sharing environment. One data sharing member is dedicated to the transactional workload, the second member is used for analytical and for reporting workload. The usual components on the host are in place. The usual components might geoplex and an automation product for high availability (HA) and disaster recovery, Workload Manager (WLM) to enforce service level agreements (SLAs) and to keep resource utilization high without sacrificing performance for users, and Resource Access Control Facility (RACF) to protect resources. Recently, a request was to add social data to the system in order to make use of sentiment data within individual transactions.

By using the DB2 Analytics Accelerator Loader for z/OS, adding the social data to the accelerator can be done easily without the need to materialize it into z/OS DB2 first. In contrast to the data currently stored on z/OS, where the assumption is that it has to be persisted for at least 10 years (due to regulatory requirements), social data is used only to improve business decisions and only valuable while it is current enough, so it should not stay on the system for a longer period of time.

With all the data loaded by the database administrator (DBA) to the accelerator, the data scientist using SPSS Modeler will be able to interactively investigate the data. With the MPP power available in the accelerator, the possibility exists to process many queries or operations for data transformation rapidly, allowing the scientist to work with the tool interactively. After the data scientist basically understands the data, a predictive model is to be created from it. This can be done within the accelerator also, by using stored procedures running there. This all works although no one has direct access to the accelerator appliance, that is, all requests go through z/OS DB2, which keeps z Systems security high although not all processing is done within the CEC.

In the relational OLTP database, data is stored with a normalized schema (such as BCNF or 3NF), avoiding data redundancy and related problems in doing updates in the transactions (“single version of the truth,” which is one of the core benefits of relational databases in general). Modeling algorithms in predictive analytics typically require that all related data is found in a single table, being denormalized. Now with recently added accelerator-only tables, the accelerator is able to transform the input data, using operations, such as join, and to materialize the intermediate results back on the accelerator, in accelerator-only tables. Potentially doing extra data cleansing and general transformation operations on the data, for example in order to remove rows where some values are NULL, those accelerator-only tables do not need to be backed up like “normal” DB2 tables: first, they can be easily re-created based on the base tables; second, they are often needed only once, and can be discarded as soon as the modeling algorithms are done.

SPSS Modeler offers many more modeling algorithms than what's available for in-database mining. Now although those "normal" mining algorithms are not as performant as the in-database counterparts, they are still available and might prove to be useful. In many cases, the streams can be organized in a way so much of the work can be done in-database, accelerated in the accelerator and then, only the (hopefully already to some degree aggregated) result needs to be queried once by SPSS Modeler, which can then apply advanced modeling algorithms. For this scenario, it is best if the SPSS Modeler server is running on Linux on z on the same CEC, with IBM HiperSockets™ connectivity. In case the data being processed is of significant size, a large enough file system for that Linux partition or guest is required. If caching is at the end of the in-database part of the stream, the data is read by SPSS Modeler only once; after it has the data in memory (or at least on own file system), the data scientist can explore the data and create multiple models from it without additional resource requirements against the z/OS DB2 partition.

After the data scientist is confident with the newly created model, the model might be deployed to the SPSS Modeler Scoring Adapter for DB2 z/OS, for in-database scoring. In the deploy step, a sample SQL is generated, which can be given to the customers COBOL programmer for integration into the existing transaction code. In-database scoring has the advantage that it can be run based on real-time data, scaling with DB2 (so z Systems can easily achieve SLAs), something that was problematic at the point in time the company was forced to do the scoring in a distributed scoring engine, where there was network in between DB2 and the distributed server and where overall response time was out of control for z/OS, leading to unpredictable results.

An alternative exploitation of the predictive models is using batch scoring, where all elements of a complete table may be stored at once, with a single operation. This can be useful if the distribution of the score results is to be analyzed. For batch scoring, the tight integration into scalable transaction processing is not required, and the cost for the service invocation is not as critical (as the overall operation takes much longer). For batch scoring, various efficient alternatives exist, including doing this in SPSS Modeler server and doing it using SPSS Modeler Scoring Adapter for DB2 z/OS, or using third-party scoring software like the one offered by Zementis, as SPSS creates the model in standard-based PMML format.

2.1.1 SPSS Modeler

Figure 2-2 shows an architecture overview of Linux on z Systems and IBM HiperSockets with SQL pushback to the database and actually to the accelerator.

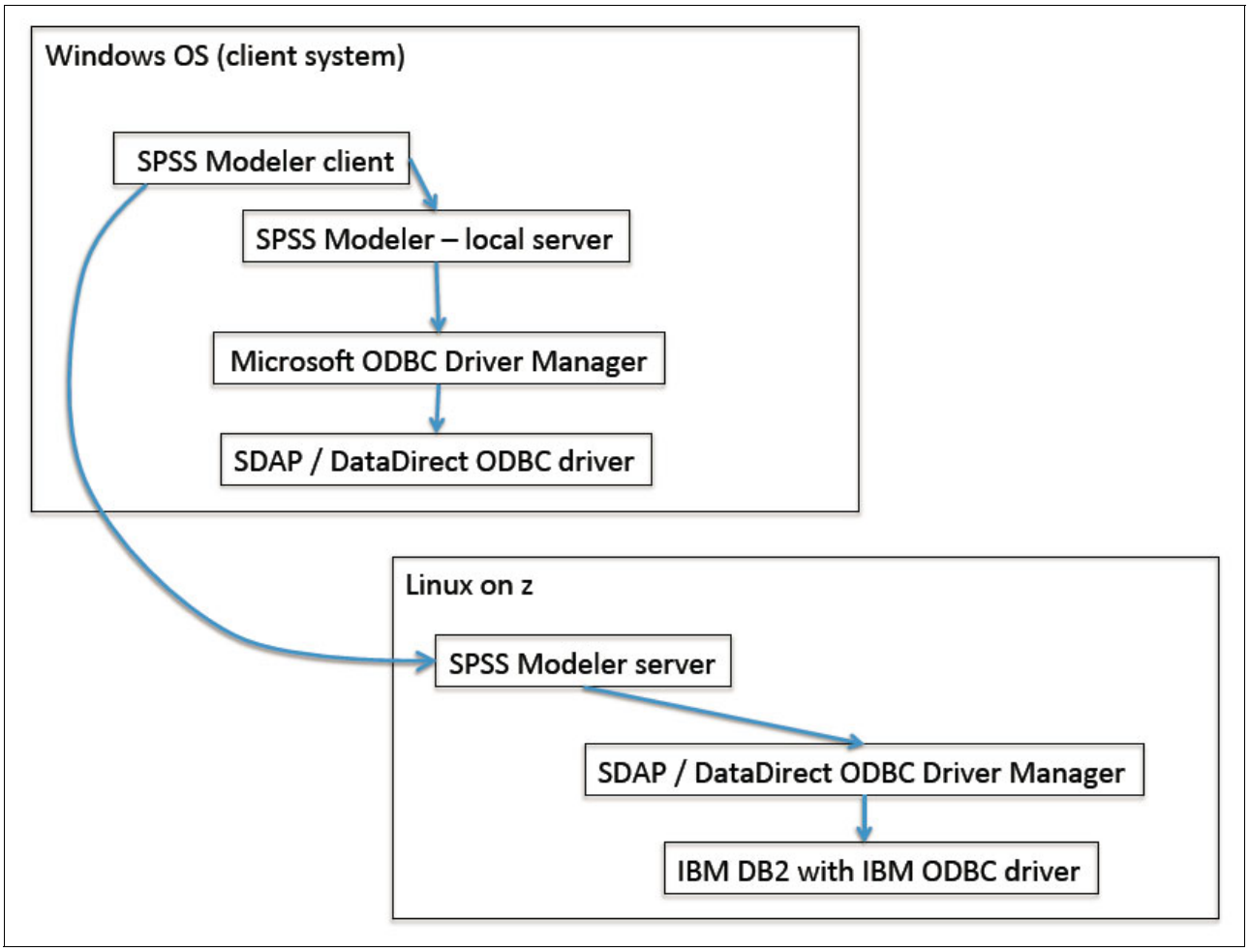


Figure 2-2 SPSS Modeler and on which layer processing can be performed (ModerlDatabase.png)

SQL generation

SPSS Modeler can generate SQL out of the stream, also known as *SQL pushback*. With this feature, SQL can be generated that can be "pushed back" to (or executed in) the source database. In general, this is efficient because eliminates data movement between the database and the SPSS Modeler server. If DB2 Analytics Accelerator comes into play and the tables used are huge (so DB2 Optimizer decides to do query evaluation in the Netezza based accelerator), the advantage is even more significant because the accelerator can benefit from its AMPP architecture, especially if the operation requires table scans (and this is often the case for streams developed using SPSS for analytics; a DB2 index seldomly helps here).

Using SQL generation in combination with in-database modeling (see 1.3, "In-database analytics" on page 5) results in streams that can be processed in-database from start to finish.

Partial stream execution in-database

To evaluate an SPSS stream, the variation that always works is that SPSS Modeler retrieves all data sources into its own memory (or to temporary files on the server if the memory available is not large enough) and then perform the stream operations on its own. Those operations include joins on various data sources or tables, aggregation operations, simple transformations, and so on.

Although SPSS does its best to generate SQL, generating SQL out of a stream is not always possible. An extreme example might be a join operation between two data sources (different DB management systems), where database federation technologies are necessary to come up with a working SQL solution (which does not exist), or the combination of Microsoft Excel data with a z/OS DB2 table, or even some expressions found in SPSS CLEM expression, a SPSS specific language with higher expressive power than standard SQL. In such cases, SPSS Modeler retrieves the required data from the database and performs the operations on its own. SPSS Modeler tries to re-order the stream, so the nodes that can be pushed down to the database are done first (of course, only within limits and without changing semantics).

2.1.2 DB2 Analytics Accelerator wrapper stored procedure

Accelerated in-database modeling is implemented using predictive modeling stored procedures. For instance, the `INZA.DECTREE` stored procedure creates a decision tree model from an input table plus output parameter indicating which attribute is being predicted. As you might know, a client application cannot connect directly to the accelerator. It can connect only to DB2 for z/OS; the accelerator itself remains completely isolated and out of reach of any client application. So for this to work, the `INZA.DECTREE` stored procedure is a DB2 for z/OS stored procedure.

It is called a *wrapper stored procedure* because the tasks are restricted to security-related processing, table name mapping, and other minor administrative tasks. After the stored procedure on z/OS checks which target accelerator, which DB2 z/OS tables (which need to be accelerated) are involved, whether the calling user ID has READ privileges to those tables and so on, it lets the accelerator create a controlled environment, a kind of a sandbox in which only the tables required for the stored procedure execution are accessible.

The next step is then to invoke the corresponding stored procedure on the accelerator, which has the same procedure name and which is an IBM Netezza Analytics stored procedure (see “IBM Netezza Analytics stored procedures” on page 190). More information about the stored procedures is in the following documents:

- *IBM Netezza Analytics Release 3.2.0.0, In-Database Analytics Developer’s Guide* (November 20, 2014)
- *IBM Netezza Analytics Release 3.0.1, IBM Netezza In-Database Analytics Reference Guide*

DB2 wrapper stored procedures parameter structure

The wrapper stored procedures are implemented generically. Every IBM Netezza Analytics wrapper stored procedure has the following parameters:

- ACCELERATOR_NAME: This is the name of the accelerator to be used.
- PARTAMETERS: This parameter string is passed to the IBM Netezza Analytics stored procedures within the accelerator after some processing to map the table names of input and output tables. For accelerator-born tables, which are first created on the accelerator, the DB2 Analytics Accelerator code remembers that in order to be able to create the metadata in z/OS DB2 catalog and to establish mapping information in case the IBM Netezza Analytics SP on the accelerator successfully finished. The output tables are

created in the database, which is specified in the AQT_ANALYTICS_DATABASE environment variable in AQTENV. All output tables are accelerator-only tables, so the data is not moved back to DB2 z/OS. If you need the data values available in DB2 z/OS base tables, an INSERT FROM SELECT SQL expression can be used, inserting into a DB2 base table and selecting from the accelerator-born output table.

- RETURN_VALUE: Although this is not SQL standard-compliant, some IBM PureData® for Analytics (Netezza) stored procedures return a value to the caller. Because DB2 z/OS does not allow stored procedures to return a value such as a user-defined function in SQL, this parameter is mapped to an output parameter in DB2. Possible types are SMALLINT, INTEGER, BIGINT, DOUBLE, REAL, VARCHAR and CLOB. This parameter is optional.
- MESSAGE: This parameter is well-known to those familiar with DB2 Analytics Accelerator, and both the input XML and the output XML are identical to what is also used by the existing administration stored procedures in DB2 Analytics Accelerator.

Result sets

Some IBM Netezza Analytics stored procedures return text output in addition to the RESULT return value. This is returned as part of the first result set of the wrapper stored procedure.

If trace is enabled in the MESSAGE parameter, the trace is returned in the next result set of the wrapper stored procedure.

If the length of the text exceeds 32698, it is split into multiple result rows. For DB2 for z/OS, this means that the information becomes part of a declared global temporary table (DGTT) that is created by the SQL statement, shown in Example 2-1.

Example 2-1 DGTT used by IBM Netezza Analytics stored procedure to return multiple rows

```
CREATE GLOBAL TEMPORARY TABLE
DSNAQT.ACCEL_TABLES_DETAILS
( SEQID INTEGER, TABLES_DETAILS VARCHAR(32698))
CCSID UNICODE
```

The concatenation of the result set VARCHAR column in ascending order of SEQNO provides the original text output of the IBM Netezza Analytics procedure.

Authorizations

The user ID under which this stored procedure is invoked must have the following privileges:

- EXECUTE on the wrapper procedure
- EXECUTE on the SYSACCEL.* packages
- Privilege to read the accelerated tables referenced in the PARAMETERS parameter as an input table
- Write access to `/tmp` directory for the user calling the stored procedure (for tracing)
- DISPLAY privilege (for calling `SYSPROC.ADMIN_COMMAND_DB2(-DIS GROUP)`)
- Authorization to create an accelerated table (specified as output table) in DB2 (in the database specified by AQT_ANALYTICS_DATABASE if this environment variable is set in AQTENV)

Prerequisites for calling a wrapper stored procedure are as follows:

- Accelerator is started.
- All input tables are enabled for query acceleration.
- Output tables do not exist in DB2.
- Authorizations are in place.
- ZPARM OPTHINT must be set to YES.

Calling INZA.KMEANS stored procedure from DB2 for z/OS

Typical invocation of the wrapper stored procedure from DB2 for z/OS is like the one shown in Example 2-2.

Example 2-2 Sample call to INZA.KMEANS stored procedure from DB2 for z/OS

```
CALL INZA.KMEANS(
'ACCEL01',
'model=customer_mdl, intable=TPCH30M_U.CUSTOMER, outtable=customer_out, id=C_CUSTKEY,
target=C_NATIONKEY, transform=S,distance=euclidean, k=3, maxiter=5,
randseed=12345,idbased=false,
'<?xml version= "1.0" encoding= "UTF-8" ?><aqttables:messageControl
xmlns:aqttables = "http://www.ibm.com/xmlns/prod/dwa/2011"   version = "1.2">
   <traceConfig keepTrace="false" forceFlush="true"
location="/tmp/INZAKMeansTableSim16.trc">
      <component name = "PROCEDURE" level = "DEBUG" />
   </traceConfig>
</aqttables:messageControl>
```

If the execution of the `CALL` shown in Example 2-2 is successful on the accelerator and a KMEANS clustering model is created, the result is like Example 2-3 (the requested debug trace was intentionally removed from this sample output).

Example 2-3 Sample XML output from a call to INZA.KMEANS stored procedure

```
RETVAL  1
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="AQT10000I"><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>
```

2.1.3 Using accelerator-only tables

After an accelerator is installed and activated, the DB2 optimizer treats the appliance as a new access path and will offload SQL processing transparently when doing so is more efficient. Not every query will be accelerated. Some might have incompatible SQL statements, or the DB2 optimizer might decide that offloading this processing is not the best choice.[1]

An accelerator-only table is automatically created in the accelerator when the CREATE TABLE statement is issued on DB2 with the IN ACCELERATOR clause. Any queries that reference an accelerator-only table must be executed in the accelerator and will be accelerated. If the query is not eligible for query acceleration, an error is returned.

Running queries and data change statements on the accelerator can significantly increase the speed of SQL statements, such as INSERT from SELECT statements. Accelerator-only tables are used by statistics and analytics tools, which can quickly gather all the data that is

[1] For more information, see *Optimizing DB2 Queries with IBM DB2 Analytics Accelerator for z/OS*, SG24-8005.

required for reports. Because the data in these tables can be modified so quickly, they are also ideal for data-preparation tasks that must be completed before the data can be used for predictive modeling.

For more information about accelerator-only tables, see 4.1.1, “Accelerator-only table (AOT)” on page 85.

2.2 Installation and customization

For this publication, the authors set up a demo with z/OS DB2 subsystems in a data sharing group being paired with DB2 Analytics Accelerator.

One DB2 subsystem contained tables in EBCDIC, the other tables in UNICODE, purposely not mixing EBCDIC and UNICODE in one subsystem.

SPSS Modeler server was running on a Linux on z LPAR on the same z13 machine, connected to z/OS DB2 through IBM HiperSockets.

The overall architecture of the systems and components installed and used is summarized in Figure 2-3.

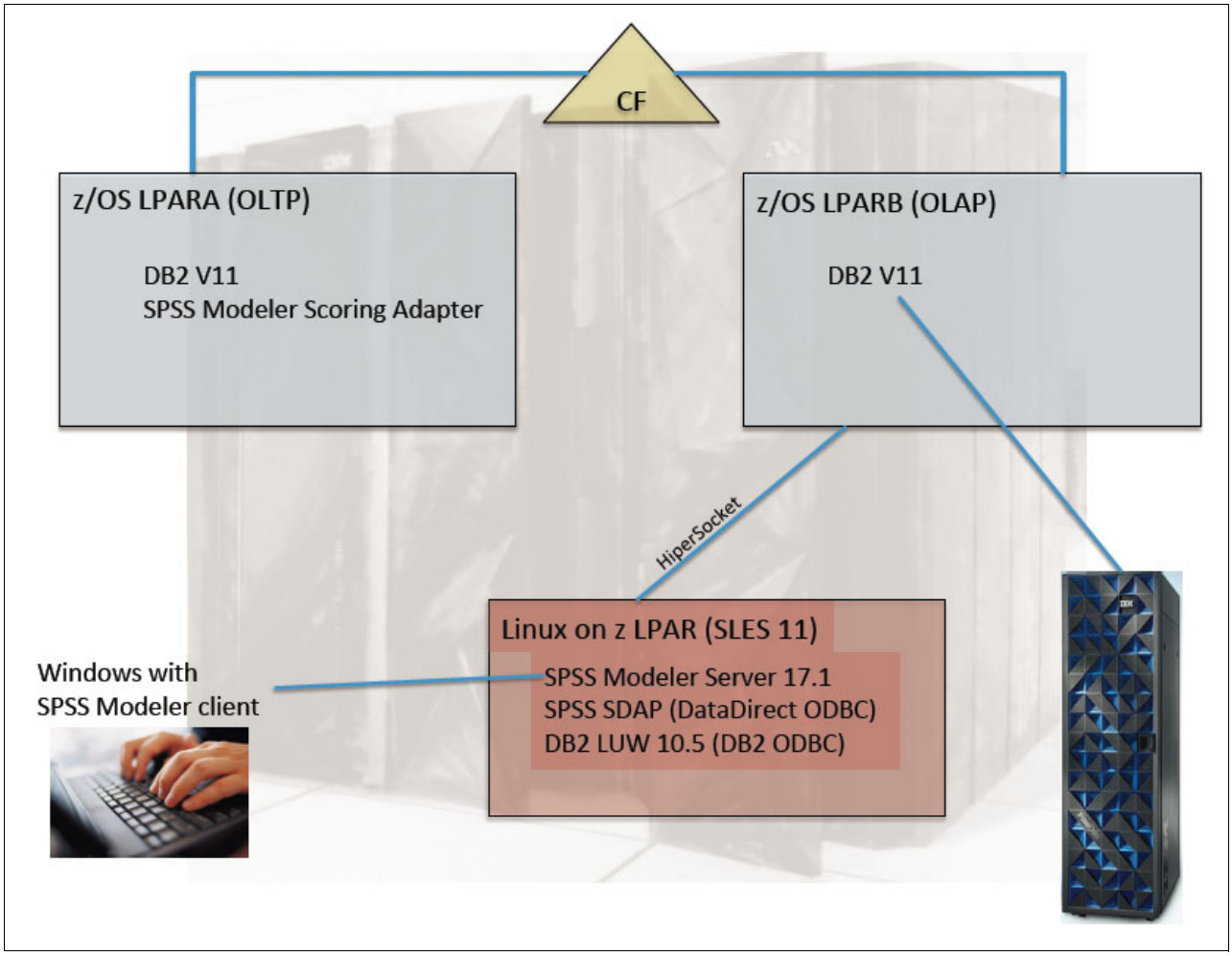


Figure 2-3 Architecture used

2.2.1 DB2 and DB2 Analytics Accelerator setup and installation

On DB2 V11, we ensured that all PTFs that are required for DB2 Analytics Accelerator 4.1 PTF6 are installed. At the time of this writing, no additional DB2 requirements were required for DB2 Analytics Accelerator 5.1.

ZPARM OPTHINT must be set to YES.

Next, we installed the DB2 Analytics Accelerator 5.1 development versions (available at the time of this writing) in the usual way, including IBM DB2 Data Studio 4.1.2 DB2 Analytics Accelerator 5.1 plug-ins, Netezza Performance Server® (NPS®) level 7.2.1.0 (build 46322, 20151102-2301), Access Server and Replication Engine for CDC replication, z/OS DB2 Stored Procedures, and the Accelerator server itself.

Note: If you are familiar with the installation of previous versions of DB2 Analytics Accelerator, SAQTOSR is no longer required.

Installation of optional server packages

IBM Netezza Analytics is the first DB2 Analytics Accelerator extension package installed in a new way. Basically, the installation process includes these steps:

1. Transfer a software package signed by IBM from z/OS hierarchical file system (HFS) mount point to the Accelerator. For integrity reasons, only packages correctly signed by IBM are applicable for transfer to the Accelerator appliance. This is to avoid uncontrolled software components being enabled for execution on the Accelerator.
2. Apply that new package so the software gets installed.
3. Enable or disable the package for a given database on the accelerator. Note that this only works after the first table is accelerated. At the time IBM Netezza Analytics is enabled, additional internal management data structures to hold information on predictive models are initialized.

Enabling the new IBM Netezza Analytics package on the Accelerator

The HFS used for transfer is, as in previous releases of DB2 Analytics Accelerator, the one specified in AQT_HOST_PACKAGE_DIRECTORY in AQTENV on z/OS. After copying the IBM Netezza Analytics package file (in this example, `dwa-inza-3.2.1-20151101-2301.tar`) there and clicking on **Transfer updates** in Data Studio (under **About** → **Server**), the Transfer dialog opens (Figure 2-4).

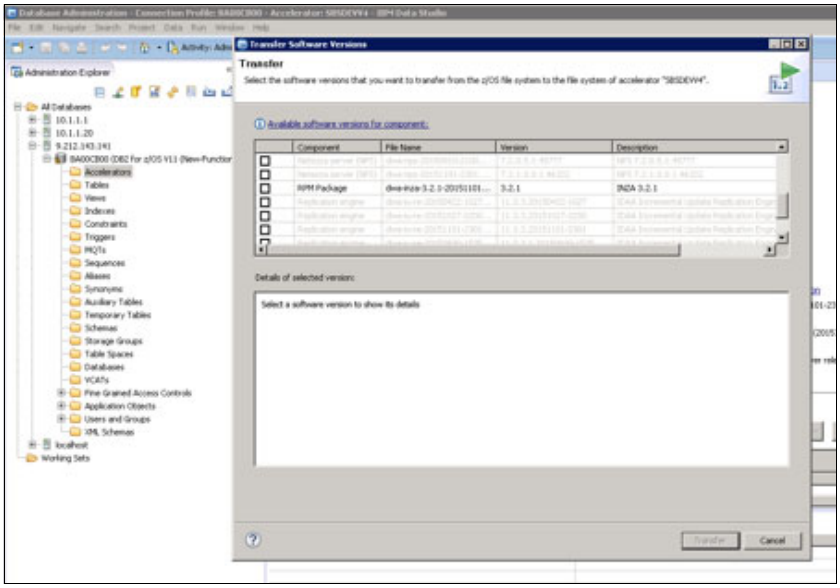


Figure 2-4 Enabling IBM Netezza Analytics package on Accelerator: Transfer Update process

Complete these steps:

1. Select the IBM Netezza Analytics **RPM** package and click **Transfer**. After the RPM file is transferred to the Accelerator, a success message is displayed (Figure 2-5).

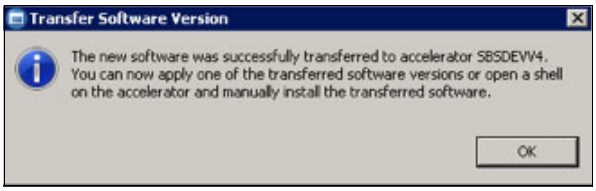


Figure 2-5 Enabling package on Accelerator: Transfer Update completion confirmation

2. Now that the new IBM Netezza Analytics package is successfully transferred to the Accelerator, you can apply it by using **Apply other software version** in Data Studio. Figure 2-6 on page 22 shows the component selection window and **RPM Package** selected.

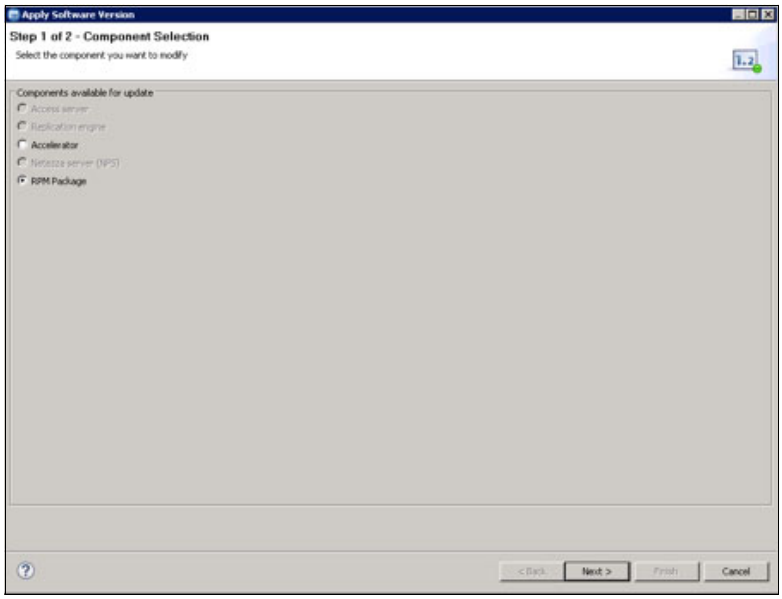


Figure 2-6 Enabling package on Accelerator: Apply Updates - RPM package selection

3. You can select one of the packages already transferred but not yet applied on the Accelerator. Figure 2-7 shows the new IBM Netezza Analytics package to be applied.

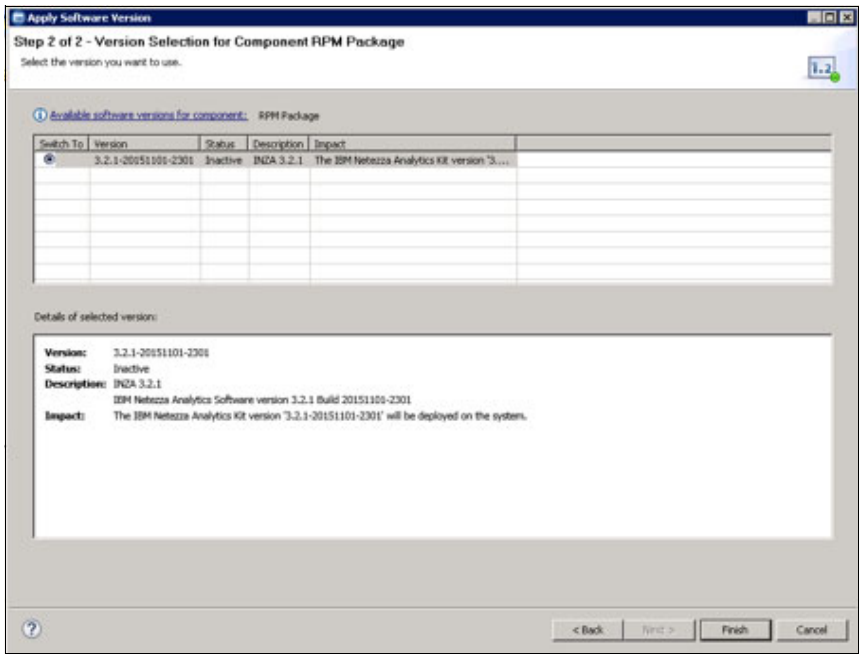


Figure 2-7 Enabling IBM Netezza Analytics package on Accelerator - version selection

To understand more about this software, the version and build numbers are those of the actually used IBM Netezza Analytics packages, also available on distributed platforms on IBM PureData System for Analytics. Software installation and system management is of course completely different because the packages are not directly reachable in the DB2 Analytics Accelerator appliance; they are more encapsulated by z/OS DB2 with DB2 Analytics Accelerator, however the semantics of the available IBM Netezza Analytics stored procedures with capabilities like predictive analytics modeling algorithms are identical.

4. Confirm that you want to activate IBM Netezza Analytics on the accelerator by clicking **OK** in the confirmation window (Figure 2-8).

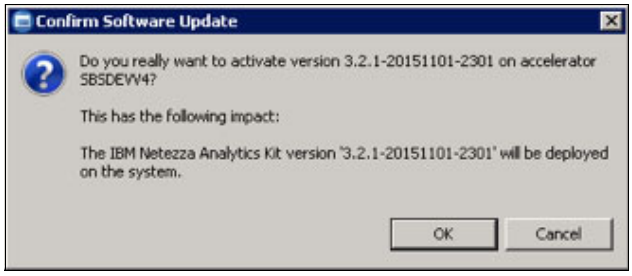


Figure 2-8 Enabling package on Accelerator: Transfer - Update completion confirmation

The system starts applying the software (Figure 2-9).

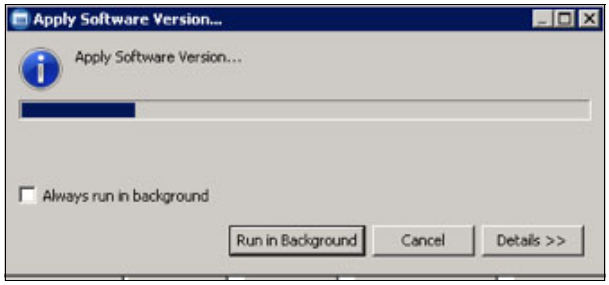


Figure 2-9 Enabling package on Accelerator: Transfer - Apply progress

The IBM Netezza Analytics software library is installed on the Accelerator, available for all subsystems loaded. Now, the About section in Data Studio GUI looks similar to Figure 2-10.

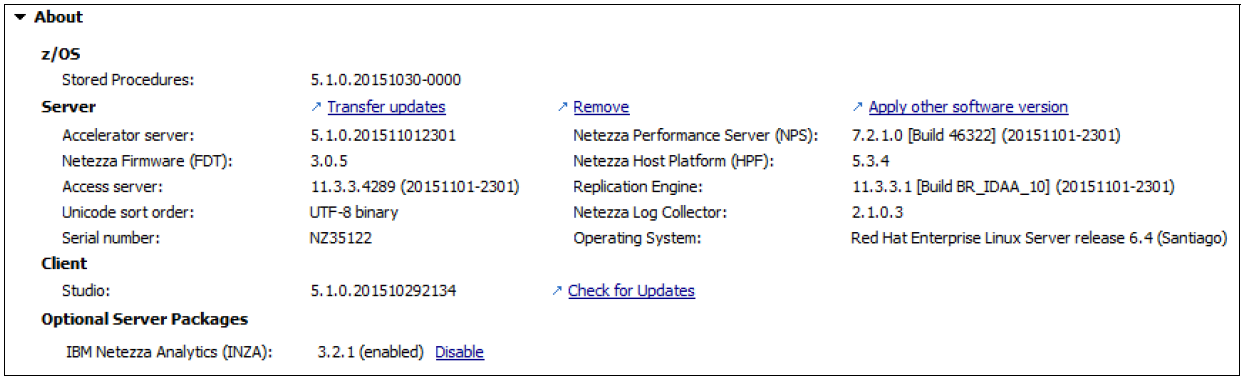


Figure 2-10 Data Studio GUI showing version details about different components of the Accelerator

For every subsystem, IBM Netezza Analytics functionality must be enabled, which works only after the first table is loaded. It has the effect that some internal data structures for model management and other similar items are initialized. Without enabling IBM Netezza Analytics, using the modeling stored procedures will lead to an error message.

Considerations for enabling and disabling IBM Netezza Analytics

After disabling and re-enabling the optional server package IBM Netezza Analytics (INZA) for a given database on the Accelerator, wait approximately 15 minutes so that you do not interfere with the Accelerator's internal maintenance daemon, which might still be cleaning up on objects in the Accelerator database.

Registering INZA wrapper stored procedures

The wrapper stored procedures are to be registered using DDL SQL as shown in Example 2-4.

Example 2-4 Registering INZA.DECTREE Stored Procedure

```
CREATE PROCEDURE "INZA"."DECTREE" (
      IN ACCELERATOR_NAME VARCHAR(128) CCSID UNICODE FOR MIXED DATA,
      IN PARAMETER CLOB(65536) CCSID UNICODE FOR MIXED DATA,
      OUT RETVAL INTEGER,
      INOUT MESSAGE CLOB(65536) CCSID UNICODE FOR MIXED DATA)
   DYNAMIC RESULT SETS 2
   EXTERNAL NAME "AQT02ISP"
   LANGUAGE C
   MODIFIES SQL DATA
   PARAMETER STYLE DB2SQL
   FENCED
   WLM ENVIRONMENT !WLMENV!
   ASUTIME NO LIMIT
   COLLID SYSACCEL
    RUN OPTIONS
    'TRAP(ON),HEAP(32K,32K,ANY,KEEP,8K,4K),POSIX(ON),
    STACK(128K,128K,ANY,KEEP,512K,128K),BELOW(4K,4K,FREE),
    XPLINK(ON),ALL31(ON),ENVAR("_CEE_ENVFILE_S=DD:AQTENV")'
   STAY RESIDENT NO
   COMMIT ON RETURN NO
   PROGRAM TYPE MAIN
   SECURITY USER
   PARAMETER CCSID UNICODE;
GRANT EXECUTE ON PROCEDURE "INZA"."DECTREE" TO PUBLIC;
```

This wraps the Accelerator stored procedure `nza.dectree()` to the DB2 Analytics Accelerator stored procedure `INZA.DECTREE`. The first parameter is always the z/OS DB2 DB2 Analytics Accelerator name, the second is being mapped to the base stored procedure (after checking security and doing table name transformations). For more details, see 2.1.2, “DB2 Analytics Accelerator wrapper stored procedure” on page 16. The job to register the wrapper-stored procedures used by SPSS Modeler is in “IBM Netezza Analytics stored procedures” on page 190.

2.2.2 Required DB2 privileges for SPSS users

If you use SPSS Modeler with in-database modeling, or more precise, you use wrapped INZA stored procedures, you must have the following privileges defined in DB2:

- EXECUTE on the wrapper SPs
- EXECUTE on the package SYSACCEL
- Privileges to read the accelerated tables that are to be used
- Authorization to create and to drop tables in the AOTDB (see 2.2.6, “AQT_ANALYTICS_DATABASE variable” on page 33)

2.2.3 SPSS Modeler client

First, install SPSS Modeler client on Windows the usual way. Currently, the client is available only on Windows; running the graphical interface of SPSS Modeler without a Windows client is not possible (this client might be virtual or on a server using Microsoft Remote Desktop if running Apple OS X on a Mac, or Linux).

Enable DB2 Analytics Accelerator

Now, enable DB2 Analytics Accelerator support for SPSS Modeler.

Until the switch is removed from the product, you must first find the configuration file and edit it accordingly:

1. Open Windows file explorer and open your user's home directory in Windows, which is typically under `C:\Users`.

   For instance, Figure 2-11 shows one way to navigate to Users home directory.

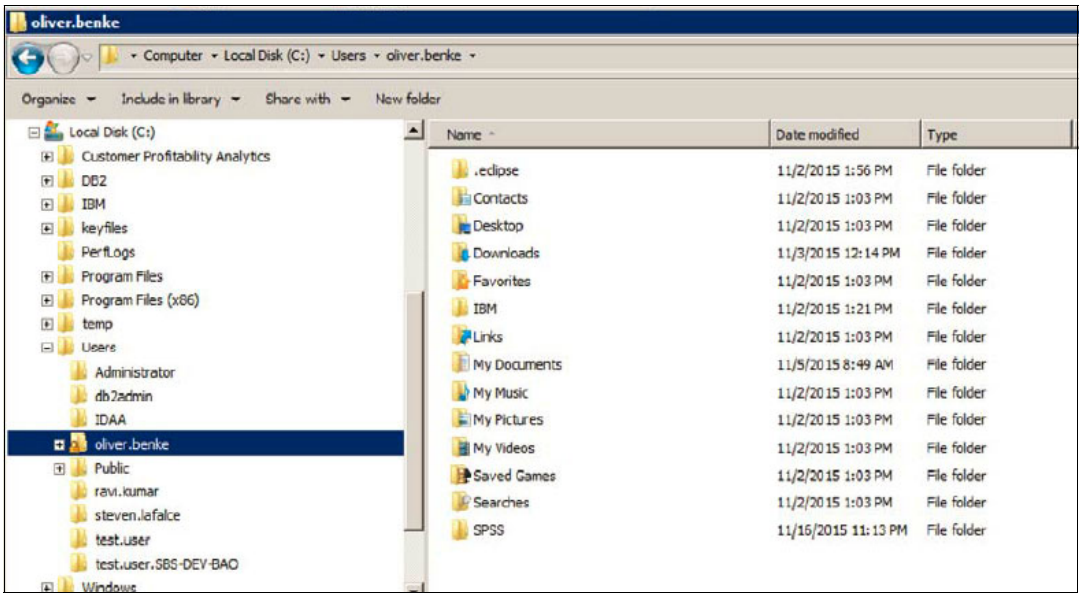


Figure 2-11 Navigating to Users home directory

2. Select the field containing the path in standard format. The path is similar to Figure 2-12.

Figure 2-12 Path name in standard format for Users home directory

3. Place the cursor beside your userid. The Windows explorer path name area changes to the traditional MS-DOS style version of the file path (Figure 2-13).

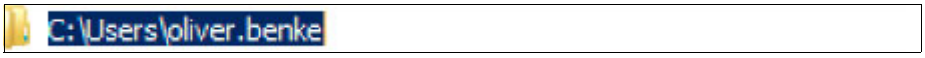


Figure 2-13 Path name in traditional format for Users home directory

4. You can now add the "hidden" directory `AppData`. You can now see the contained directories. Go to the following folder:

   `C:\Users\oliver.benke\AppData\Roaming\IBM\SPSS\Modeler\17.1\Defaults`

If you installed the user preferences in SPSS Modeler client, you see a `user.prefs` file here. If not, create one.

In any case, make sure it contains a line `EnableIDAA=true` (update the line to `true` if it says `EnableIDAA=false`). A sample files is shown in Figure 2-14.

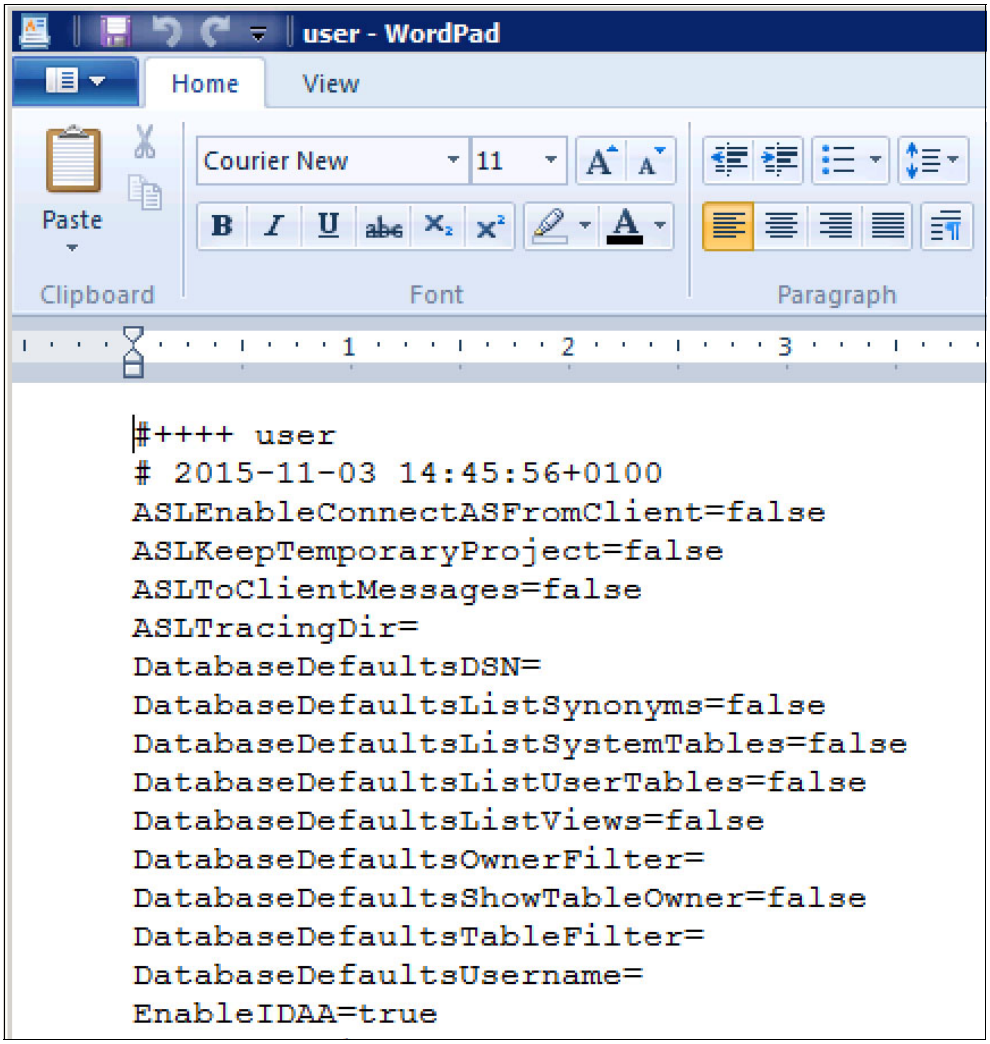


Figure 2-14 SPSS Modeler client user preference file

Client customization

From the Tools pull-down menu of SPSS Modeler client, open the **Options** section, and then select **Helper Applications**. The Helper Applications window opens (Figure 2-15).

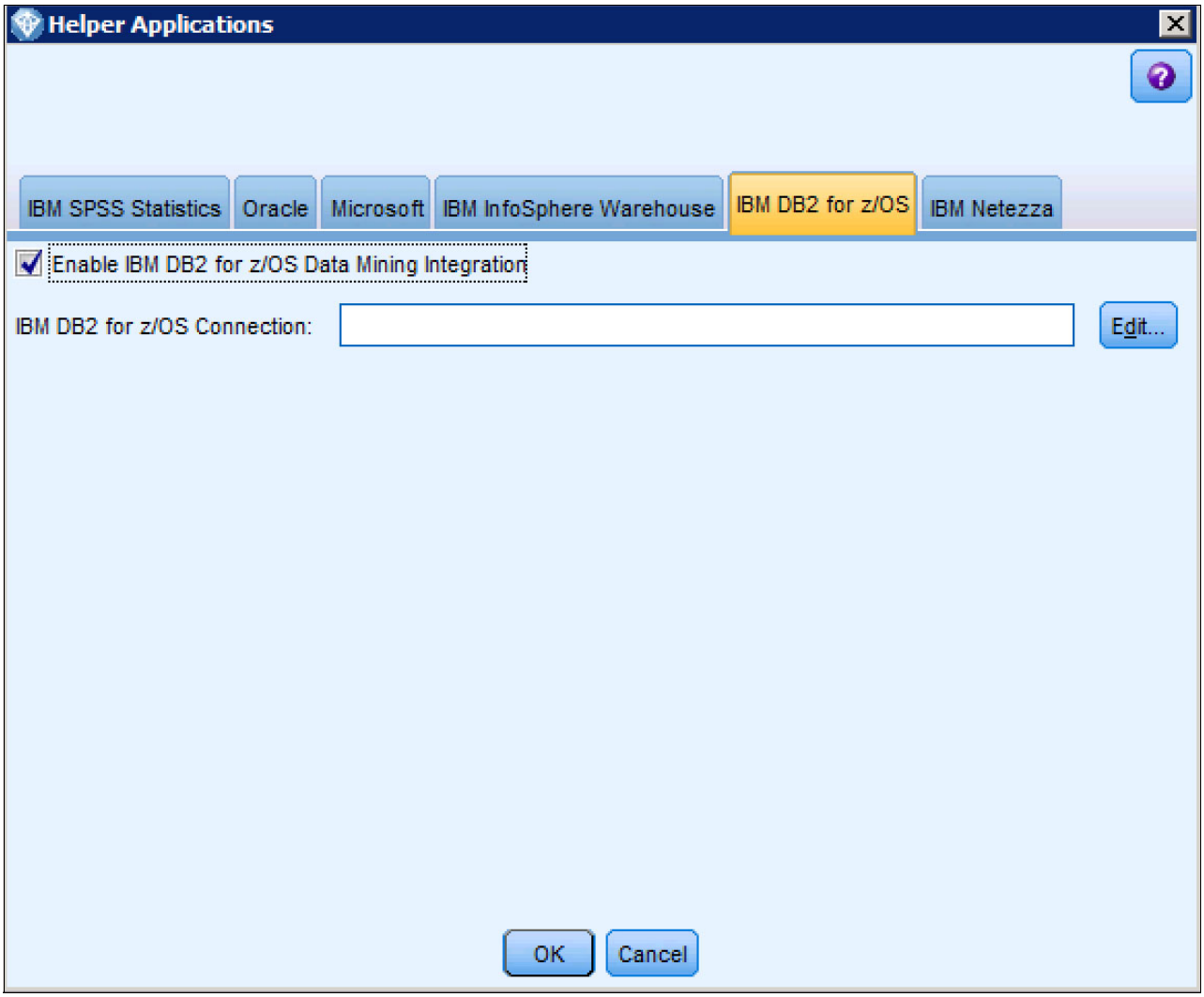


Figure 2-15 Helper Applications on SPSS Modeler Client

Select the **Enable IBM DB2 for z/OS Data Mining Integration** check box, which is not enabled, by default.

In the IBM DB2 for z/OS Connection field, you can define a DB2 connection with the associated Accelerator. It is used by SPSS Modeler in case data is processed in the SPSS Modeler but must return to the Accelerator. So SPSS Modeler uses this connection then to push intermediate data down to the Accelerator, using intermediately created accelerator-only tables. For now, you can leave this field blank because it is typically not required and this is to support a scenario that does not perform well due to data transfer between DB2, Accelerator, and SPSS Modeler.

If the stream you created requires pushback from SPSS Modeler to DB2 (which in this case basically means, to the accelerator) but you did leave this field empty, an error message occurs (as indicated by the red circle with the white letter “X” shown in Figure 2-16 on page 28) when you try to run your stream.

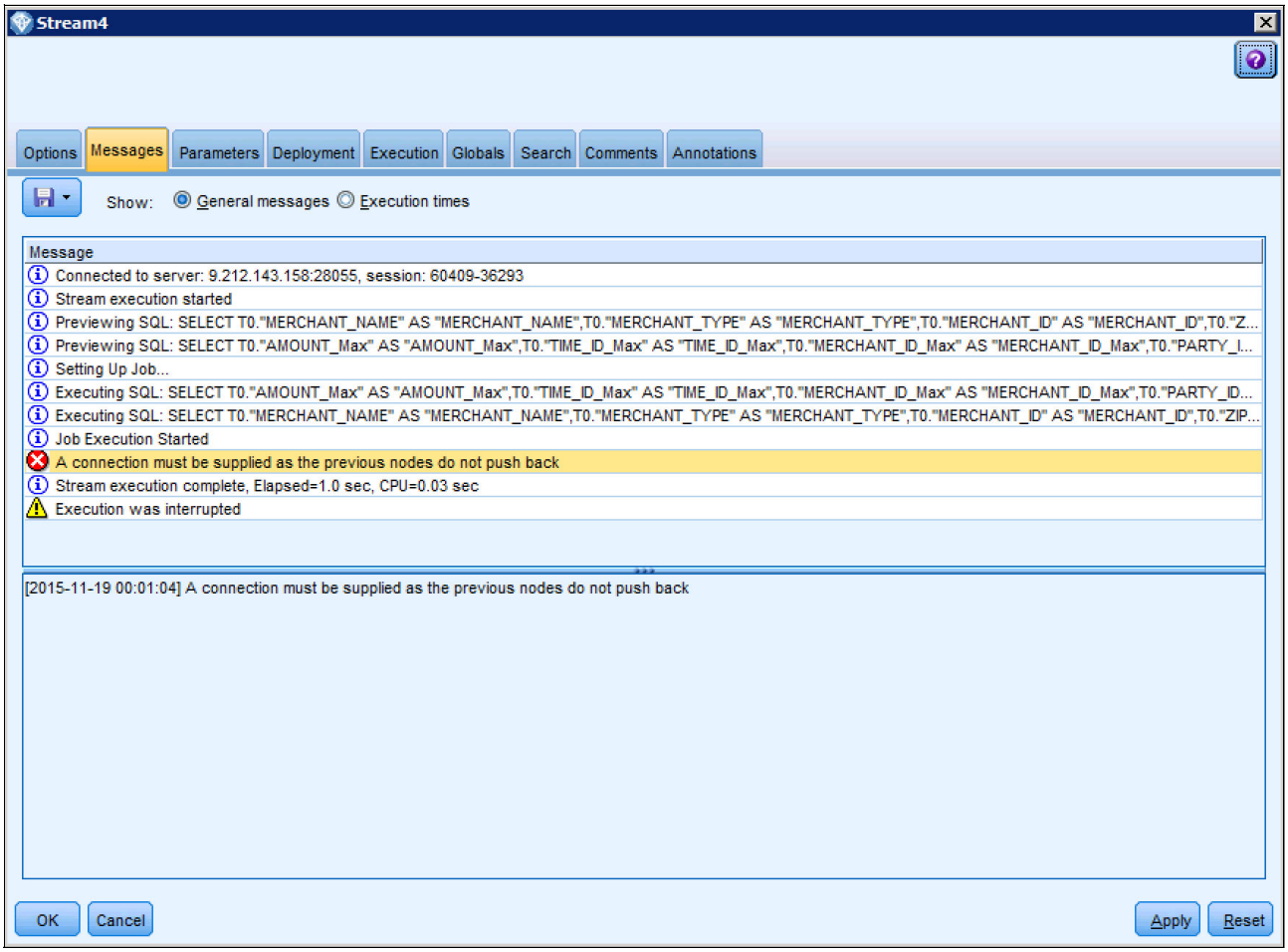


Figure 2-16 Sample error message when DB2 for z/OS connection definition is missing

Although you can add something here so that it always works, be aware that with currently available technology, the pushback might be so resource hungry for your amount of data to process that you might prefer changing the stream slightly rather than paying this resource consumption. *Pushback* means that all the data produced in SPSS Modeler must be loaded back to DB2 with Accelerator. And in many cases, slight semantical difference exist that force SPSS Modeler to do calculations within the SPSS Modeler server. For example, to trigger this error, we created a stream with a merge node where the merge operation was not a standard join expressible in SQL but Modeler default, **"**`Merge Method: Order`**"**. Probably most people would have preferred an SQL inner join here, if looking at the settings.

After you enable IBM DB2 for z/OS data mining integration, a new Database Modeling tab (palette) becomes available. It shows z/OS DB2 in-database modeling nodes (Figure 2-17).

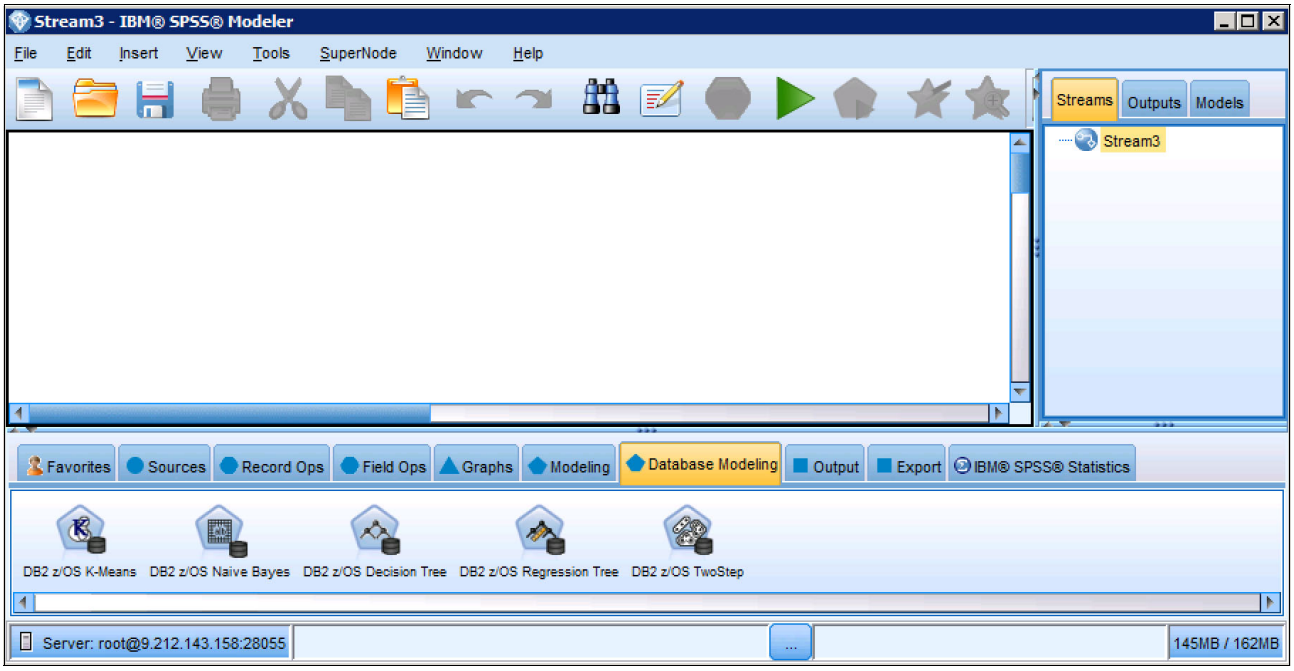


Figure 2-17 SPSS Modeler client showing "Database Modeling" palette with DB2 for z/OS predictive models

Let SPSS Modeler show you what SQL it generates for z/OS DB2

Seeing what SQL Modeler generates can be interesting, especially regarding in-database processing. By default, generated SQL is not displayed. To change what is displayed, select **Tools** → **Stream Properties** → **Options** → **Logging and Status**. On this page, enable the SQL display, as shown in Figure 2-18.

After you apply these options, you can view the generated SQL statements, upon execution of your stream, by selecting **Tools** → **Stream Properties** → **Messages**.

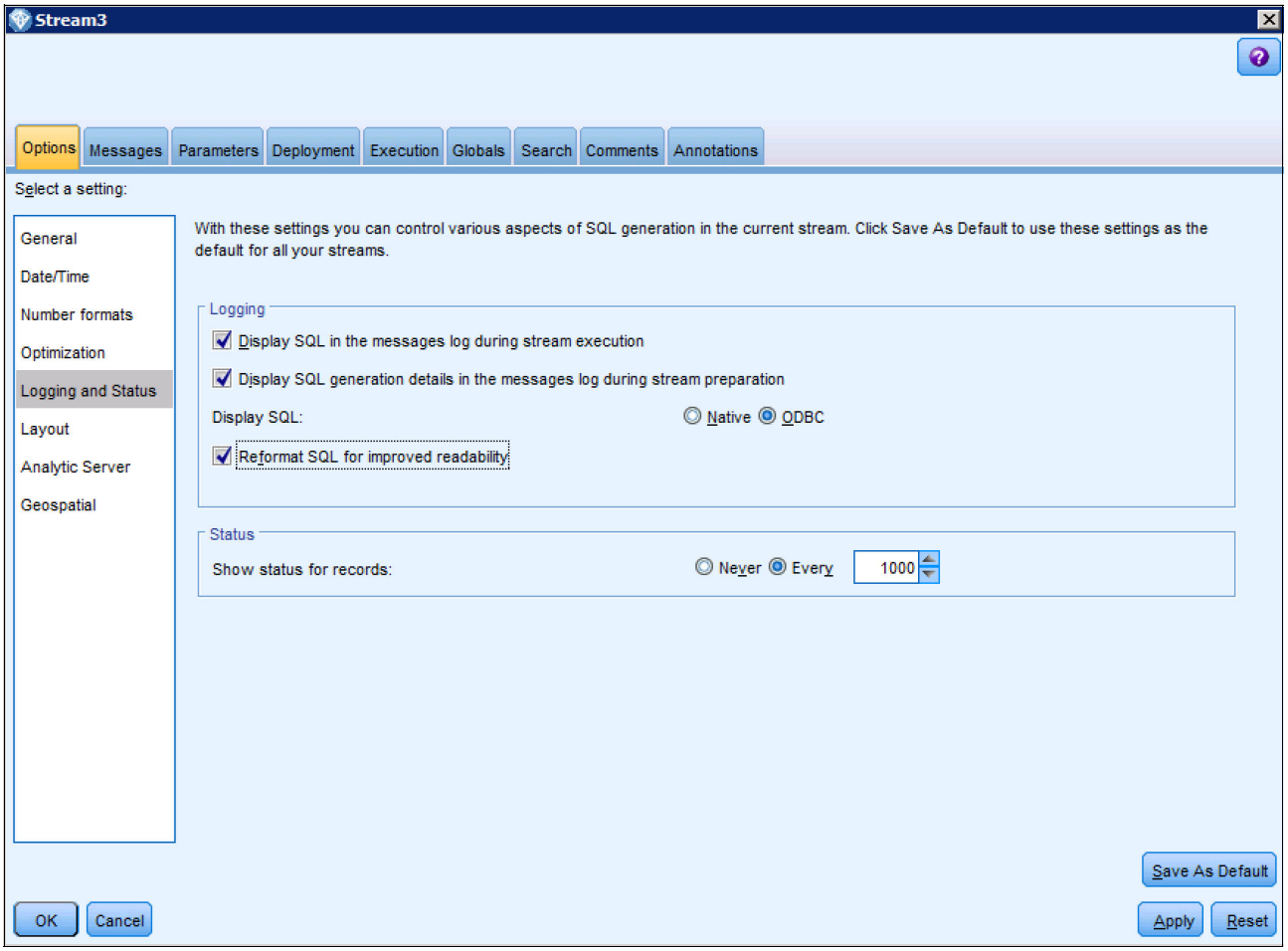


Figure 2-18 Options to view generated SQL statements upon stream execution

Optimize SQL for in-database operations

SPSS Modeler can optimize SQL for in-database operations. To enable this function, select **Tools** → **Stream Properties** → **Options** → **Optimization** and make the selections shown in Figure 2-19.

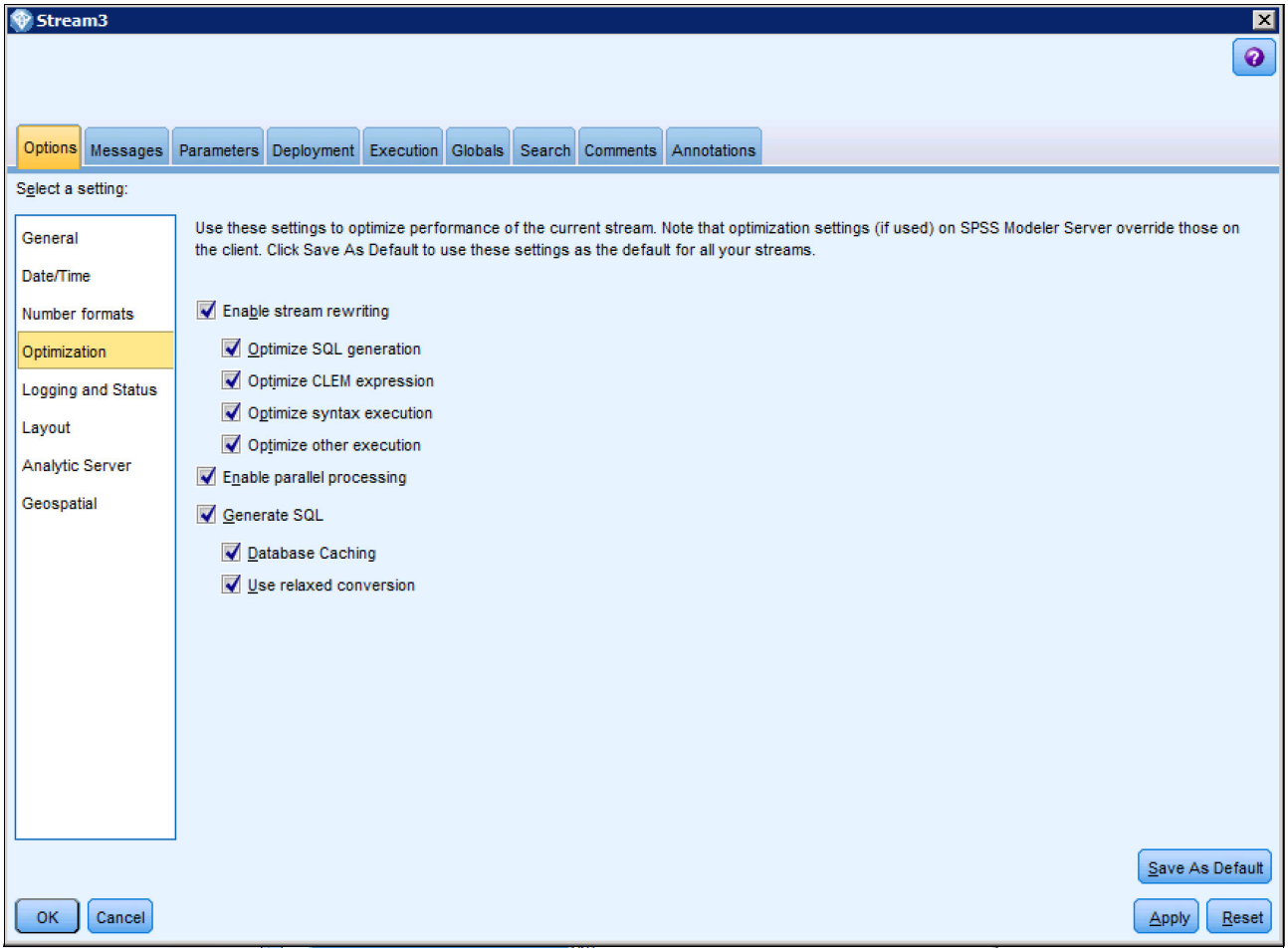


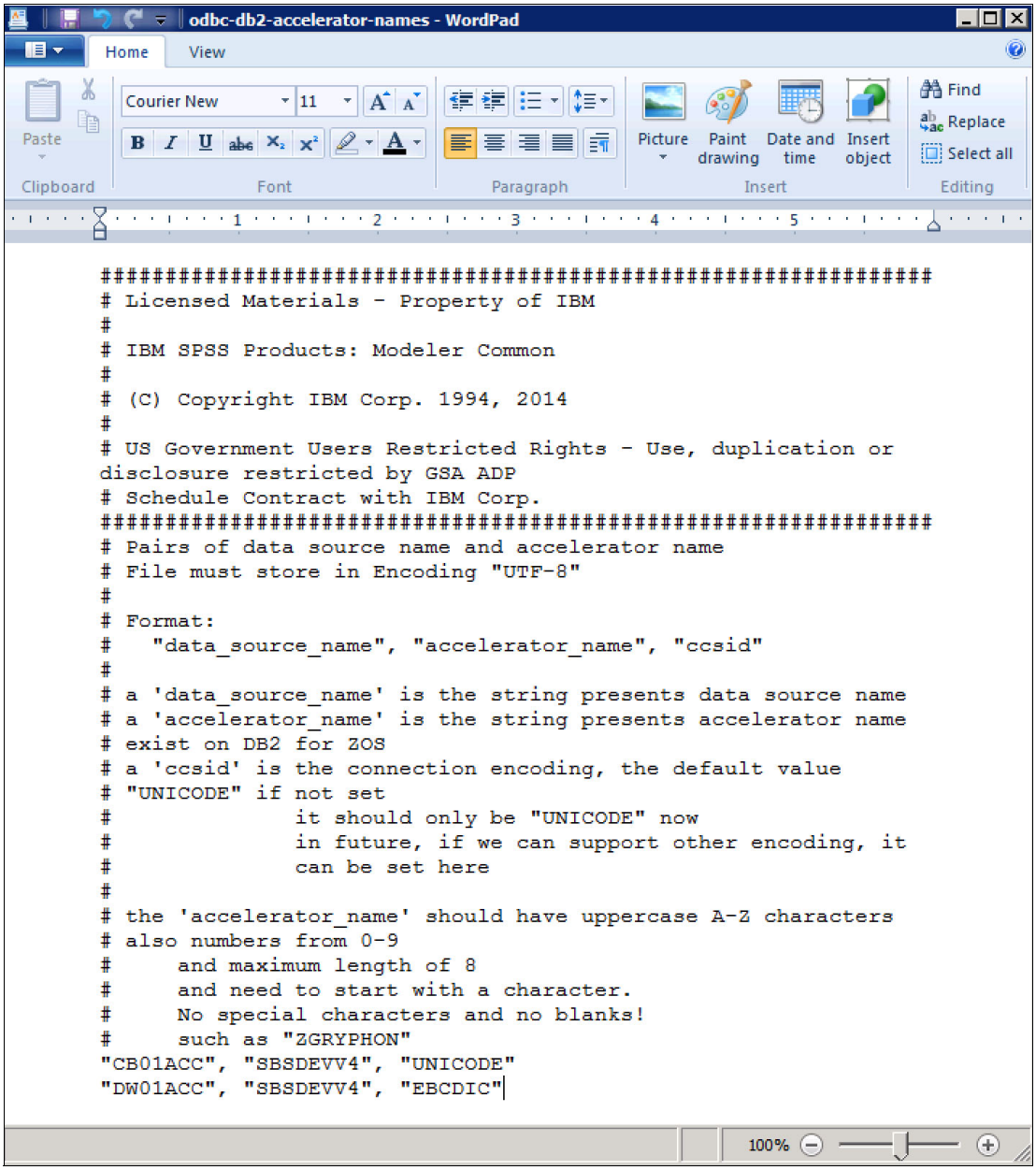
Figure 2-19 SPSS Client setting to optimize SQL for in-database operations

Client-connected data sources

Using the SPSS Modeler client without a stand-alone SPSS Modeler server is possible. Doing so is sensible if you can ensure that most or all of the triggered processing is running in-database. As in this case, z/OS DB2 and SPSS Modeler server are forwarding the requests scheduled by SPSS Modeler while the accelerator is doing most of the real work.

In this case, first make sure you have SPSS SDAP DataDirect ODBC DSN connections or DB2 ODBC DSN connections to your host. In the SPSS Modeler `config` directory (the default directory is `C:\Program Files\IBM\SPSS\Modeler\17.1\config`), edit the `odbc-db2-accelerator-names.cfg` file.

A sample configuration file is shown in Figure 2-20.




```
#################################################################
# Licensed Materials - Property of IBM
#
# IBM SPSS Products: Modeler Common
#
# (C) Copyright IBM Corp. 1994, 2014
#
# US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP
# Schedule Contract with IBM Corp.
#################################################################
# Pairs of data source name and accelerator name
# File must store in Encoding "UTF-8"
#
# Format:
#   "data_source_name", "accelerator_name", "ccsid"
#
# a 'data_source_name' is the string presents data source name
# a 'accelerator_name' is the string presents accelerator name
# exist on DB2 for ZOS
# a 'ccsid' is the connection encoding, the default value
# "UNICODE" if not set
#              it should only be "UNICODE" now
#              in future, if we can support other encoding, it
#              can be set here
#
# the 'accelerator_name' should have uppercase A-Z characters
# also numbers from 0-9
#     and maximum length of 8
#     and need to start with a character.
#     No special characters and no blanks!
#     such as "ZGRYPHON"
"CB01ACC", "SBSDEVV4", "UNICODE"
"DW01ACC", "SBSDEVV4", "EBCDIC"
```

Figure 2-20 Sample configuration file for client-connected data sources

2.2.4 SPSS Modeler server

If significant processing is to be done within SPSS Modeler, use a stand-alone server with large enough memory and file system space, preferably on Linux on z on the same CEC where your OLAP DB2 member is running, connected using HiperSockets.

2.2.5 Data sources in SPSS

Be sure you create two ODBC DSNs per database; you can use one for DB2 base tables and one for accelerated tables. For in-database transformation, it is required that the source nodes are accelerated tables. For the SPSS Scoring Adapter, the source table required to be scored is a DB2 table if used for batch scoring (if used for in-transactional scoring, you do the transaction processing out of the scope of SPSS Modeler client application anyway, and of course there you also run it in DB2).

The `odbc-db2-accelerator-names.cfg` file must have lines similar to the following lines:

```
“UB01ACCEL, “MYIDAA”, “UNICODE”
“XB01ACCEL”, “MYIDAA”, “EBCDIC”
```

In these lines, `UB01ACCEL` is a DSN with UTF8 Unicode encoding and Accelerator `MYIDAA`, and `XB01ACCEL` would be another DSN with Accelerator `MYIDAA` and EBCDIC encoding.

EBCDIC

EBCDIC data sources are not supported with the IBM SPSS Modeler prior to Version 18.

2.2.6 AQT_ANALYTICS_DATABASE variable

In AQTENV, a new environment variable, AQT_ANALYTICS_DATABASE, denotes a database for creation of accelerator-only output tables in INZA stored procedures and in generic streams for data transformation, driven from SPSS Modeler. The z/OS DB2 should create a new database for that and give all users of SPSS Modeler (or other exploiter of the INZA SPs) the privileges to use DDL statements (CREATE TABLE) in that database.

In addition, a current requirement is to add a file to the `config` subdirectory of SPSS Modeler server. Assuming you named your database AOTDB, the information shown in Figure 2-21 should be stored in the `odbc-db2-custom-properties.cfg` file.

```
zbapi:/usr/IBM/SPSS/ModelerServer/17.1/config # more odbc-db2-custom-properties.cfg
table_create_temp_sql_acc, 'CREATE TABLE <table-name> <(table-columns)> IN DATABASE AOTDB'
table_drop_temp_sql_acc, 'DROP TABLE <table-name> IN DATABASE AOTDB'
```

Figure 2-21 Specifying Accelerator Database name in the customProperties file of SPSS Modeler server

After making this type of change to the configuration file, you must *restart* SPSS Modeler.

2.2.7 User management in the SPSS Modeler server

To connect to the SPSS Modeler server, enter the correct user credentials. With a default setup, the credentials might be the root user ID of the Linux on z Systems image, which is probably not what you want. To change this behavior, see the User Authentication section of the *IBM SPSS Modeler Server Administration and Performance Guide*, or see the IBM Knowledge Center:

http://www.ibm.com/support/knowledgecenter/SS3RA7_17.0.0/clementine/server/adminguidesource/data_background.dita?lang=en

2.2.8 Installing SPSS Modeler scoring adapter for DB2 z/OS

The scoring adapter for DB2 z/OS provides score user-defined functions (UDFs) within z/OS. The scoring adapter runs as a UNIX System Services application. A WLM application environment that is specific to the scoring adapter makes sure you can tune it in a way so that all your service level agreements (SLAs) are reached. The scoring adapter runs completely within z/OS, integrated with DB2.

Application programs invoke the score functions in order to get score results. For them, the score UDFs must be installed within DB2, with required access rights given to the users under which the applications are running.

The SPSS Modeler also communicates with the scoring adapter by publishing new models. If publishing a new model, it adds data to specific DB2 tables, and the scoring adapter reads from them in order to get details about the models to be scored.

Prerequisites for the installation are as follows:

- Be sure to have DB2 for z/OS, with all prerequisites, up and running.
- Create a new WLM environment for the scoring adapter.
- Make sure the UNIX System Services is configured as required, and provides a file system that is usable by the scoring adapter with ability to write to a temporary directory. A common error situation is not having enough free space in UNIX System Services TMPDIR, or that the scoring adapter is not able to access it (because the HFS is no longer mounted, or because of missing privileges).
- Install the scoring adapter by using SMP/E.
- Run job control language (JCL), which do the required changes to DB2 z/OS, like creating the tables used to hold the models (as BLOBs), installing the required user-defined functions, set required RACF privileges and bind the scoring adapter package by using DB2 BIND.

Details of the installation are in *IBM SPSS Modeler 17.1 Scoring Adapter Installation*:

`ftp://public.dhe.ibm.com/software/analytics/spss/documentation/modeler/17.1/en/ModelerScoringAdapterInstall.pdf`

After the installation, you can see the following objects in DB2 for z/OS:

- Tables HUMSPSS.COMPONENTS and HUMSPSS.PUBLISHED_COMPONENTS in HUMSCFDB database.
- HUMSPSS.FLUSH_COMPONENT_CACHE and HUMSPSS.SCORE_COMPONENTS user-defined functions.

2.3 Real-time analytics lifecycle

This section describes the data analytics lifecycle in relation to the business scenario used here. Figure 2-22 on page 35 illustrates data loading, data preparation and transformation, data mining and predictive modeling, and scoring.

The z Systems warehousing and analytics software portfolio is continually evolving and improving. Over the past several years, it has grown to take on new roles. Software products available in the analytics portfolio can exist natively on z Systems, run under Linux for z Systems, and also is available on hybrid platforms.

Data is the new raw material for the 21st century. Data is meant to provide insight into business and offer a competitive edge. Online transaction processing (OLTP) is where this mission-critical data is generated based on business activity. OLTP usually is associated with high volume transaction processing applications like IBM CICS®, IMS™, and WebSphere®, and is a necessity for running a business because that is where the customer interactions and customer transactions are taking place. Data is gathered in the OLTP system and eventually the insights from the operational data is also consumed by the OLTP system, as shown in Figure 2-22.

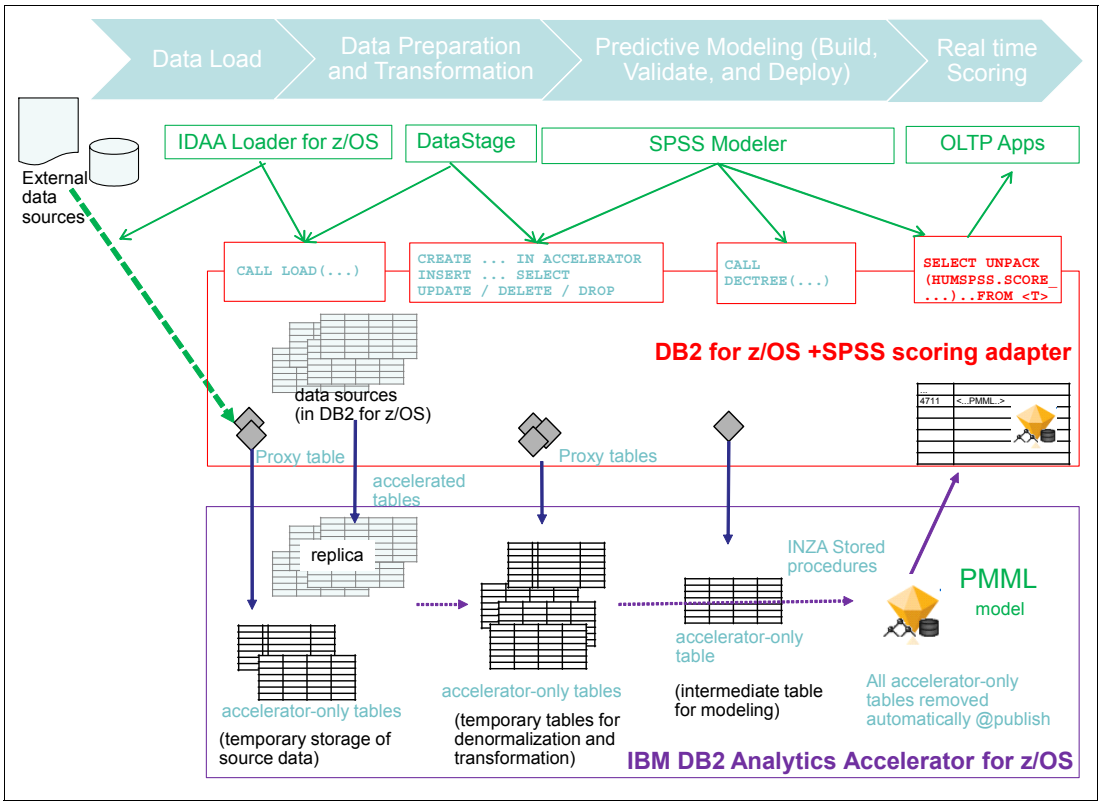


Figure 2-22 Analytics lifecycle in relation to real-time predictive analytics solution on z Systems

The z Systems environment can support the full lifecycle of a real-time analytics solution on a single, integrated system, combining transactional data, historical data, and predictive analytics.

The following four phases of the in-database analytics lifecycle are depicted at the top of Figure 2-22:

- Data Load or Data Integration phase
- Data Preparation and Transformation phase
- Predictive Modeling phase
- Real-time Scoring phase

Below the four phases, the figure shows the software solutions that can be used to accomplish the corresponding activities involved in the phases. For instance, the DB2 Analytics Accelerator Loader for z/OS can be used to load data into the accelerator (which essentially becomes the “play area” of the data scientist). The DB2 Analytics Accelerator Loader for z/OS can also be used to integrate data from various external sources. Additional information about DB2 Analytics Accelerator Loader for z/OS is in Chapter 3, “Data integration using IBM DB2 Analytics Accelerator Loader for z/OS” on page 41.

IBM DataStage® shown beside the DB2 Analytics Accelerator Loader for z/OS in Figure 2-22 on page 35 can be used to perform in-database transformation, (extract, transform, and load (ETL) or extract, load, and transform (ELT)), from external data sources and so on.

After all the external data is loaded into the accelerator, further data preparation and transformation can be done in-database by using IBM SPSS Modeler. For more information about IBM DataStage and IBM SPSS Data preparation and transformation, see Chapter 4, “Data transformation” on page 83.

The DB2 for z/OS area of Figure 2-22 on page 35 shows a glimpse of what kind of API is involved in the four phases of the in-database analytics lifecycle. This area also reaffirms that the data is always under the control of DB2 for z/OS even when the loading, transforming, and modeling activities are taking place in the accelerator, which is symbolically represented by the diamond shaped proxy tables for each AOT holding the data. The small rectangular boxes with rows and columns represent tables. The tables marked as “Data sources (in DB2 for z/OS)” will be used only by the OLTP transactions whereas the replica tables shown within the accelerator area (bottom section of Figure 2-22 on page 35) will be used by the in-database transformation and in-database modeling processes.

Note: The replica tables are not AOTs but they are the accelerated tables pertaining to each of the corresponding DB2 for z/OS source tables. All other tables shown in the Accelerator area are AOTs. Some of the AOTs can be used to hold external data (populated by DB2 Analytics Accelerator Loader for z/OS or IBM DataStage) and the rest of the AOTs might be created by SPSS Modeler to hold transformed and cached node data. The AOT marked as `intermediate table for modeling` will be the one used by INZA stored procedures to build the predictive analytics model. For more details about modeling, see Chapter 5, “Data modeling using SPSS and DB2 Analytics Accelerator” on page 107.

The generated model nugget is shown in the bottom right corner of Figure 2-22 on page 35. When the model is published successfully, a pair of DB2 for z/OS tables will be populated and all the temporary AOTs are deleted automatically. Finally, the model information on the DB2 tables can be used for real-time scoring. More details about real-time scoring are in Chapter 6, “Model scoring” on page 117.

2.3.1 Swim lane diagram of in-database analytics lifecycle

Figure 2-23 shows the key process interactions involved in a typical in-database analytics lifecycle. This swim-lane diagram is an attempt at explaining how SPSS, DB2 for z/OS, and DB2 Analytics Accelerator interact from a data scientist's perspective.

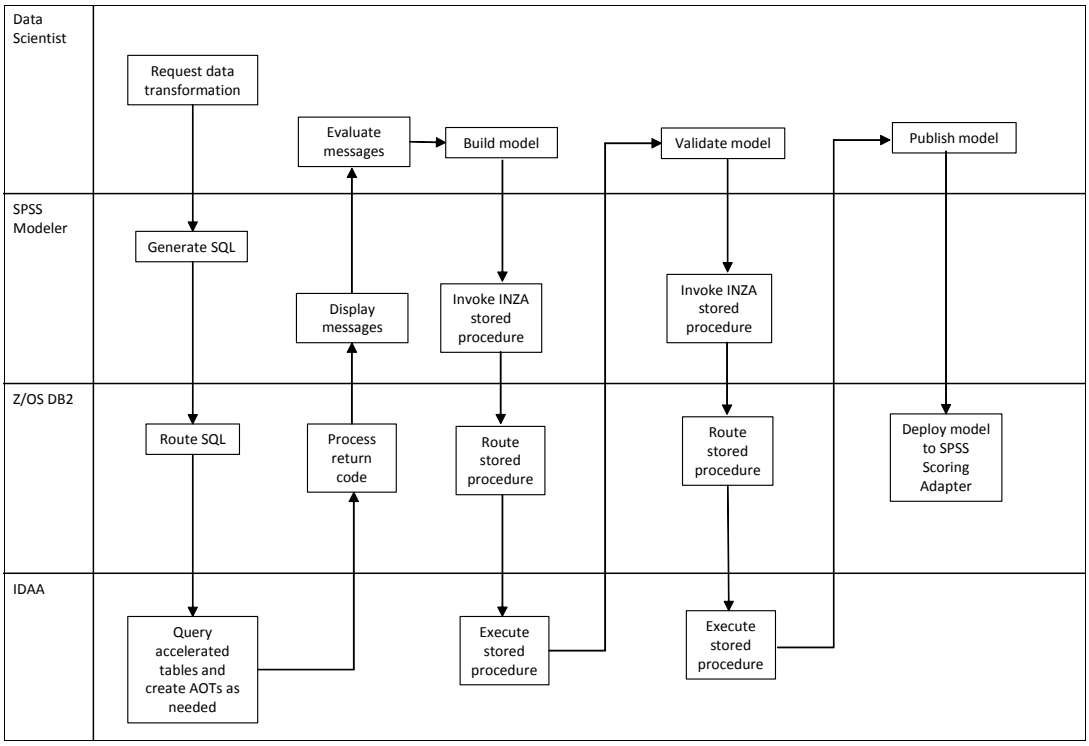


Figure 2-23 Swim-lane diagram showing in-database analytics process interactions

2.3.2 Interaction between a DB2 DBA and a data scientist

This section summarizes the high level responsibilities and interactions between a database administrator (DBA) and a data scientist with reference to the solution discussed in this book.

Database administrator (DB2 for z/OS)

The DBA makes sure that tables used for analytics are available to the data scientist. As always, the DBA or the security administrator makes sure that exactly the required access privileges are granted to appropriate userids in order to access the data. The tables required for accelerated in-database modeling are loaded by the DBAs through the Data Studio GUI or by other tools capable of invoking the DB2 Analytics Accelerator stored procedures.

Data scientist

The data scientist interactively creates SPSS modeling streams. SPSS Modeler can help to merge tables (using SQL joins), to filter or transform input data. It also provides advanced visualization capabilities and offers various predictive analytics modeling algorithms for use by the data scientist. For real-time analytics, the data scientist makes sure that the generated SQL statement pertaining to the scientist's selected model can achieve desired performance goals by working with the DBAs.

2.3.3 Key strengths of various components

Table 2-1 lists the key strengths of various components that make up the real-time (in-database) analytics solution discussed in this book.

Table 2-1 Key strengths of components that can be used in your analytics solution on z

Your real-time analytics requirement	Software Solution (component)
Highly scalable secured environment for your mission-critical data	IBM DB2 for z/OS
High transaction throughput capability for applications accessing your data	
Perform predictive analytics on your mission-critical data	IBM SPSS Modeler with SPSS Scoring Adapter for DB2 z/OS
Frequently build predictive models based on operational data	IBM DB2 Analytics Accelerator for z/OS appliance
Enable machine learning capability on your OLTP applications on z Systems	
Combine data from other sources (VSAM, IMS, Oracle, social media, and so on) and perform analytics	IBM DB2 Analytics Accelerator Loader for z/OS tool
Combine business rules with your analytics predictions	IBM Collaboration and Deployment Services, and Operational Decision Manager
Perform data transformation from varied sources	IBM DataStage
Create business analytics visualization reports	IBM Cognos® for z/OS
In-application scoring as part of your real-time analytics solution	Zementis

Figure 2-24 shows various software packages and tools that can be part of the real-time in-database analytics lifecycle shows in Figure 2-22 on page 35. At the core of this solution is the IBM DB2 Analytics Accelerator for z/OS, as depicted in the center of this lifecycle diagram.

Figure 2-24 also can be correlated with the key strengths of software components and tools that can be used in this solution.

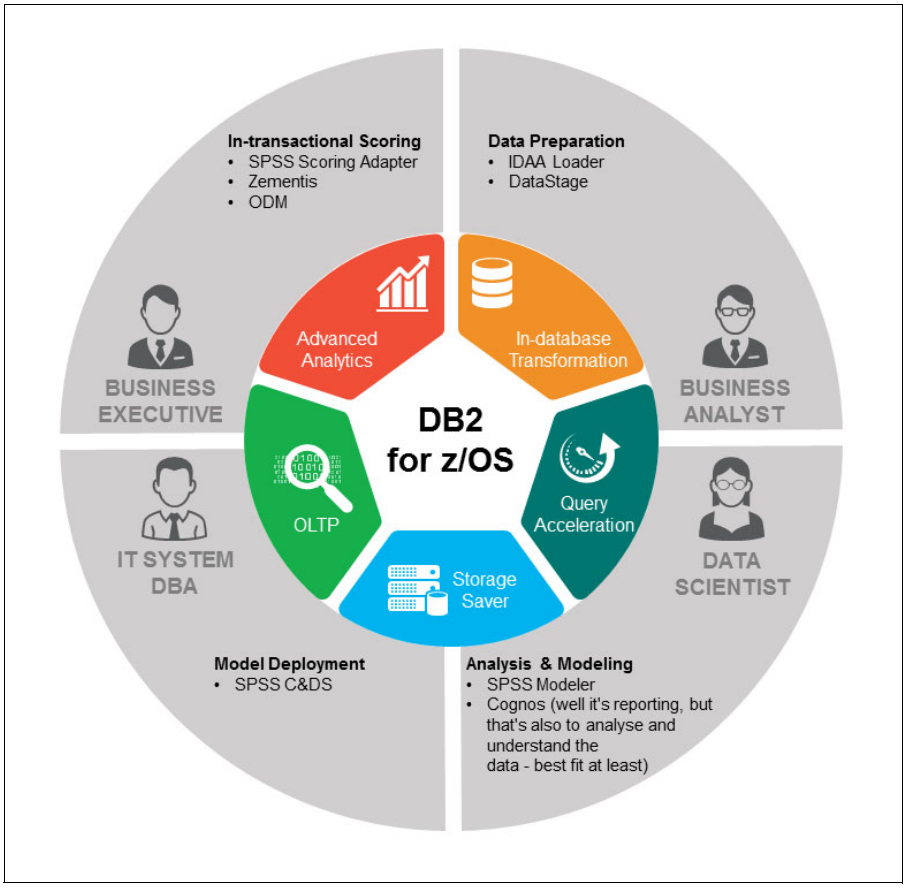


Figure 2-24 Tools and actors of real-time analytics lifecycle with in-database processing at the core

3

Data integration using IBM DB2 Analytics Accelerator Loader for z/OS

Data integration using DB2 Analytics Accelerator Loader for z/OS v2.1 is described in this chapter.

This chapter contains the following topics:

- Functional overview
- Getting started
- Scenarios
- System Management Facility (SMF)

© Copyright IBM Corp. 2016. All rights reserved.

3.1 Functional overview

The IBM DB2 Analytics Accelerator Loader for z/OS v2.1, referred to as just *Loader* in this publication, is a tool designed to load data directly into the IBM DB2 Analytics Accelerator for z/OS, referred to here as *Accelerator*. Loader reduces elapsed time while consuming less CPU when compared to traditional methods used to load data into Accelerator. Loader also uses a less complicated process capable of moving data from DB2 and non-DB2 sources, both remote and local, directly into the Accelerator. Although Loader has the ability to load data into both DB2 for z/OS and the Accelerator in parallel, loading data directly into an Accelerator without the need to first load that data into DB2 for z/OS is one of its major strengths.

That simpler process allows data to be moved and transposed in fewer steps involving less administrative time. Loader is a complete end-to-end solution that manages all your Accelerator data movement in a single product.

Note: Loader's strength is its ability to take data and load it directly into the DB2 Analytics Accelerator without the need of first moving that data into DB2.

The data source is accessed and converted to the appropriate format and then loaded directly into the Accelerator without the necessity of intermediate files. Extracting the data prior to a load is no longer necessary.

By using Loader, data can be loaded into just the Accelerator or into both the Accelerator and DB2 for z/OS in parallel. Both the LOAD REPLACE or LOAD RESUME options are available when using Loader.

Along with the performance advantage naturally inherent in a process that does not have to "go through" DB2 to get to the Accelerator, Loader makes extensive use of the IBM z Systems Integrated Information Processor (zIIP). By allowing Loader to take advantage of zIIP specialty processors, a percentage of Loader's CPU costs can be redirected to processors that have no IBM software charges, thus freeing up general computing capacity and lowering overall total cost.

IBM allows customers to purchase additional processing power exclusively for select data serving environments without affecting the total million service units (MSU) rating or machine model designation on zIIP, or z Systems Application Assist Processor (zAAP).

Originally, formatting the input was the client's responsibility, but now Loader helps with the data mappings.

Loader accepts input from multiple sources and multiple locations. In addition to IMS, Oracle, SMF, z Systems Data Compression (zEDC), Virtual Storage Access Method (VSAM), physical sequential existing locally, these sources could also be located on a different Sysplex or on Linux on z Systems. Figure 3-1 on page 43 shows the relationship of the various data sources to the Loader components. The Accelerator Loader Server, referred to in this document as *HLV* or *HLV started task*, performs the mapping and movement from any one of the data sources and prepares the batch JCL for execution on z/OS. Data is passed from HLV to the Accelerator Loader component, referred to as the *HLO* in this document, using shared memory objects. HLO calls the DB2 utility DSNUTILB, invokes the Loader intercept, and loads the data into the Accelerator.

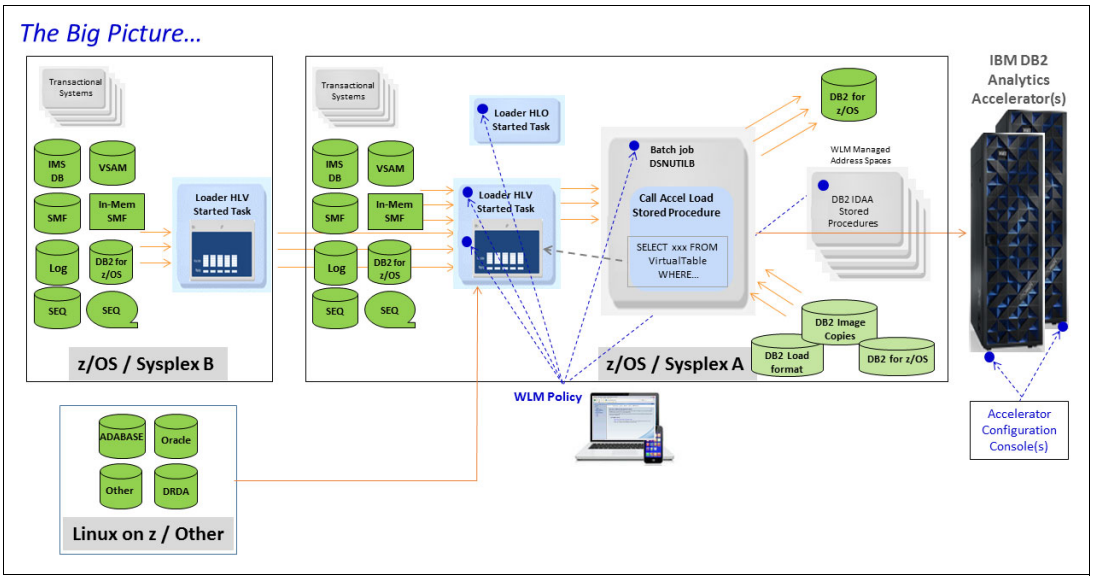


Figure 3-1 An overview of the entire Loader process

Loader also now provides a high availability load by loading multiple accelerators. Currently, you need to load Accelerator 1, then Accelerator 2, then Accelerator 3, and so on, but after applying the PTF, one statement will allow multiple accelerators to be loaded at once.

DB2 Analytics Accelerator Loader for z/OS v2.1 is documented in the IBM Knowledge Center:

`http://www.ibm.com/support/knowledgecenter/SSKRKK_2.1.0/topics/hlou_welcome.html`

Ultimately, the DB2 Analytics Accelerator Loader provides a simple end-to-end process to consolidate data from multiple sources into an Accelerator.

3.1.1 Loader v2.1 enhancements

DB2 Analytics Accelerator Loader for z/OS, V2.1 provides the following benefits:

- Minimized application impact when loading or refreshing external DB2 for z/OS data by loading in parallel to both DB2 for z/OS and DB2 Analytics Accelerator.
- Increased flexibility when performing business analytics by enabling the load of a consistent set of data with an historical point-in-time into DB2 Analytics Accelerator for query consideration.
- A z Systems solution to quickly load data directly to the Accelerator from many data sources, including IMS, VSAM, sequential files, and Oracle, without the need to land or stage the data in intermediate files. The entire process from data extraction, conversion, and loading is automated in a single batch job step.
- An easy-to-use SQL interface to identify data to be loaded into the Accelerator, including the ability to join and load data from multiple sources to a single Accelerator table.
- Potential further reduction in CPU cost associated with data loading into DB2 Analytics Accelerator as DB2 Analytics Accelerator Loader directs such processing to be executed under an enclave service request block and such processing is authorized to execute on a zIIP specialty engine 1.
- Support for IBM Change Data Capture for z/OS (CDC) replication when loading to both a DB2 table and an Accelerator table.

- The ability to append data to an existing table. This LOAD RESUME capability can help to reduce elapsed time and CPU consumption by avoiding the need to reload the entire table to include the new data to be loaded.
- The ability to automatically create the table on the DB2 Analytics Accelerator if it does not exist and to enable the table for acceleration after the load is successful. This can help to improve user efficiency and may reduce the manual steps required to prepare a table on the appliance for acceleration.
- When performing a load from an external file, you can load SYSREC data that is already in DB2 internal row format. The DB2 UNLOAD utility supports an option to unload the data from a table in FORMAT INTERNAL. This enhancement provides the benefits of reduced CPU consumption and elapsed time in both the UNLOAD and in Accelerator Loader jobs.

For a complete description of DB2 Analytics Accelerator Loader v2.1, see the January 25, 2016 announcement letter:

http://www.ibm.com/common/ssi/cgi-bin/ssialias?infotype=an&subtype=ca&appname=gpateam&supplier=897&letternum=ENUS216-015#hardx

3.1.2 Loader methods to move data

Loader uses four methods to move data in the Accelerator:

- Dual. Specifies options for loading table data into both the accelerator and DB2 from an external data input file.
- Accelerator Only. Specifies options for loading table data into only the accelerator from an external data input file.
- Consistent. Specifies options for loading data for multiple tables into the accelerator from a cataloged DB2 image copy.
- Image copy. Specifies options for loading data for a single table into the accelerator from a user-defined DB2 image copy.

These methods are discussed in this chapter.

3.1.3 Components and interfaces

This section describes the components and interfaces inherent to Loader.

HLO: Accelerator Loader

The following components are inherent to the HLO Accelerator Loader, which calls the DB2 utility DSNUTILB:

- The batch load utility along with intercept processing.
- The Accelerator Loader communicates with the Accelerator Loader Server with the purpose of loading directly from data sources.
- The DSNUTILB batch job communicates with IDAA and sends data to IDAA.
- Supports loading from a sequential file (without the Accelerator Loader server) when the file is in DB2 Load File format (external load).
- Supports loading the Accelerator from image copies and logs (consistent load).
- The procedure name usually associated with batch component is `hloidPROC`.

In Example 3-1, a **DISPLAY** command is issued for the Accelerator Loader started task to verify that the intercept is enabled. Message HLOS0817I identifies that the local DSNUTILB intercept is indeed enabled. The **DISPLAY** command in the example is preceded with a slash (/) only because the command was issued from SDSF.

Example 3-1 Verifying Loader intercept is active

```
/F HLO1PROC,DISPLAY INTERCEPT,ALL

HLOS0814I 120 15:44:18.55 Command issued: DISPLAY INTERCEPT,ALL
HLOS0817I 120 15:44:18.55 LOCAL  DSNUTILB intercept status is: ENABLED
HLOS0817I 120 15:44:18.55 GLOBAL DSNUTILB intercept status is: ENABLED
HLOS0822I 120 15:44:18.55 DB2 SSID=DBZZ 1110 ID=HLO1 DSNUTILB interception is
installed
```

HLV: Accelerator Loader Server

The Accelerator Loader Server (HLV) performs all access to the data sources for purposes of direct loading. This includes mapping and transforming the data from its source form to a relational record form.

Accelerator Loader Server is accessed through the Data Studio plug-in. It drives the mapping process and builds the appropriate JCL to perform the load process.

The procedure name usually associated with Accelerator Loader Server is `hlvidPROC`.

Interactive System Productivity Facility (ISPF)

The ISPF interface provides options for building JCL to load or refresh data on the Accelerator. It will generate the JCL, giving the options to save the JCL in a data set of your specifications.

Accelerator Loader studio

The Accelerator Loader studio is a plug-in for the IBM Data Studio. It can be used to quickly transform and load relational and non-relational data to an Accelerator. This plug-in allows for real-time access to the data, which can be read directly from the mapped source and transformed while it is being loaded into the Accelerator tables.

IBM Data Studio enables developers and administrators to create and manage heterogeneous database environments using Eclipse technology. Data Studio is a fully licensed product available at no charge and with no time restrictions that can be downloaded from the IBM Support website:

http://www.ibm.com/support/docview.wss?uid=swg24037681

Accelerator Loader started task

Accelerator Loader runs as a started task on a z/OS. The started task receives input from the interfaces through the supervisor call (SVC) communicating with the DB2 subsystems to run the JCL and load data to the Accelerator. A single started task can process simultaneous requests from multiple users across the system. The started task must be started before you can perform any product functions.

3.2 Getting started

One of the first steps to take before installing Loader is the verification of all the prerequisites. To ensure that you have the most recent prerequisites, preinstallation, and installation notes, see the following locations in the IBM Knowledge Center:

- Preparing to customize

  https://ibm.biz/BdrB7d

- DB2 Analytics Accelerator Loader V2.1 documentation

  https://ibm.biz/BdrB7X

3.2.1 Installation

For information about installing the Accelerator Loader studio (required component of the DB2 Loader), see the IBM Knowledge Center:

https://ibm.biz/BdrB7b

DB2 Loader can be fully installed with the SMP/E program. The installation instructions for the actual DB2 Loader product, along will all SMP/E requirements, are in the *Program Directory for IBM DB2 Analytics Accelerator Loader for z/OS* publication:

http://publib.boulder.ibm.com/epubs/pdf/i1346130.pdf

This document is program number 5639-OLE with FMIDs HHLO210, HALE210, H25F132.

If data is loaded from distributed relational database architecture (DRDA) sources, DB2 Analytics Accelerator Loader for z/OS, V2.1 requires one of the following items:

- IBM DB2 Advanced Enterprise Server Edition, V10.5 (5725-L47)
- IBM InfoSphere® Federation Server, V9.7 (5724-N97)
- IBM InfoSphere BigInsights® (IBM Big SQL), V1.0 (5725-C09)

Analytics Accelerator Loader uses the partitioned data set extended (PDSE) format for the SHLOLOAD, SHLVLOAD, and SHLVRPC target libraries. Some operational differences exist between PDS and PDSE data sets. Make sure you understand the differences between a PDS and PDSE data set before using PDSE. Some of the differences are described at the following web page:

https://www.ibm.com/support/knowledgecenter/SSLTBW_2.1.0/com.ibm.zos.v2r1.idad400/d4289.htm

3.2.2 Customization

After Loader is installed, it must be customized for use at your installation. See the Customizing DB2 Analytics Accelerator Loader web page:

https://ibm.biz/BdrBWn

Also see the DB2 Analytics Accelerator Loader V2.1 documentation web page:

https://ibm.biz/BdrB7X

3.2.3 Workload Management (WLM) performance goals

With z/OS WLM, you can define performance goals and then assign a business importance to each goal. z/OS will decide the resources, such as CPU and storage, that should be allotted so that goal can be met. The system and its resources are constantly monitored and adjusted to meet those defined goals.

The DB2 Analytics Accelerator Loader components should be defined to WLM to ensure that performance is maximized. This includes started tasks, servers, batch workloads, and stored procedure address spaces. Along with Loader, resource management of the load that is running on the Accelerator must also be examined. A typical suggestion is to leave the default settings unless, after observation, you need to change the prioritization. However, the Accelerator resource management is beyond the scope of this book.

Terms used

The following terms are used in this book:

Response time goal
: Indicates how quickly you want work to be processed.

Velocity goal
: Defines how fast work should run when the work is ready.

Discretionary goal
: Indicates low-priority work with no particular performance goal.

Enclave
: Is the mechanism for managing and monitoring transactions that can span multiple dispatchable units, SRBs, and tasks, in one or more address spaces.

Dispatching priority (DP)
: Is a number in the range of 0 - 255 associated with a dispatchable unit of work (TCBs or SRBs). DP is placed in an address space or in an enclave control block to indicate the priority of the dispatchable units for getting CPU service. DP is assigned by WLM and ready work with the highest priority is dispatched first.

Service request block (SRB)
: Represents a request to execute a system service routine. SRBs are typically created when one address space detects an event that affects a different address space; they provide one mechanism for communication between address spaces.

Task control block (TCB)
: Represents a task executing within an address space, such as user programs and system programs that support the user programs.

  TCBs or SRBs may be dispatched on any logical processor of the type required; standard CP or zIIP.

SYSOTHER
: Is a “catch all” service class for work with no classification and is assigned to a discretionary goal.

Accelerator Loader Server

WLM considers the Accelerator Loader Server component to consist of two parts; both parts must be defined to WLM:

- The Data Server started task

  The Accelerator Loader Data Server (`hlvidPROC` started task) provides key support for *direct load from source* Loader operations. The Accelerator Loader Data Server started task will initiate the HLV threads (enclaves) and is responsible for conducting I/O on behalf of the same.

  Accelerator Loader Data Server should be classified to an existing or new service class with the appropriate goals relative to your existing work on the same z/OS image.

- The Loader HLV subsystem that manages the enclaves

  The WLM subsystem HLV, a new subsystem defined during the installation of Loader, manages the enclaves running on behalf of Loader jobs and will do the bulk of the work, including data extraction, data conversions, and managing the pipe into DSNUTILB which in turn will pipe into the Accelerator. See Example 3-2.

  HLV should be classified to the same service class as `hlvidPROC`.

Example 3-2 HLV start task proc example

```
//HLVS    PROC SSID=HLVS,OPT=INIT,TRACE=B,MSGPFX=HLV,
//             MEM=32G,REG=0000M,HLQ='hlq',HLQ1='hlq1',HLQ2='hlq2'
//*
//IEFPROC EXEC PGM=HLV2IN,REGION=&REG,TIME=1440,DYNAMNBR=100,MEMLIMIT=&MEM,
//              PARM='&OPT,&SSID,&TRACE,&MSGPFX'
 //*
//* additional DD statements intentionally excluded
//*
```

Note: Currently there is no means, no qualifiers exist, for granular classification of various work within the HLV subsystem.

Do *not* allow work for the HLV subsystem or the Accelerator Loader Data Server to default to the WLM service class SYSOTHER. If work that uses enclaves is not classified, that work will default to SYSOTHER.

Accelerator Loader started task

The Accelerator Loader started task (`hloidPROC`) provides the basic services for DSNUTILB plus restart and DB2 catalog access. The Accelerator Loader should be classified to a service class with less than or equally aggressive goals as compared to the Accelerator Loader Server component, the `hlvidPROC` started task, shown in Example 3-3.

Example 3-3 HLO started task example

```
//HLOHLO1 PROC OPT=HLO1OPTS,PLCY=HLO1PLCY
//*
//* additional DD statements intentionally excluded
//*
//STCRUN   EXEC PGM=HLOSINIT,TIME=1440,REGION=0M
```

Loader batch job

All Loader tasks are initiated through a JES2 batch job, a sample of which is shown in Example 3-4. These batch jobs call the DB2 utility DSNUTILB with appropriate Loader and DB2 LOAD control cards, invoke the Loader intercept, and load the data into the Accelerator.

Classify existing service classes or new service classes with goals equal to or more aggressive than the HLV WLM subsystem. For critical Loader work, consider using a more aggressive service class (higher importance, velocity, or both) than the rest of the Loader work. This is a way to differentiate resource management of different Loader jobs.

For granular reporting, use unique WLM report classes for all of the Loader components, including the batch jobs just described.

Example 3-4 Loader batch JCL

```
//ACCLLOAD EXEC PGM=DSNUTILB,REGION=0000M,PARM=('ssid','utility-id')
//*
//* additional DD statements intentionally excluded
//*
//HLODUMMY DD  DUMMY
```

Classification Starting point only

Although not a recommendation, a possible starting point for a workload classification could be the values used on the z/OS system used to gather the metrics in this Redbooks publication. However, do not consider these values to be “the correct” way to set up WLM but rather one example of how a system might be set up. Table 3-1 contains an example of how Loader was set up in WLM in a test environment. This figure also includes samples of other subsystem classifications that Loader might interface with.

Table 3-1 Loader set up in our WLM test environment: goal examples

Work	WLM Subsystem for Classification	WLM Service Class	Goal Importance	Goal
Critical STCs: IRLM, TCPIP, etc.	STC	SYSSTC	1	
High STCs: DBM1, DDF, MSTR, etc.	STC	STCHIGH	1	Vel = 90
DB2 Transactions / Queries	DDF, JES2, TSO, etc.	Likely various	Likely various	Multi-period Response Time Velocity
.....				
Critical Loader Job	JES2	BATLDHI	3	Velocity = 80
Loader Job	JES2	BATLDMED	4	Velocity = 50
Accelerator Loader Server Enclaves	HLV	ACLOADR	4	Velocity = 50
Accelerator Loader Server started task (*hlvidPROC)*	STC		4	Velocity = 50
Loader started task (*hloidPROC)*	STC		4	Velocity = 50

Note: When implementing resource management for an Accelerator Loader workload, update considerations should not be limited to the z/OS WLM service definitions. Also consider resource management of the load process running on the Accelerator. This portion of the end-to-end load resource management is controlled from the Accelerator configuration console. Although the typical preference is to leave the default settings, you might need to change the prioritization, for example, based on time of the day.

A task’s assigned dispatching priority is critical to that task’s performance in a z/OS environment. WLM is responsible for adjusting the dispatching priority based on velocity and importance defined to a service class. For that reason, an important steps is to take time to ensure that the dispatching priority for the Accelerator Loader started tasks are set correctly with respect to other dispatching priorities. The dispatching priority determines the order in

which a task uses the processor in a multitasking environment. The Accelerator Loader dispatching priority must be lower than the priority values for the DB2 subsystems that Accelerator Loader will use, but higher than the priority values for any DB2 LOAD utilities for which Accelerator Loader will perform processing. Set the dispatching priorities for these items in the following order, from highest to lowest priority:

1. The address spaces of the DB2 subsystems that Accelerator Loader will use (highest dispatching priority).
2. The Accelerator Loader started tasks.
3. The DB2 LOAD utility that Accelerator Loader intercepts (any dispatching priority under the Accelerator Loader started tasks).

Display WLM Information

Details about the level of WLM being used and the WLM service policy currently active can be displayed by using the IBM z/OS MVS™ **DISPLAY WLM** command. An example of the output from this command is shown in Example 3-5.

Example 3-5 Results of a z/OS MVS DISPLAY WLM command

```
D WLM
IWM025I  17.03.00  WLM DISPLAY 121
  ACTIVE WORKLOAD MANAGEMENT SERVICE POLICY NAME: IZASBASE
  ACTIVATED: 2016/04/14  AT: 12:54:50  BY: USERxx      FROM: P52
  DESCRIPTION: IZASBASE Changed 01/25/2013 USERxx
  RELATED SERVICE DEFINITION NAME: IZASBASE
  INSTALLED: 2016/04/14  AT: 12:54:39  BY: USERxx      FROM: P52
  WLM VERSION LEVEL:       LEVEL029
  WLM FUNCTIONALITY LEVEL: LEVEL011
  WLM CDS FORMAT LEVEL:    FORMAT 3
  STRUCTURE SYSZWLM_WORKUNIT STATUS: DISCONNECTED
  STRUCTURE SYSZWLM_7EB72964 STATUS: DISCONNECTED
  STATE OF GUEST PLATFORM MANAGEMENT PROVIDER (GPMP): INACTIVE
```

WLM managed stored procedure address spaces

Loader takes advantage of two WLM application environments. In the z/OS system that was used to gather the metrics used in this book, the environments were as follows:

- DSNWLM_IDAA. This environment is used by all of the Accelerator stored procedures. These examples of stored procedures use this environment and are also used by Loader:
  - SYSPROC.ACCEL_ADD_TABLES
  - SYSPROC.ACCEL_REMOVE_TABLES
  - SYSPROC.ACCEL_LOAD_TABLES
  - SYSPROC.ACCEL_SET_TABLES_ACCELERATION
- DSNWLM_UTILS. This environment is used for processing by the DB2 utility stored procedures DSNUTILU and DSNUTILS (deprecated). This environment is defined to DB2 when DB2 is installed. It is modified by Loader to include the SHLOLOAD load library in the STEPLIB concatenation (see JCL in Example 3-8 on page 52).

To validate the working status of these two stored procedures, the WLM display commands in Example 3-6 can be used. These two display examples use our naming convention for the environment names; your names will differ.

Example 3-6 Display example of WLM managed stored procedure address spaces status

```
D WLM,APPLENV=DSNWLM_IDAA
IWM029I  10.00.30  WLM DISPLAY 183
  APPLICATION ENVIRONMENT NAME     STATE     STATE DATA
  DSNWLM_IDAA                      AVAILABLE
  ATTRIBUTES: PROC=DBZZWLMI SUBSYSTEM TYPE: DB2

D WLM,APPLENV=DSNWLM_UTILS
IWM029I  10.03.10  WLM DISPLAY 185
  APPLICATION ENVIRONMENT NAME     STATE     STATE DATA
  DSNWLM_UTILS                     AVAILABLE
  ATTRIBUTES: PROC=DBZZWLMU SUBSYSTEM TYPE: DB2
```

If the characteristics of these WLM managed stored procedure address spaces change, for example NUMTCB keyword was modified, the environment should be refreshed.

Example 3-7 shows two samples of the WLM modify command to refresh the environments. Again, change the environment names used in the example to match your system.

Example 3-7 A WLM managed stored procedure address space refresh

```
V WLM,APPLENV=DSNWLM_IDAA,REFRESH
IWM032I VARY REFRESH FOR DSNWLM_IDAA COMPLETED

V WLM,APPLENV=DSNWLM_UTILS,REFRESH
*IWM031I VARY REFRESH FOR DSNWLM_UTILS IN PROGRESS
 IWM034I PROCEDURE DBZZWLMU STARTED FOR SUBSYSTEM DBZZ 501
 APPLICATION ENVIRONMENT DSNWLM_UTILS
 PARAMETERS DB2SSN=DBZZ,APPLENV='DSNWLM_UTILS'
 IWM032I VARY REFRESH FOR DSNWLM_UTILS COMPLETED
 $HASP100 DBZZWLMU ON STCINRDR
 $HASP373 DBZZWLMU STARTED
 -                                        --TIMINGS (MINS.)--
  ----PAGING COUNTS---
 -JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV  PG
    PAGE   SWAP    VIO SWAPS
 -DBZZWLMU          DBZZWLMU    00     20 ******    .00     .9    249   0
       0      0      0     0
 -DBZZWLMU ENDED.  NAME-                     TOTAL CPU TIME=   .00  TOTAL
  ELAPSED TIME=    .9
 $HASP395 DBZZWLMU ENDED - RC=0000
```

Example 3-8 is one sample of the JCL used for the started task procedure for the WLM managed stored procedure address spaces. It represents the DSNWLM_UTILS environment included with DB2 with the modifications to include Loader's intercept module (SHLOLOAD load library).

Example 3-8 Sample utility WLM managed stored procedure address space procedure

```
//ssidWLMU PROC APPLENV='DSNWLM_UTILS',
//            DB2SSN=ssid,RGN=OK,NUMTCB=1
//IEFPROC EXEC PGM=DSNX9WLM,REGION=&RGN,TIME=NOLIMIT,
//            PARM='&DB2SSN,&NUMTCB,&APPLENV'
//STEPLIB  DD  DISP=SHR,DSN=CEE.SCEERUN
//         DD  DISP=SHR,DSN=db2hlq.SDSNEXIT
//         DD  DISP=SHR,DSN=db2hlq.SDSNLOAD
//         DD  DISP=SHR,DSN=hlq.HLO210.SHLOLOAD
//UTPRINT  DD  SYSOUT=*
//RNPRIN01 DD  SYSOUT=*
//STPRIN01 DD  SYSOUT=*
//DSSPRINT DD  SYSOUT=*
//SYSIN    DD  UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSPRINT DD  UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
```

Example 3-9 is the JCL sample for the started task used by the Accelerator stored procedures. It includes DD statements that reference the location of the Accelerator stored procedures (in out case, IDAA51.SAQTMOD), the location of the environmental variables used by the Accelerator stored procedures (member AQTENV in library SAQTSAMP), and the PDS hlq.IDAA41.SAQTOSR containing the preprocessed Optimized Schema Representations (OSRs) for the XML System Services parser in member AQTOSR (no longer used in Accelerator 5.1).

The NUMTCB value for the WLM address space that manages the Accelerator stored procedures should be equal to `3 × <value of AQT_MAX_UNLOAD_IN_PARALLEL> + 1`. For example, if AQT_MAX_UNLOAD_IN_PARALLEL is set to 4, the NUMTCB value used for this address space should be 13.

Example 3-9 Sample Accelerator WLM managed stored procedure address space proc

```
//ssidWLMI PROC APPLENV='DSNWLM_IDAA',
//            DB2SSN=ssid,RGN=OK,NUMTCB=61
//IEFPROC EXEC PGM=DSNX9WLM,REGION=&RGN,TIME=NOLIMIT,
//            PARM='&DB2SSN,&NUMTCB,&APPLENV'
//STEPLIB  DD  DISP=SHR,DSN=CEE.SCEERUN
//         DD  DISP=SHR,DSN=db2hlq.SDSNEXIT
//         DD  DISP=SHR,DSN=db2hlq.SDSNLOAD
//         DD  DISP=SHR,DSN=db2hlq.SDSNLOD2
//         DD  DISP=SHR,DSN=hlq.IDAA51.SAQTMOD
//         DD  DISP=SHR,DSN=XML.HXML1A0.SIXMLOD1
//SYSTSPRT DD  SYSOUT=*
//CEEDUMP  DD  SYSOUT=*
//OUT1     DD  SYSOUT=*
//UTPRINT  DD  SYSOUT=*

//DSSPRINT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//AQTENV   DD  DISP=SHR,DSN=hlq.IDAA51.SAQTSAMP(AQTENV)
//* Include the following DD card when using an Accelerator prior to v5.1
//AQTOSR   DD  DISP=SHR,DSN=hlq.IDAA41.SAQTOSR(AQTOSR)
```

3.2.4 IBM z Systems Integrated Information Processor (zIIP)

An IBM z Systems Integrated Information Processor (zIIP) is an IBM specialty engine. Specialty engines are the same type of hardware as a general purpose central processor (CP) with the same characteristics and speed as full capacity CP. However, specialty engines can run only specific workloads. Because only specific workloads are run on those processors, IBM can price those workloads efficiently. Thousands of applications can share a general processor, which makes general processors cost-effective for virtualized workloads.

The primary purpose of a zIIP is to help reduce your z Systems software cost. Software charges for many mainframe products are based on CPU consumption. The more CPU used, the greater the increase in your software cost. However, by installing one or more zIIP specialty engines, some of the millions of instructions per second (MIPS), or maybe it should be billions of instructions per second (BIPS) now, that would have been used by a CP, can be redirected to a zIIP. When the software is redirected to run on a zIIP rather than CP, the processors used by that redirected software are not included in the total MSU rating and it does not affect the machine model designation. When the MSU or model does not change, the software charges do not increase.

zIIPs can be used for far more than the HLV and HLO functions of Loader. Between DB2 and z/OS, other functions exist that can also take advantage of a zIIP specialty engine. To help explain why zIIP activity can be reported even when performing non Loader processing, some of the specific workloads that might potentially be redirected to a processor repurposed as a zIIP are listed.

Note: On a sub-capacity z13 processor, zIIP specialty engines run at the same speed of a full-capacity processor.

The following DB2 for z/OS work is eligible for zIIP:

- The utility BUILD and REBUILD phases for index maintenance
- The UNLOAD phase of REORG
- Most index, COLCARD, FREQVAL, HISTOGRAM statistics collected by RUNSTATS
- Inline statistics for LOAD, REBUILD INDEX, and REORG (about 80%)
- 100% of delete processing for LOAD REPLACE with dummy input
- 100% of all prefetch (dynamic, list, and sequential)
- 100% group buffer pool (GBP) write, castout, p-lock, and notify/exit processing
- 100% index pseudo delete cleanup
- Log reads and writes
- Call, commit, and result-set processing for remote stored procedures
- XML parsing

The following non DB2 (or z/OS) activities can use a zIIP and are sometimes overlooked:

- IPSec: Encryption processing, header processing, and crypto validation (93% for bulk data movement).
- HiperSockets for Large messages.
- Simultaneous multithreading (SMT) mode processing is available only for zIIPs, implemented on a z13 and later with z/OS v2.1 and later. With SMT enabled, zIIPs can potentially have 140% or more capacity.
- zAAP on zIIP: Anything that would (or could) run on a zAAP is not eligible to run on a zIIP.
- SDSF in z/OS v2.2 is designed to offload a portion of certain kinds of processing.
- DFSMS SDM of z/OS (processing associated with IBM z/OS Global Mirror (zGM), also known as Extended Remote Copy (XRC).

- IBM SDK for z/OS, Java Technology Edition.
- z/OS XML System Services.
- System Data Mover (SDM).
- DFSORT (Sorting of fixed length rows: 10 -4 0% for utilities).
- DB2 Sort (10 - 20% potentially eligible for redirect).
- Global Mirror for z/OS (formerly Extended Remote Copy): most SDM processing.
- IBM Platform for Apache Spark.

Measurements were performed using 2 - 14 zIIPs, enabled depending on the test being performed. The measurement team does acknowledge that testing with 14 zIIPs might not be realistic and might not translate back to most customer environments. However, with 14 zIIPs, the intention was to minimize or eliminate all zIIP on CP time from becoming part of the measurements and achieve maximum throughput.

While performing measurements using Loader v2.1, the Accelerator Loader Server (referred to as HLV in this document) is responsible for extracting data from any of the allowed data sources where they reside and in memory loading, that address space has a zIIP generosity factor of 100%, attempting to allow all HLV work to execute on a zIIP. Unlike the Accelerator Loader Server (HLV), the zIIP redirect measured for the batch interface responsible for interfacing with DSNUTILB (referred to in this document as HLO) varied and was always lower, sometimes significantly, than Loader's HLV counterpart.

The Accelerator Loader Server, which performed the data extraction (HLV) and handing the extracted data off to the batch process, experienced more zIIP redirects.

Taking advantage of parallelism (a process also frequently referred to as Map Reduce Client, MRC, or just mapreduce), improves zIIP redirect. However, caution should be taken when choosing the degree of parallelism if attempting to minimize zIIP on CP time (referred to as `APPL % IIPCP` in the Service Policy Time report, which is part of the IBM RMF™ Workload Activity Report). The concept exists in HLV (not in HLO, classic Loader) where you can specify how many parallel threads you want when loading any data asset, for example VSAM. What that means is that HLV logically partitions by introspecting various VSAM metadata that exist in control blocks.

With HLV, if running a degree of parallelism of 20, there will be 40 enclave SRBs created. This means that for each parallel count value, 2 SRBs running inside the HLV address space exist. No locking occurs between any of these groups of two allowing them to run completely in parallel improving the overall performance of Loader. Within each pair of SRBs, one SRB is responsible for physical I/O while the other SRB performs data transformations and moving data into a shared memory object (shared memory objects are used when moving data between the HLV and HLO components of Loader). Because of this paired SRB per parallel stream concept, attempting to use more zIIPs than available is impossible. If the result of multiplying the parallel value being specified by 2 results in a value greater than the number of zIIPs assigned to the LPAR, RMF is more than likely going to report zIIP on CP time. This is not an issue, but rather a warning to be aware that achieving 100% zIIP on CP redirect might not be possible without considerably overloading an LPAR with zIIPs.

Note: To divide the data into logical partitions and process the partitions in parallel when processing a data source using the Data Studio Loader plug-in, click **Advanced**. Enter a Thread Count value for MapReduce (Server Parallelism Settings). The number of zIIP processors is checked at run time, and one thread is used for each processor that is discovered. The value that you specify overrides the default value (2) and the discovered value.

HLO (the batch component of Loader) does not use that pairing architecture. HLO zIIP usage is almost exclusively based on the number of parallel threads delivering data to the Accelerator. However, Loader is still switching back and forth on larger buffers of memory when converting data from the source format received from HLV to the format required by the Accelerator. The majority of HLO zIIP time is used by data conversions. HLO is somewhat limited by how fast it can process data conversions by the rate the data is excepted by Accelerator. Reaching any upper bound of zIIP processing is difficult because HLO is not doing everything on zIIP.

This explains the potential of observing zIIP on CP even with an LPAR overloaded with zIIPs (14 in our example). The 40 SRBs (2 SRBs per each of 20 parallel threads) running nearly much concurrently for HLV can easily exceed the 14 defined zIIPs. Especially Loader required both HLV and HLO together, both consuming zIIPs, even competing with each other for zIIP time, potentially driving more processing to the general purpose processors (CP).

To complete the described scenario with an MRC of 20, across the two Loader address spaces, Loader could potentially use as many as 60 enclave SRBs, all of which could be eligible for zIIP redirect.

This behavior was observed during the measurements performed for this book. Referring to Figure 3-2, the first set of bars on the left represents the CPU cost and elapsed time when running an extract to a sequential file, then using that sequential file as input to the Loader batch interface (HLO). The second set of bars on the right represents using the Accelerator Loader server (HLV). HLV did the extraction and managed moving the data to the batch interface for loading in the Accelerator. An observation was that the HLV process had a 91% decrease in elapsed time over the HLO process. HLV also experienced a 28.6% decrease in total CPU processing cost. In addition to the reduced total CPU cost, there was a 420% increase in the amount of zIIP redirect by the HLV process. These results are dramatic. In addition to reducing elapsed and CPU times by allowing Loader to handle everything from extraction to transformation to Accelerator load, the amount of additional zIIP redirect demonstrates the total overall savings is significant while reducing the complexity of moving a non-DB2 data source into the Accelerator.

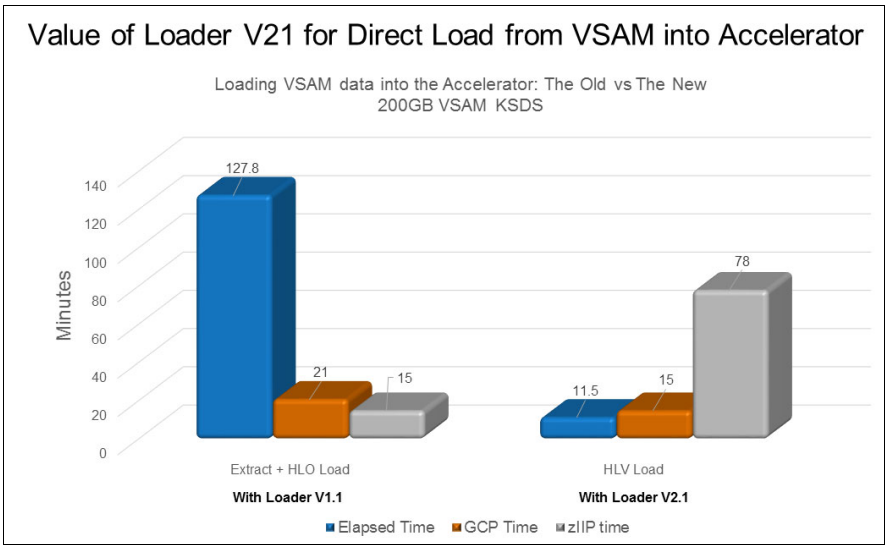


Figure 3-2 zIIP redirect comparison

These are expected zIIP redirect generosity factors (theoretical cap) for some of Loader's data sources:

- VSAM: as high as 100%
- SEQUENTIAL: mid 90%
- SMF Logstreams: as high as 100%
- DRDA: as high as 100%
- ADABAS: as low as 90%

Note: A strong suggestion is that the processing option `IIPHONORPRIORITY` specified in the IEAOPTxx member of PARMLIB never be set to `NO`. Setting this keyword to `NO` prevents zIIP processor eligible work from executing on standard general purpose processors (CP) unless resolving resource contention with non-zIIP processor eligible work is necessary. In theory, if one was to specify `"IIPHONORPRIORITY = NO"`, RMF IPPCP time might be loosely equated to wait time. IIPC time would be interpreted as wait time if `"HONORPRIORITY = YES"` is specified.

3.2.5 z Systems advantage

The IBM z Systems platform, including the z13s, z13, and the zEC12, provide function, capacity, and processing power to improve business performance and ensure future growth. Loader fully takes advantage of numerous features made available on the z Systems platform. All of the following features are used by Loader to its advantage. Some are exclusive to the z13 and a few are available on both the z13 and zEC12. Features delivered by the mainframe that could be of benefit to Loader's performance and operation are as follows:

- z13 family:
  - Simultaneous multithreading (SMT) to execute two instruction streams (or threads) on a processor core, which delivers more throughput for Linux on z Systems and zIIPs. See *z Systems Simultaneous Multithreading Revolution*, REDP-5144.
  - Single-instruction, multiple data (SIMD), a vector processing model that provides instruction level parallelism, to speed workloads such as analytics and mathematical modeling. See *SIMD Business Analytics Acceleration on z Systems*, REDP-5145.
  - Higher processor maximums: 141 CPs and 94 zIIPs.
  - Availability of up to 10 TB of memory. See *z/OS Infrastructure Optimization using Large Memory*, REDP-5146.
  - 2 GB large pages.
- zEC12 and z13 families:
  - 1 MB pageable large pages and large frame exploitation
  - IBM z Systems Integrated Information Processor (zIIP)
  - Shared Memory Communications over RDMA (SMC-R)
  - IBM z Systems Data Compression (zEDC), compression designed for high performance for use with SMF through logger, zlib, Java, BSAM/QSAM, DFSMShsm and DFSMSdss, plus others
  - Reduced storage occupancy and increased operations efficiency with IBM zEnterprise® Data Compression

3.2.6 Parallelism

Parallelism is controlled in Loader (HLO) when SYSREC is specified by partition.

Accelerator Loader supports processing multiple partitions of the same table and loading them into the Accelerator in parallel. To enable parallelism and improve performance when loading partitioned objects, you can specify multiple SYSREC data sets. The options module parameter ACCEL_LOAD_TASKS and extended syntax option ACCEL_LOAD_TASKS support this enhancement.

The ACCEL_LOAD_TASKS should be set equal to or less than the Accelerator environmental variable AQT_MAX_UNLOAD_IN_PARALLEL (Example 3-10).

Example 3-10 Parallel parameter from IDAA51B.SAQTSAMP(AQTENV)

```
AQT_MAX_UNLOAD_IN_PARALLEL=20
# Maximum number of parallel UNLOAD invocations:
# One parallel utility for each partition in a table is theoretically
# possible, but requires significant CPU availability
#
```

3.3 Scenarios

Loader's purpose is to copy data into the DB2 Analytics Accelerator. It provides multiple solutions to complete this task and from multiple sources. Regardless of your source, DB2 or non DB2, Loader provides you with the means to accomplish your data load.

Loader has two methods available that this book focuses on in this section:

- Loading directly only to the Accelerator
- Loading data in parallel to both the Accelerator and DB2

The method chosen will be driven by the data and data source. If the data is coming from a non-DB2 source, and there is no need to keep a copy of the data in DB2, why put it there. Loader allows you to load directly or only into the Accelerator. Data sources like SMF, VSAM, Oracle, and others will more than likely not require a copy in DB2. However, data that is used for analytics and for other processes, like online transaction processing (OLTP), might have to be kept in both places: a copy of the data in the Accelerator for analytics and a copy in DB2 for transactions.

Loader does provide other data load options:

- Image copy load (IDAA_LOAD_IC)
- Consistent load (IDAA_CONSISTENT_LOAD).

An image copy load, as one might expect, loads data into a single Accelerator table from a DB2 image copy. The image copy can be from an external DB2 subsystem or the same DB2 subsystem but loading into an alternative table (redirected load). The user must verify the source and target DB2 table structures are the same. JCL examples of image copy load are at the following web page:

http://www.ibm.com/support/knowledgecenter/SSKRKK_2.1.0/topics/hloucon_syntax_ICLoad_jcl.html?lang=en

Using the IDAA_CONSISTENT_LOAD keyword refreshes the data of multiple tables on the IBM DB2 Analytics Accelerator by working with the `Use FlashCopy` option. When used with this option, it does these actions:

- Creates a new IBM FlashCopy® image copy for a single table or a list of tables that are transactionally consistent.
- Loads the data from the new image copies into the Accelerator.
- Loads the data from the consistent image copies into the Accelerator.

JCL examples of loading data into the Accelerator using consistent load are at the following web page:

`http://www.ibm.com/support/knowledgecenter/SSKRKK_2.1.0/topics/hloucon_syntax_consistent_jcl.html?lang=en`

One useful feature of Loader is the simplicity and ease of getting it running for the first time. After the tool is successfully installed, the easiest way to start loading by using Loader is to modify an existing DB2 LOAD job.

To a previously created DB2 LOAD job, add a card and a statement:

- Add either an IDAA_ONLY ON *accelerator_name* or IDAA_DUAL ON *accelerator_name* Loader control card.
- Add the `//HLODUMMY DD DUMMY` statement to the batch JCL to enable the intercept and optionally add the Loader control card `ACCEL_ON_SUCCESS_ENABLE YES` to enable the newly loaded table in the accelerator.

With a minimum of only two changes, a batch DB2 LOAD job is converted to a Loader job that allows data to be added directly to an accelerator.

3.3.1 ACCEL_LOAD_TASKS

One of the Loader control cards that can affect your load processing performance is ACCEL_LOAD_TASKS. As discussed, this control card affects the amount of parallel processing, and therefore the performance impact that Loader can expect. Increasing the value on ACCEL_LOAD_TASKS can improve your throughput. During testing, the value of ACCEL_LOAD_TASKS was increased from 4 to 20 in increments of 4. For this measurement a single standalone load was performed. No other tasks were competing for resources, other than standard z/OS activity and whole performing measurements.

3.3.2 Sequential Input IDAA_ONLY and IDAA_DUAL

Most measurements were performed using the equivalent of the DB2 LOAD REPLACE function. Loader assumed the table in the Accelerator was either empty or the data in the table could be completely replaced. (Adding new rows to an existing table in the Accelerator using LOAD RESUME YES are described in 3.3.3, “Load RESUME” on page 64.) In this section, loading the Accelerator from an existing table in DB2, loading using Loader’s IDAA_DUAL, and loading using Loader’s IDAA_ONLY are discussed. Some terms used in these JCL samples are explained in “Terms used” on page 47.

The control cards, shown in Figure 3-3 on page 59, are used to produce the results labeled `DB2 LOAD followed by Accelerator load` or `Native Accelerator Load` in subsequent figures. A basic LOAD REPLACE with logging disabled was performed in each of the 100 individual partitions. The red arrow in Figure 3-3 on page 59 highlights the `ISYSREC` template state used in the corresponding `INDDN` statement. One load input data set existed for each partition.

The partition selection was managed using the template variable for the partition number (&PA). One keyword not used is KEEPDICTIONARY. Although compression is more than likely to be used in a real analytics application, disabling it for these measurements was done for simplicity. These control statements only represent one half of the load process.

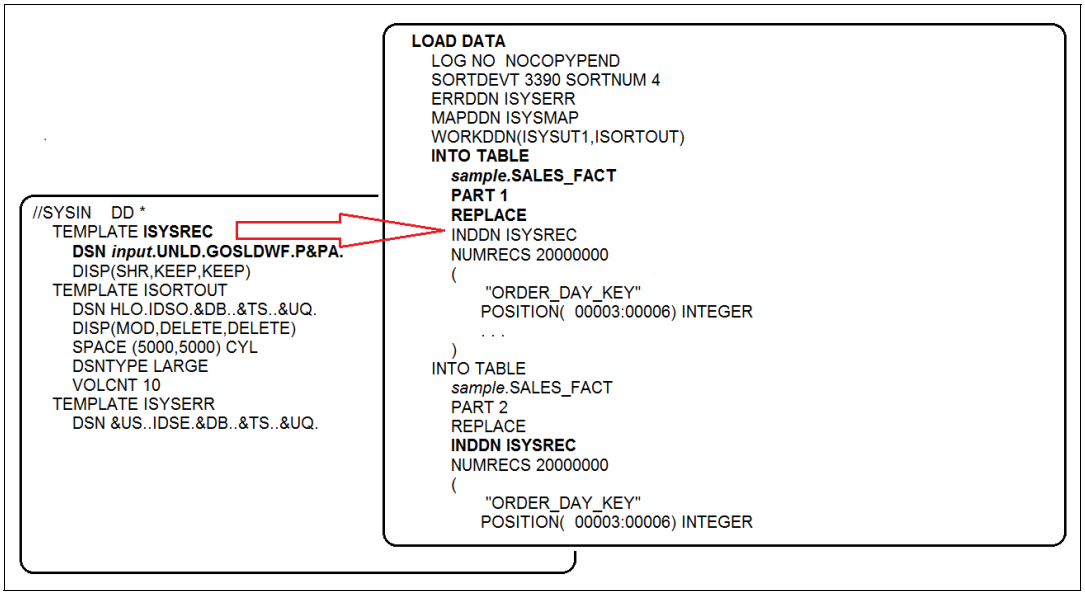


Figure 3-3 DB2 LOAD REPLACE control cards

The second job stream, Figure 3-4 on page 60, contains the JCL and control cards to invoke the Accelerator stored procedure, SYSPROC.ACCEL_LOAD_TABLES, to load the data just placed into a DB2 table into a table at the Accelerator. This is the second half of the measurement labeled DB2 LOAD followed by Accelerator load or Native Accelerator Load.

Consider the following information about Figure 3-4 on page 60:

- In this batch XML job, label “A” points to the section where the schema and table name to be loaded are specified.
- On the DSN RUN statement, the parameter LOADTABLES (label “C”) causes the program AQTSCALL (label “B”) to call the Accelerator stored procedure SYSPROC.ACCEL_LOAD_TABLES to perform the unload from DB2 and the subsequent load into the Accelerator.
- The AQTP1 statement (line 3) and AQTP2 DD (line 7) have the input control statements that name the Accelerator that will be loaded, and the lock mode that should be used during the load, respectively.
- The lock mode specified on the AQTP2 DD statement can have one of five settings:
  - TABLESET (protects all tables being loaded)
  - TABLE (protects just the current loading table)
  - PARTITIONS (protects the table-space partition)
  - NONE (no locking)
  - ROW (protects only the row)
- Other keywords can also be specified on the AQTP3 DD statement to control which partitions to load and whether only new partitions should be loaded. These are only examples and there are others. However, they pertain to the Accelerator stored procedure LOAD and do not directly affect how Loader was measured or used during the measurements.

```
//ACCLLOAD EXEC PGM=IKJEFT01,DYNAMNBR=20
//* parameter #1 for accelerator name
//AQTP1    DD *
ssid MAKO
/*
//* parameter #2 for LOCK_MODE
//AQTP2    DD *
NONE
/*
//* Parameter #3 table to be loaded - sample.SALES_FACT
//AQTP3    DD *
<?xml version="1.0" encoding="UTF-8" ?>
<aqt:tableSetForLoad
xmlns:aqt="http://www.ibm.com/xmlns/prod/dwa/2011" version="1.1">
<table schema="sample" name="SALES_FACT" forceFullRel
</aqt:tableSetForLoad>
/*
//* optional parameter for message input is omitted (NULL)
//AQTMSGIN DD  SYSOUT=*
//SYSTSPRT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSTSIN  DD  *
 DSN SYSTEM(ssid)
  RUN PROGRAM(AQTSCALL) PLAN(AQTSCALL) PARMS('LOADTABLES') LIB('IDAA51B.SAMPLOAD')
 END
/*
```

A

B

C

Figure 3-4 Accelerator stored procedure LOADTABLES

The second and third sets of measurements were performed using Loader and the control cards listed in Figure 3-5 on page 61. The second measurement used Loader's IDAA_DUAL ON load, loading both the DB2 table and Accelerator table in parallel from the same input files. The third measurement performed a load into the Accelerator using only the Loader control card IDAA_ONLY ON.

Consider the following information about Figure 3-5 on page 61:

- Label "A" highlights three Loader control cards used during measurements:
  - IDAA_DUAL ON accelerator specifies a parallel load into both DB2 and the Accelerator along with the name of the Accelerator that should be loaded.
  - ACCEL_LOAD_TASKS is the number of parallel task that can be used. This value was always set equal to the Accelerator environmental variable, `AQT_MAX_UNLOAD_IN_PARALLEL`. The ACCEL_LOAD_TASKS can have a maximum value of 20.
  - ACCEL_ON_SUCCESS_ENABLE instructs what action to take at the end of a successful load. These measurements not only used this control card but set it to YES so that the table in the accelerator was immediately available at the conclusion of the load.
- Labels "B" and "C" specify the location of the load input file, `INDDN`, and the `REPLACE` option. `INDDN` points to `ISYSREC01` at label "C" and ISYSREC02 at label "B". To improve Loader's performance and to take advantage of parallelism, each table part must use a separate input file: `ISYSREC01`, `ISYSREC02`, up to `ISYSRECxx`.
- Label "D" shows the IDAA_ONLY loader control card used when loading the Accelerator only.

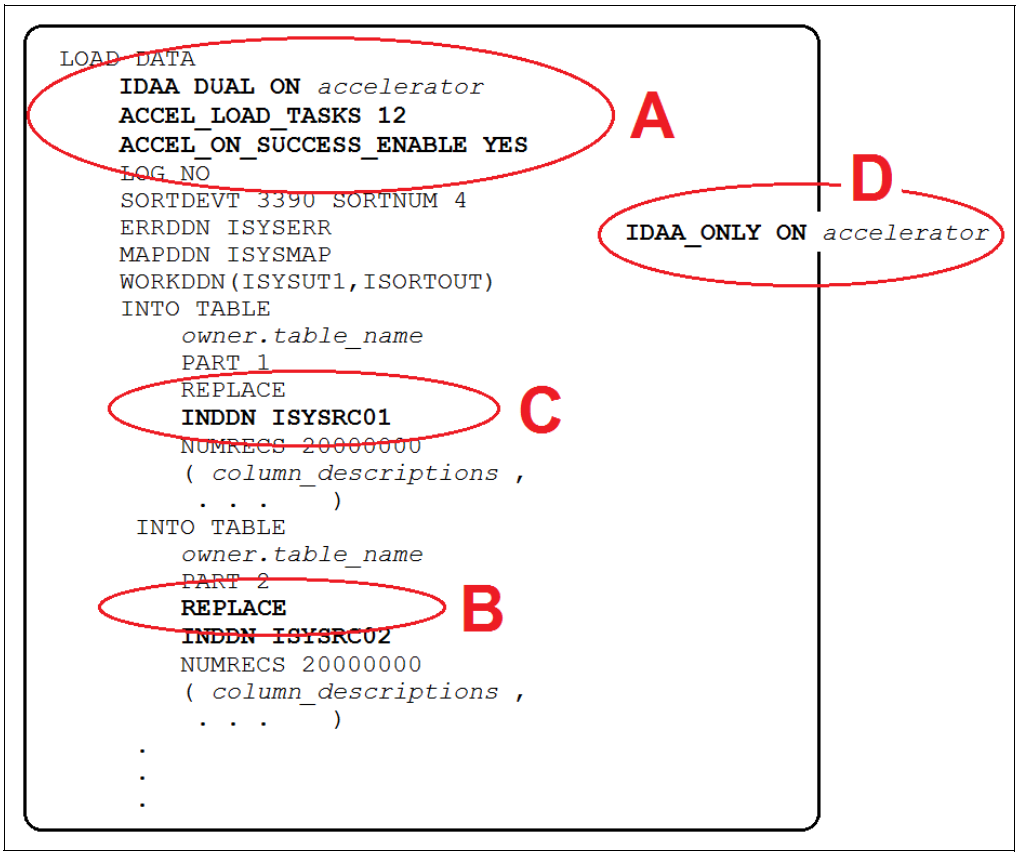


Figure 3-5 LOAD DATA Control Card Example

The results from this measurement are in the third set of columns labeled `Loader to Accelerator Only` in Figure 3-6 on page 62 through Figure 3-9 on page 64.

The results from these measurements are displayed four ways:

- Figure 3-6 on page 62 represents the total CPU consumed by the three measurements. This single CPU number includes CP, SRB, and zIIP CPU times.
- Figure 3-7 on page 63 represents the second measurement, which compares total CPU time (CP, SRB, and zIIP again) with the zIIP portion broken out.
- Figure 3-8 on page 63 represents the third way of displaying the data obtained. In this figure, the CP plus SRB time (excluding all zIIP time) is compared to the zIIP time.
- Figure 3-9 on page 64 contrasts the elapsed time of the three measurements.

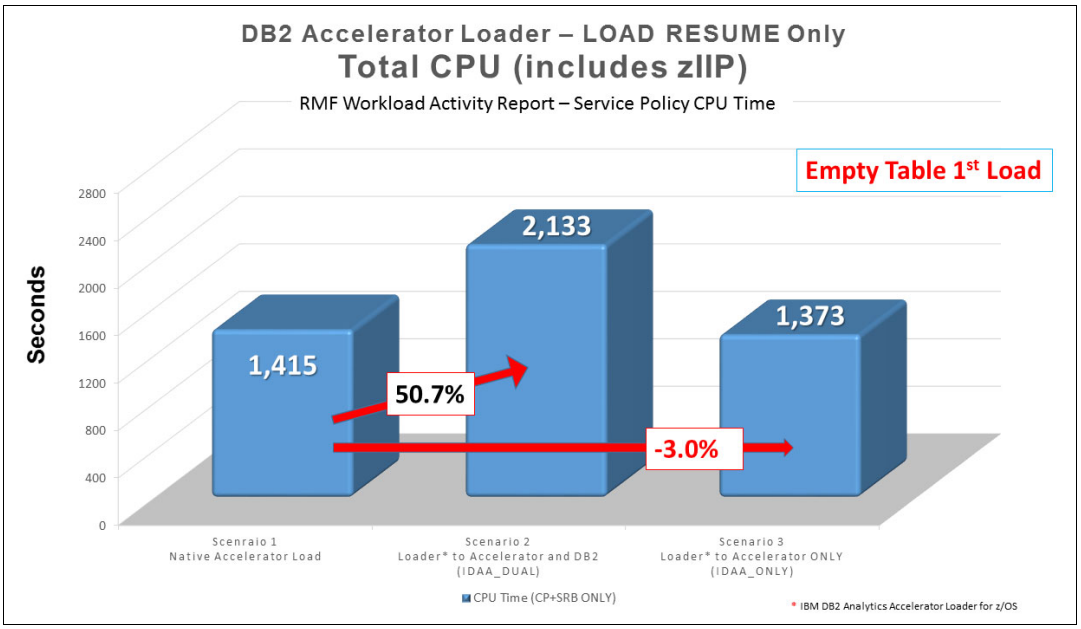


Figure 3-6 Total CPU with zIIP

In the first scenario (Figure 3-6), the cost to load 2 billion rows into the DB2 using the DB2 LOAD utility and then load that same data in the Accelerator was approximately 1,415 CPU seconds, or about 24 CPU minutes. To load those same 2 billion rows into both DB2 and the Accelerator in parallel was about 51% more expensive at 2,133 CPU seconds. Consider the following two factors when observing this increase.

- The simplicity of running only the single Loader job to load both DB2 and the Accelerator as compared to running the DB2 LOAD followed by an Accelerator load.
- The more important measurement to consider is the 28% decrease in elapsed time demonstrated in Figure 3-9 on page 64.

The numbers from this measurement become more interesting when the zIIP CPU time is examined. Remember that software charges do not apply to work that runs on a zIIP. Any time work can be redirected to a zIIP, the result can be a potential decrease in software cost.

Regarding the measurements, the CPU cost for the combined DB2 LOAD followed by an Accelerator load was again 1,415 CPU seconds. Of that, only approximately 7% was eligible for zIIP redirection. However, examining the second scenario using Loader to perform a parallel load into both DB2 and the Accelerator, a full 50% of the CPU cost was eligible for redirect to a zIIP. That is a significant portion of the total CPU cost for the load. Examining the results of loading only the Accelerator with Loader's IDAA_ONLY function, up to 78% of the total CPU is eligible for zIIP redirect. Realizing that less work is being accomplished by the last scenario, it is still a significant amount of processing being redirected to zIIP, especially when you consider those times that the data must be loaded into the Accelerator that does not have a DB2 counterpart, Oracle or DB2 LUW for example.

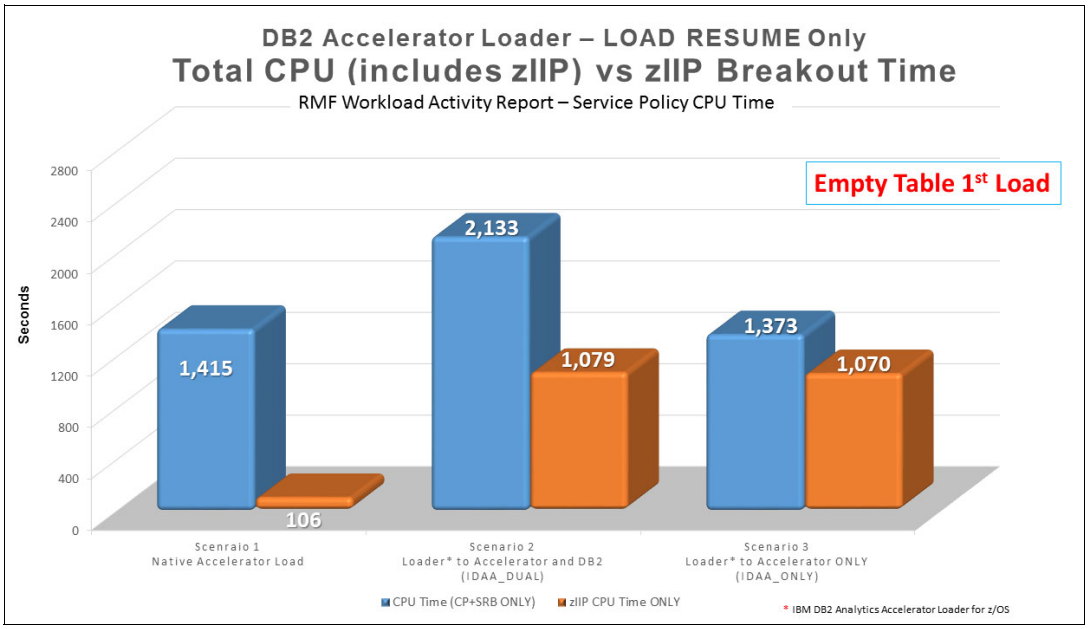


Figure 3-7 Total CPU with zIIP versus breakout time

Figure 3-8 shows the same measurement results except the CPU/SRB time minus zIIP time is being mapped against only the zIIP time.

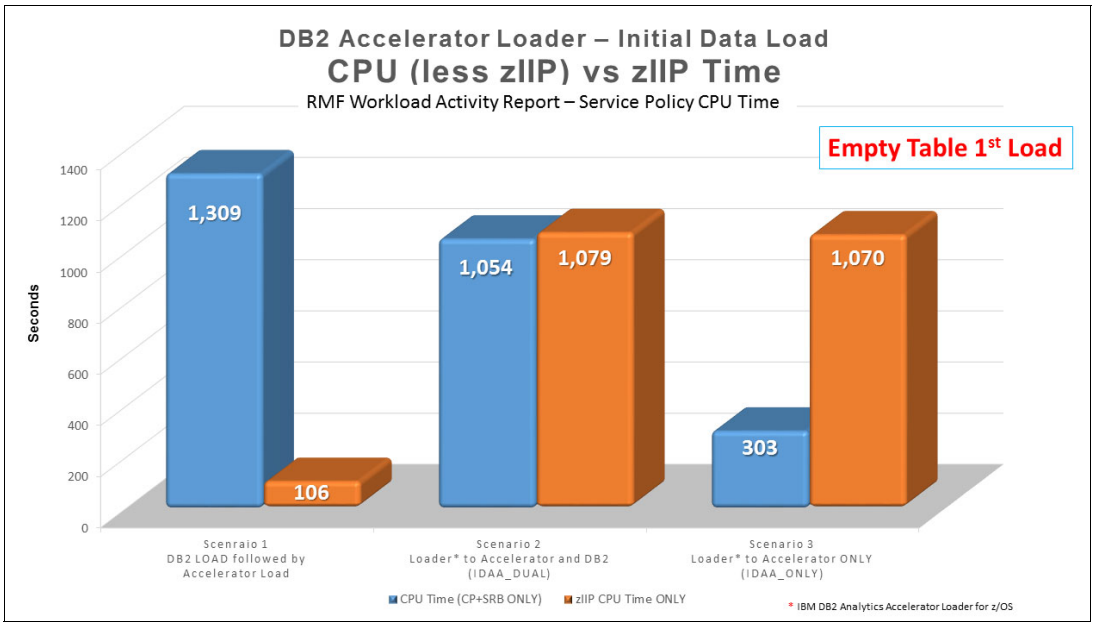


Figure 3-8 Workload activity results: CPU minus zIIP versus zIIP

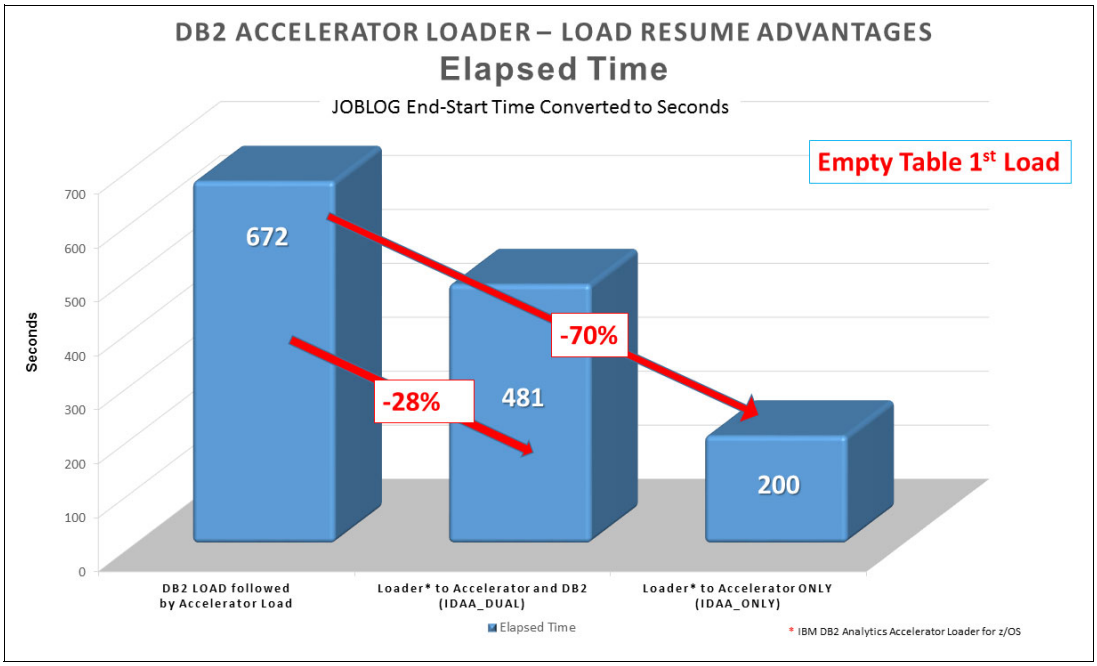


Figure 3-9 Elapsed time on initial load

3.3.3 Load RESUME

Loader's ability to perform the DB2 LOAD utility's equivalent of a LOAD RESUME YES option requires, at a minimum, DB2 Loader v2 and IBM DB2 Analytics Accelerator for z/OS V4.1.5 (PTF 5). Although DB2 can add data to an existing table that already contains data through LOAD RESUME YES, only Loader has the capability to add data to an existing table in the Accelerator using LOAD RESUME YES. The Accelerator's native load process, using the stored procedure `SYSPROC.ACCEL_LOAD_TABLES`, can replace the data in the Accelerator only with a copy of what is in DB2.

LOAD RESUME and LOAD REPLACE have been LOAD utility keywords for almost as long as DB2 has been a product. For clarity, and to help you better understand the differences between LOAD and Loader, the differences between LOAD REUME YES, LOAD RESUME NO, and LOAD REPLACE, as they apply to placing data into the Accelerator, are explained in the following list:

- LOAD REPLACE

  Replaces all data in Accelerator with a new copy for IDAA_DUAL and IDAA_ONLY. For an IDAA_ONLY load, the data in the Accelerator replaced and, if any rows exist in the table that resides in DB2, all data in that table is deleted.

- LOAD RESUME

  Either YES or NO must be specified on the LOAD RESUME option because no default value exists:

  – LOAD RESUME YES

    For IDAA_DUAL, data is added in parallel to the existing data in the DB2 table and to the existing data in the Accelerator table. None of the data that existed in either of the tables before the load is affected. For an IDAA_ONLY load, data is added to the existing table in the Accelerator. However, no action is taken against the table in DB2. The data that already exists in the DB2 table is unaffected by the LOAD RESUME YES process.

Note: If the Data Studio plug-in is used to create the LOAD RESUME YES Loader job, the drop table and create table functions are disabled and RESUME YES is mutually exclusive with the use of the ACCEL_REMOVE_AND_ADD_TABLES control card.

– LOAD RESUME NO

  Specifying RESUME NO in Loader has the same effect as specifying LOAD REPLACE. All existing data is replaced.

Load RESUME scenarios

This section describes three DB2 load scenarios.

All numbers shown reflect the sum of the elapsed time from the eight batch LOAD RESUME JOB LOAD RESUME JOBLOGs and the sum of the eight GCP times (CP+SRB) and eight IIP times from the RMF service policy report for each of these scenarios.

Scenario 1: DB2 LOAD followed by Accelerator load

Input data was 200 GB consisting of 2,000,000,000 rows; each row was 100 bytes. The columns in the row consisted of 10 integer columns, 5 decimal (19,2) columns, and 1 decimal (7,5) column. These were the steps:

1. Using the DB2 LOAD utility, a DB2 table was loaded with 200 GB.
2. The Accelerator native load stored procedure was used to load that 200 GB DB2 table from step 1 into the Accelerator.
3. Following the initial table load and simulating a 7-day maintenance week, seven subsequent DB2 `LOAD RESUME YES` jobs, each with a 10 MB input (add 100,000 rows) were executed.
4. At the completion of the initial LOAD and the seven `LOAD RESUME YES` jobs, a total of 2,000,700,000 rows or 200070 MB were loaded.

The numbers shown reflect the sum of the elapsed time from the batch LOAD RESUME job, the LOAD RESUME job log, and the RMF service policy CPU time (CP+SRB) and IIP time.

Scenario 2: Loader to Accelerator and DB2 (IDAA_DUAL)

The second scenario had these steps:

1. The 200 GB was loaded in both DB2 and the Accelerator using Loader IDAA_DUAL with REPLACE.
2. Seven more Loader IDAA_DUAL RESUME YES batch LOAD RESUME JOBs were loaded, each using a 10 MB input file.
3. At the conclusion of the 8 loads, a total of 2,000,700,000 rows or 200070 MB were loaded directly into the Accelerator and the DB2 table in parallel.

These numbers reflect the sum of the elapsed time from the batch LOAD RESUME JOB LOAD RESUME JOBLOG and the RMF service policy CPU time (CP+SRB) and IIP time.

Scenario 3: Loader to Accelerator only (IDAA_ONLY)

The third scenario had these steps:

1. Using the Loader IDAA_ONLY feature with REPLACE, the Accelerator was loaded using the same 200 GB file used in the previous scenario.
2. Seven additional Loader IDAA_ONLY RESUME YES batch LOAD RESUME JOBs, each using a 10 MB input file, were run next.
3. At the completion of the initial Loader REPLACE and the seven Loader RESUME YES LOAD RESUME JOBs, a total of 2,000,700,000 rows or 200070 MB were loaded directly into the Accelerator.

Measurements

If the same data exists in a table loaded to the Accelerator and within DB2, Loader becomes a far more efficient and less expensive alternative to an Accelerator native load using the stored procedure. In measurements performed at the IBM Poughkeepsie lab, the determination was that using Loader's LOAD RESUME YES feature successfully loaded additional data to an existing table. In loading 100,000 new rows to an existing 2 billion row table seven times to represent seven days of updates, Loader's LOAD RESUME YES used 95% less CPU and completed 81% faster keeping DB2 and the Accelerator data in sync than using the Accelerator's native load stored procedure. See Figure 3-10 and Figure 3-11 on page 67.

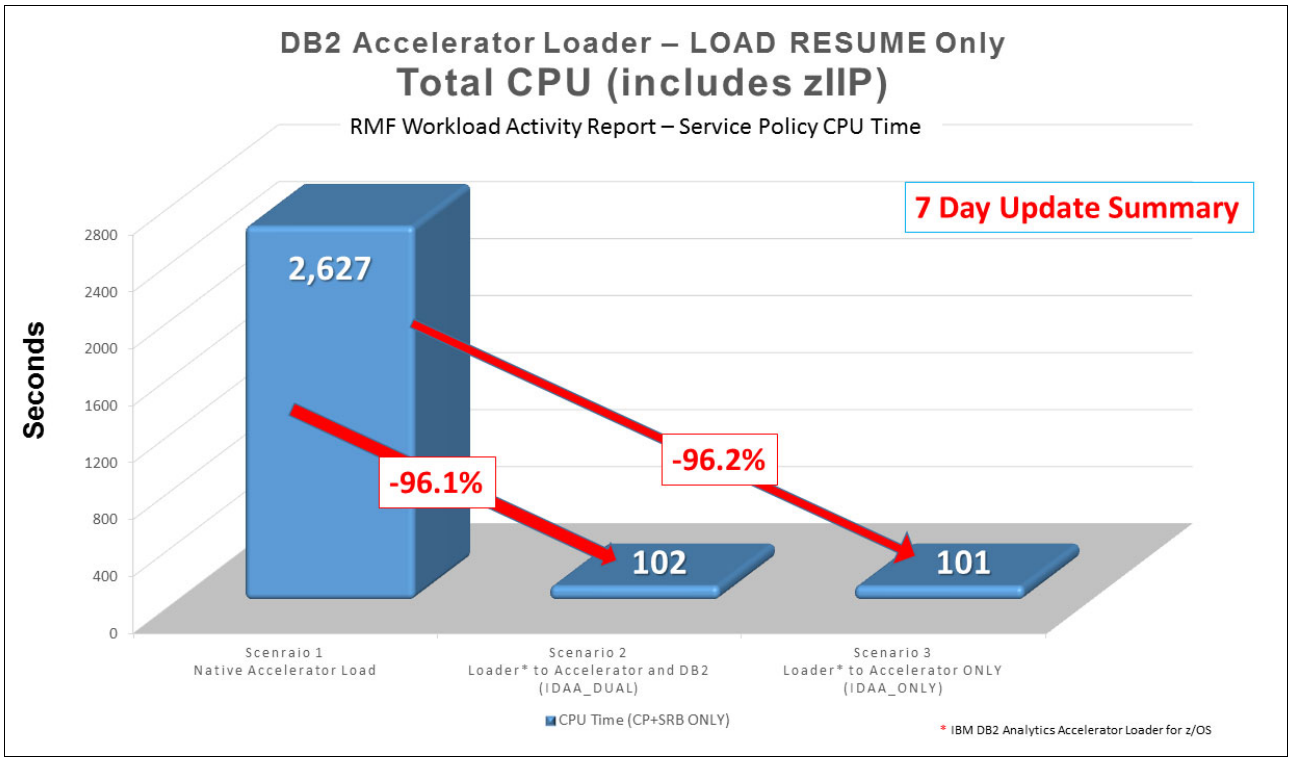


Figure 3-10 LOAD RESUME total CPU (including zIIP)

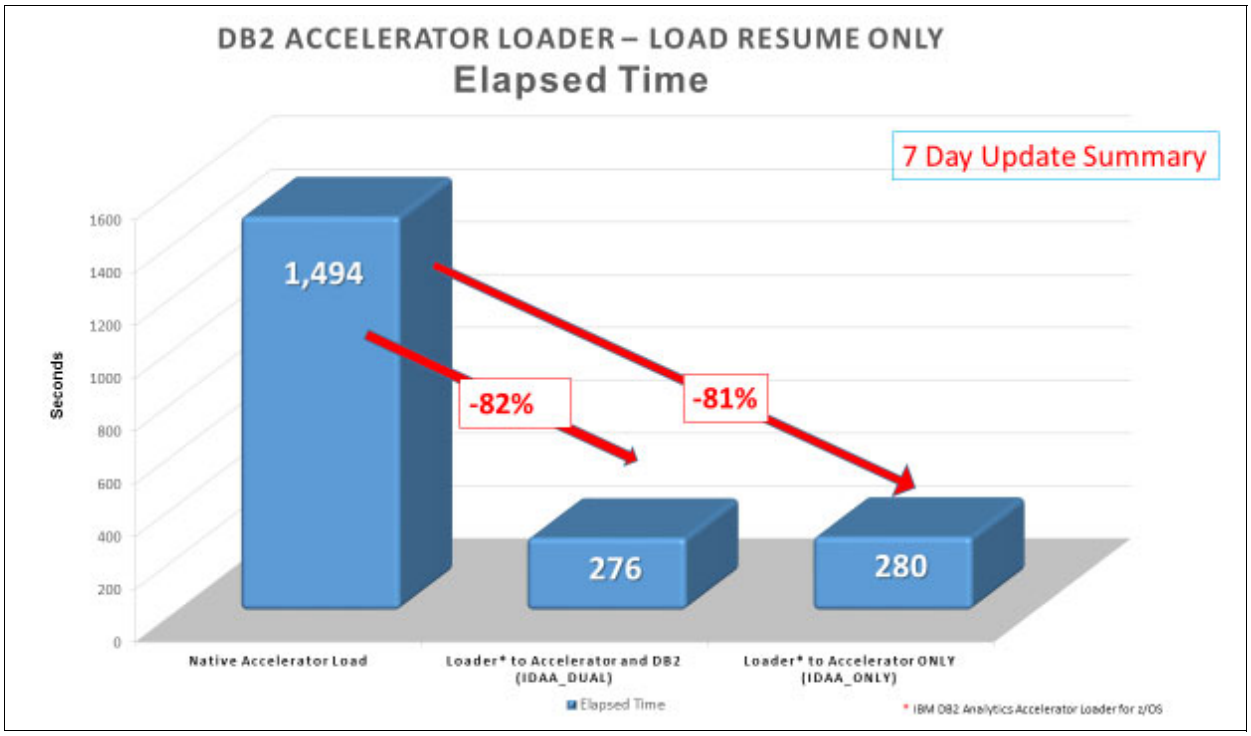


Figure 3-11 LOAD RESUME elapsed time

3.3.4 IBM DB2 Analytics Accelerator Loader image copy input

Loader's image copy load will load data for a single table into the Accelerator from the DB2 image copy defined. Example 3-11 demonstrates an image copy load. You use the IDAA_LOAD_IC keyword to load data on the IBM DB2 Analytics Accelerator from an image copy data set.

Example 3-11 Image Copy load using translate OBID feature

```
IDAA_LOAD_IC
(
    GROUP
    (
        SPACE
        (
            CREATOR 'USER01'
            NAME 'TBHLOA05_T01'
            TO_IC 'RSTEST.QA1A.DBHLOTS1.TSHLOSTA.DB2IC1'
            OBIDXLAT (
               DBID '863,868'
               PSID '2,2'
               OBID '3,3'
                     )
        )
     )
     ACCELNAME QA1AACC1
     PARALLEL '0,1'
     LOG_COPY_PREFERENCE R1R2A1A2
     USER_INDICATOR HLO
)
/*
//
```

An IDAA_CONSISTENT_LOAD is often included in discussions about Loader's use of image copies. However, using IDAA_CONSISTENT_LOAD specifies the options for loading data for multiple tables into the Accelerator from a cataloged DB2 image copy; image copy specifies options for loading data for a single table into the Accelerator from a user-defined DB2 image copy.

3.3.5 VSAM

This section describes loading data from Virtual Storage Access Method (VSAM) data sets into DB2 and then into the Accelerator.

Table 3-2 provides a description of the environment used for the VSAM measurements.

Table 3-2 Environment used for the VSAM load

Processor	z13 Model 739: z/OS LPAR with 3 dedicated CPs, 14 dedicated zIIPs
Memory	65 GB
z/OS	V2R2
DB2	V11
Accelerator DB2 SPs	V51 Beta
Accelerator hardware	N3001-010 (Mako full-rack)
Accelerator software	V51 Beta with NPS 7205
Loader V2.1	Pre-general availability (GA): Code Drop 12
Loader V2.1 key parameters	ACCEL_LOAD_TASKS = 16; MR threads = ENABLED

Measurements were performed for VSAM in three runs:

1. Using a more traditional extract method, data was unloaded form the VSAM KSDS, loaded into DB2, and then loaded into the Accelerator.
2. Again using a traditional extraction process, but this time Loader batch function (HLO) was used to load the data into the Accelerator.
3. Finally, a load was performed using the Accelerator Loader Server (HLV) to map the VSAM KSDS directly to a DB2 table, select the data from the table using SQL, and then pass the data to a batch program that loads the Accelerator.

These runs provided start-to-finish measurements from the start of the COBOL extraction program through DB2 LOAD and then through Accelerator.

As shown in Figure 3-12, a significant decrease in CPU time and elapsed time was measured across the three runs.

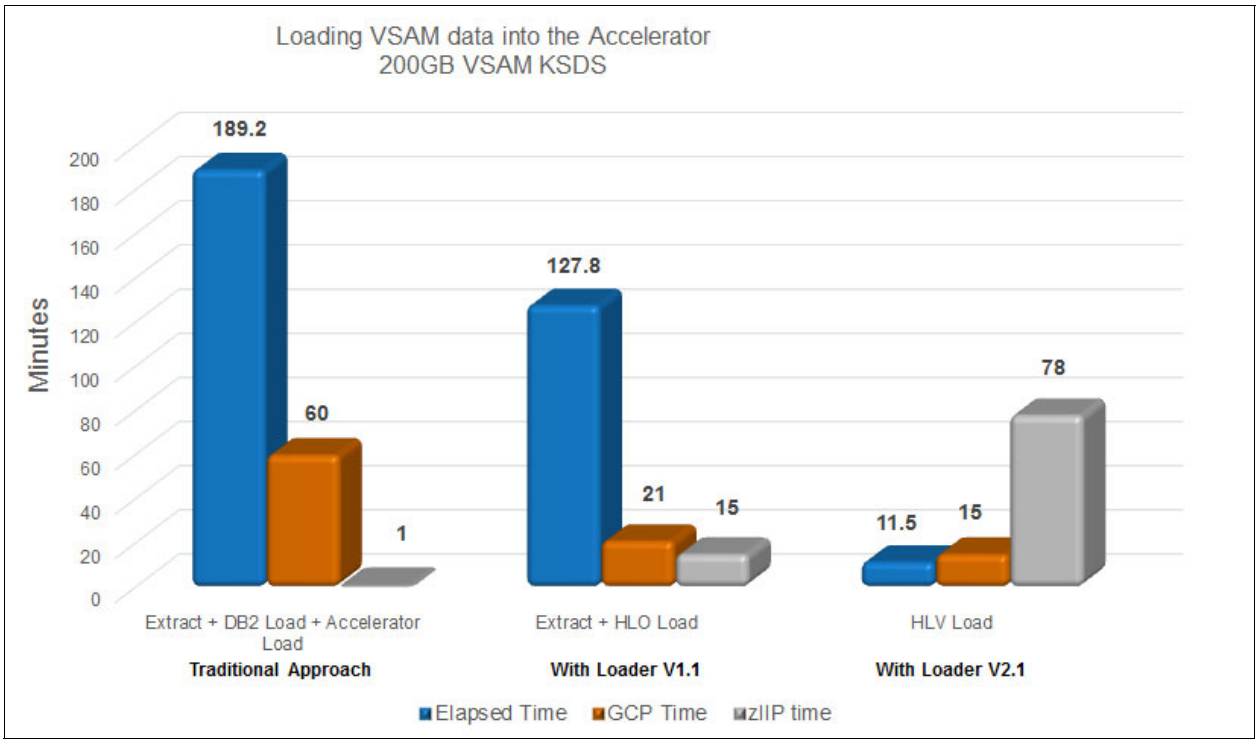


Figure 3-12 Value of Loader V21 for Direct Load from VSAM into Accelerator

The VSAM measurements were performed using an HLV server to reference the VSAM data sources. The following paragraphs and figures describe how the process was performed.

1. Using the Accelerator Loader perspective from Data Studio, shown in Figure 3-13 on page 70, expand **SQL** → **Data**, and then click the server that will be used to create a virtual table. In this example, the server is **HLVS**.
2. Right-click **Virtual Tables**, highlighted in Figure 3-13 on page 70. In the pop-up window, click on **Create Virtual Table**.

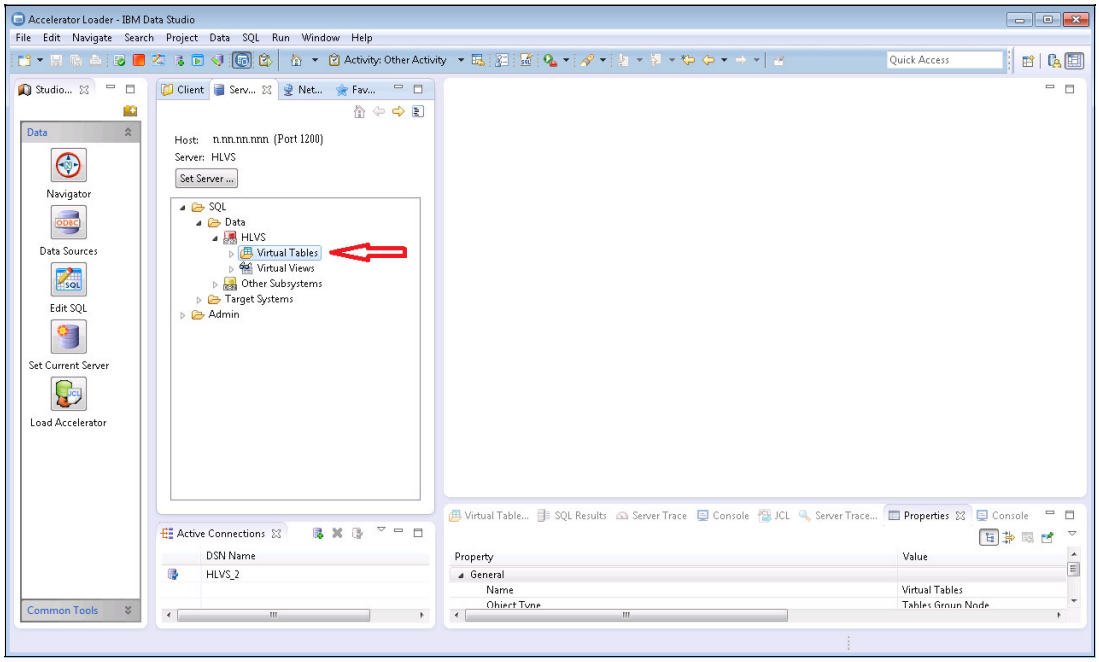


Figure 3-13 Accelerator Loader Data Studio entry panel

3. A list of wizards is displayed. Select a wizard (in this scenario, select **VSAM**, as highlighted in Figure 3-14) and then click **Next**.

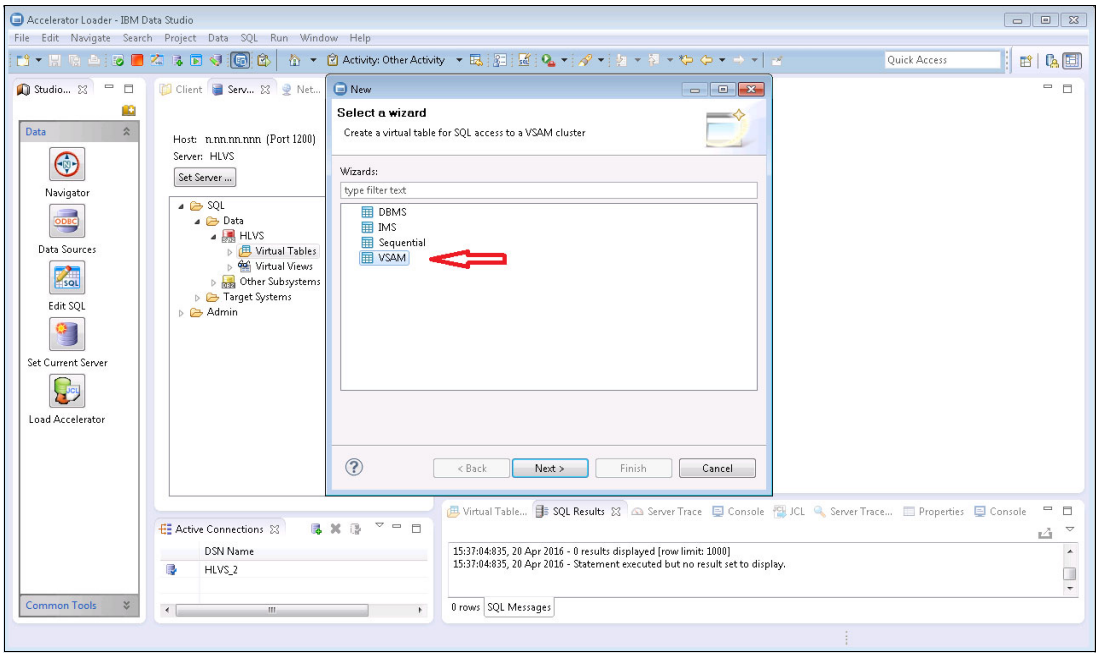


Figure 3-14 VSAM wizard starting point

4. The dialog window for defining a new VSAM virtual table opens (Figure 3-15). Provide the following information and then click **Next**:
   - In the Name field, provide a name to use for the virtual table.
   - The Target field indicates the name of the data set that will contain the metadata for the new created object.
   - (Optional) In the Description field, you may add a meaningful description.
   - The Current Server field lists the current server being accessed. If you want to change this, click **Set Server**.

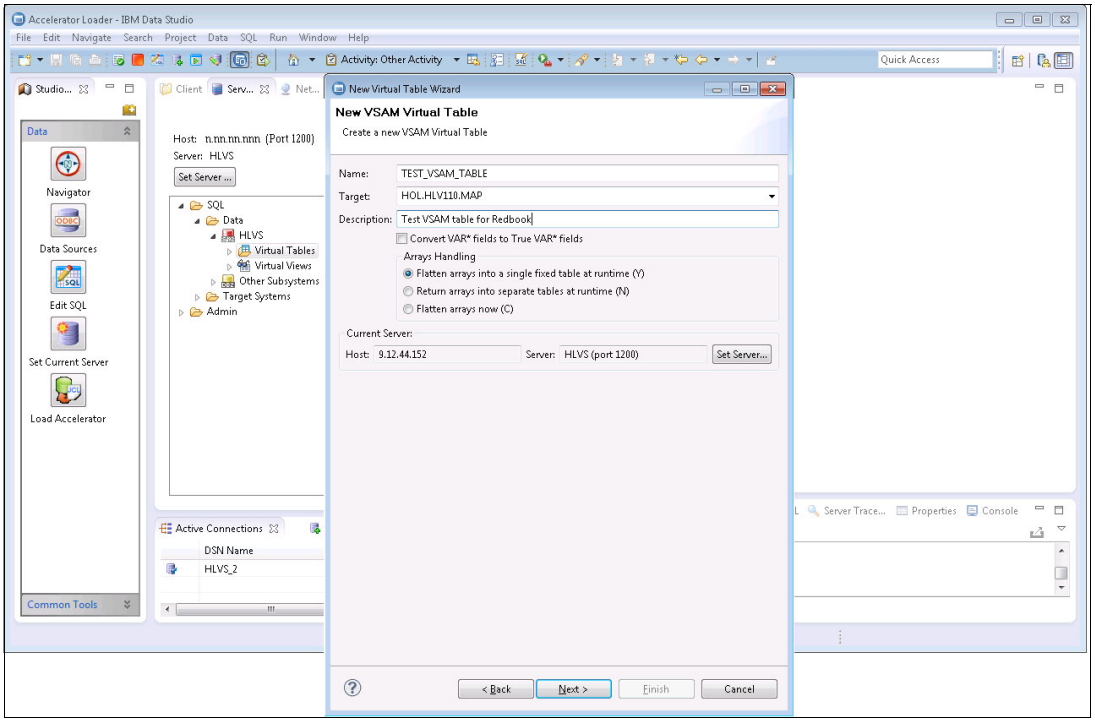


Figure 3-15 Create a VSAM virtual table

Figure 3-16 and Figure 3-17 point to the copybook specifications. The copybook is used by HLV to map the data in the VSAM KSDS to the Accelerator table. Figure 3-16 shows the correct copybook selection. This copybook must match the VSAM file exactly or the mapping process between the VSAM file and the Accelerator table will not work.

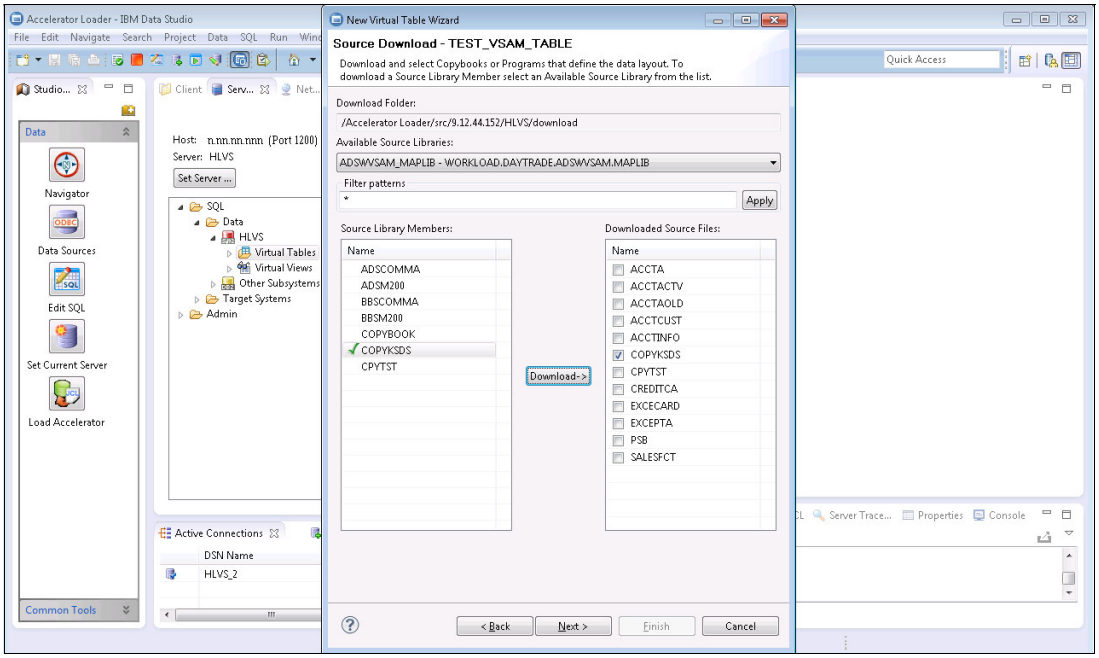


Figure 3-16 Selecting a copybook

Figure 3-17 shows the contents of the copybook, the COBOL code that will be used in the data mapping.

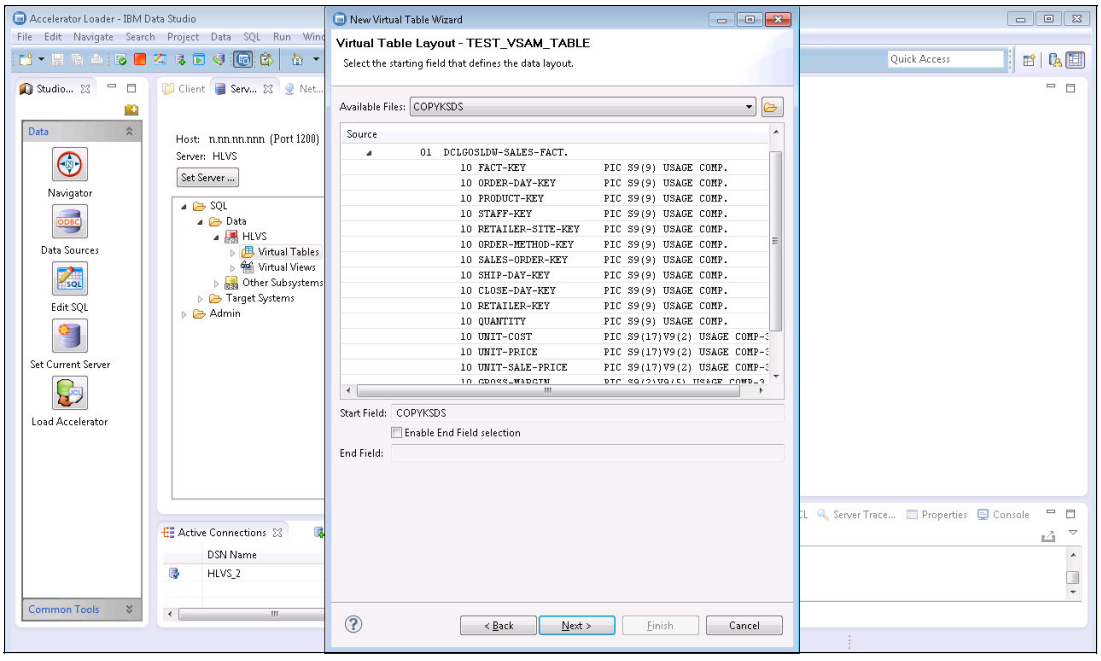


Figure 3-17 Contents of the copybook

This final step involves Loader extracting and loading the Accelerator all in a single combined task. To generate the batch job stream to perform this function, complete the following steps if you use the Accelerator Loader studio plug-in for Data Studio:

1. Click on **Generate JCL to load Accelerator** (labeled as “A” in Figure 3-18).
2. A dialog window opens (labeled “B”). The source subsystem is selected and an SQL query against the virtual table is displayed. Click **Validate** to verify that the virtual table can be accessed on the specified server. When you are satisfied that all is accurate, click **Next**.

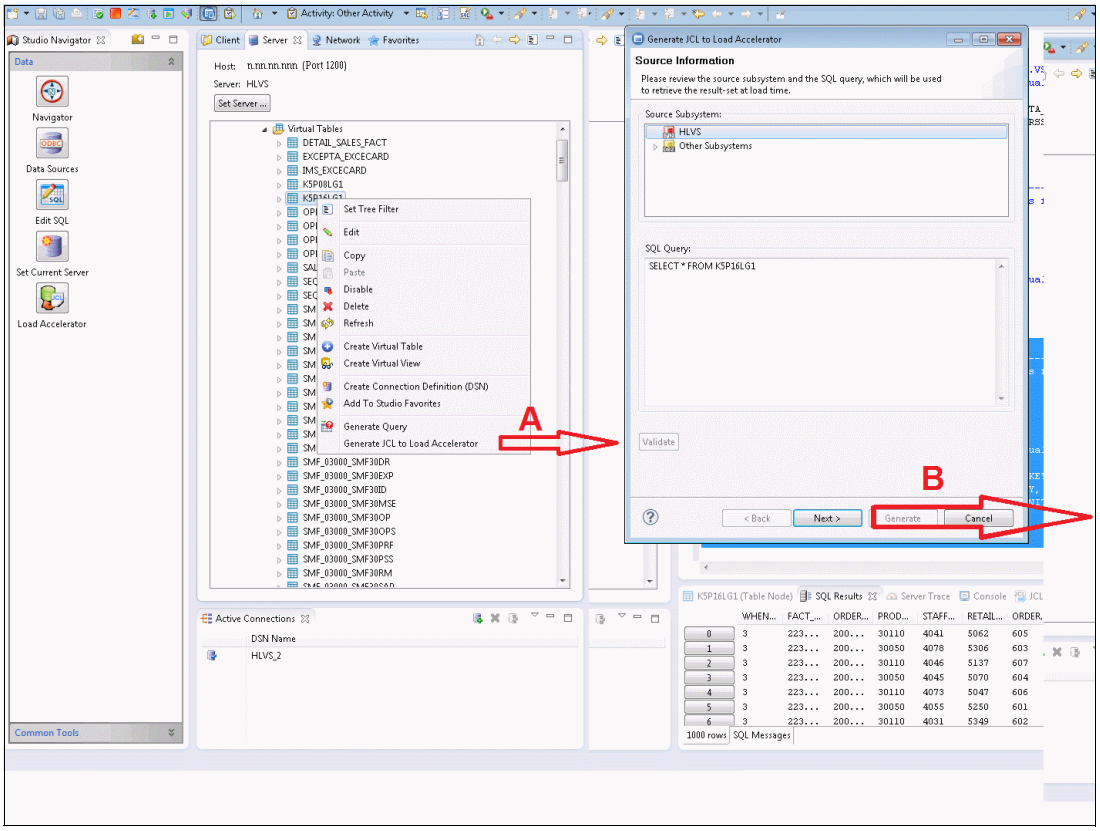


Figure 3-18 Generating the Load Accelerator JCL

3. The next dialog window opens, which shows the specified target table (Figure 3-19 on page 74). This table will be loaded in the Accelerator. In the panel labeled “C” in the figure, verify that the correct DB2 subsystem and accelerator are specified, enter the authorization ID of the table creator, and the table name. Although you may enter the database and table space information, if you do not supply them, they will be implicitly created.

4. (Optional) Enable parallelism by selecting the **Enable** check the box in the Parallelism check box and specify a degree of parallelism. If you select Enable, you must also indicate a degree of parallelism in the range of 2 - 20, otherwise, the Next button is not activated. The default value is 1 (no parallelism will be used).
5. Click **Next**.

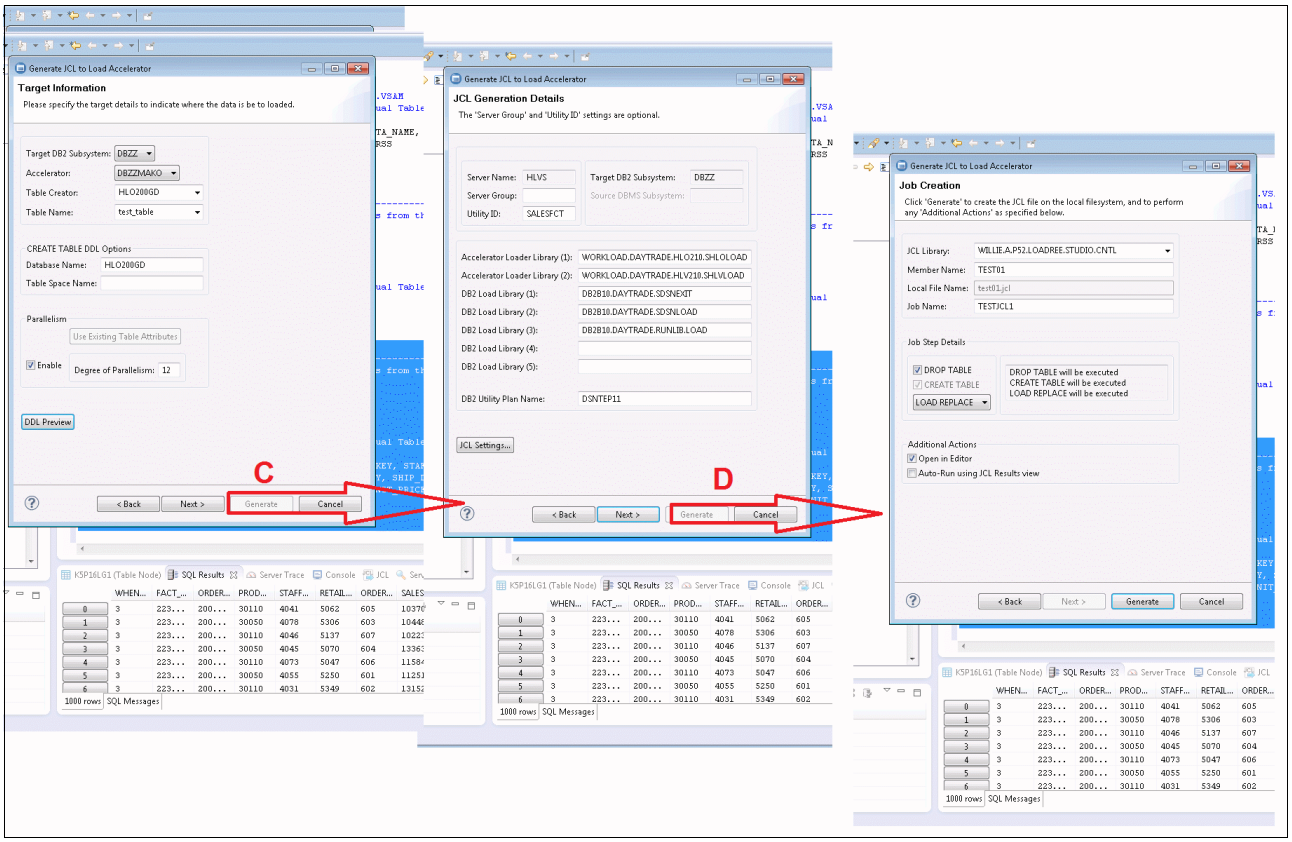


Figure 3-19 Generating the Load Accelerator JCL

6. The dialog labeled “D” opens (Figure 3-19). It specifies data sets used by the job stream along with a utility ID. If you click **JCL Settings** to allow values for job name suffix, job class, message class, regions size, and temporary disk name (like SYSDA, the default). Click **Next**.
7. The next dialog is for generating the JCL. You can specify the PDS name that will hold the JCL that is about to be generated, the PDS member name for the JCL, and the job name to be used when generating the JCL. You may also specify whether the target table should be created by the Loader by selecting the appropriate check boxes. LOAD REPLACE is the default. Although this value can be changed to LOAD RESUME, if RESUME is used, the ability to drop and create the target table is disabled. Figure 3-20 on page 75 is an example of specifying RESUME rather than the REPLACE option when generating the JCL.

8. After you complete the fields adequately, the Generate button becomes activated so you can click it and the JCL can be created.

   If **Auto-Run** has not been enabled, switch over to the TSO session, locate the appropriate PDS, and submit the job stream just created.

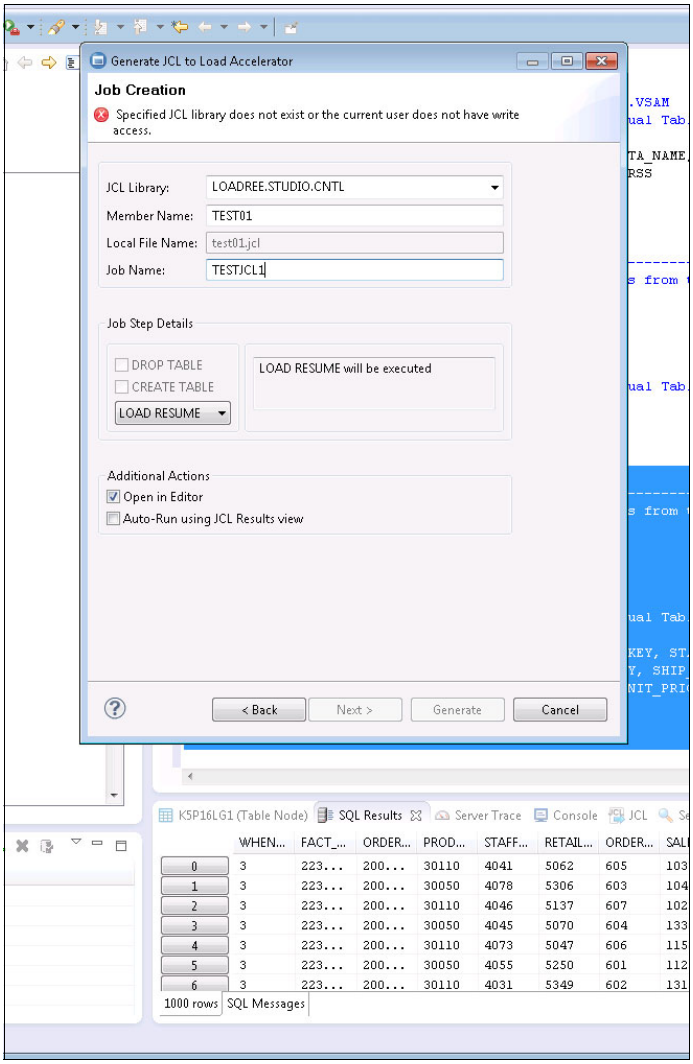


Figure 3-20 Alternate job creation using RESUME

9. After the virtual table is created and associated to the Accelerator table, a job stream can be created to actually complete the load operation. Right-click on the virtual table representing the VSAM file just created.

Measurements

The preceding steps are used for the VSAM measurements in the following sections:

- "Traditional extract measurement" on page 76
- "Extract with batch Loader (HLO) measurement" on page 77
- "Accelerator Loader Server (HLV)" on page 77

Figure 3-21 on page 76 is referred to in each section.

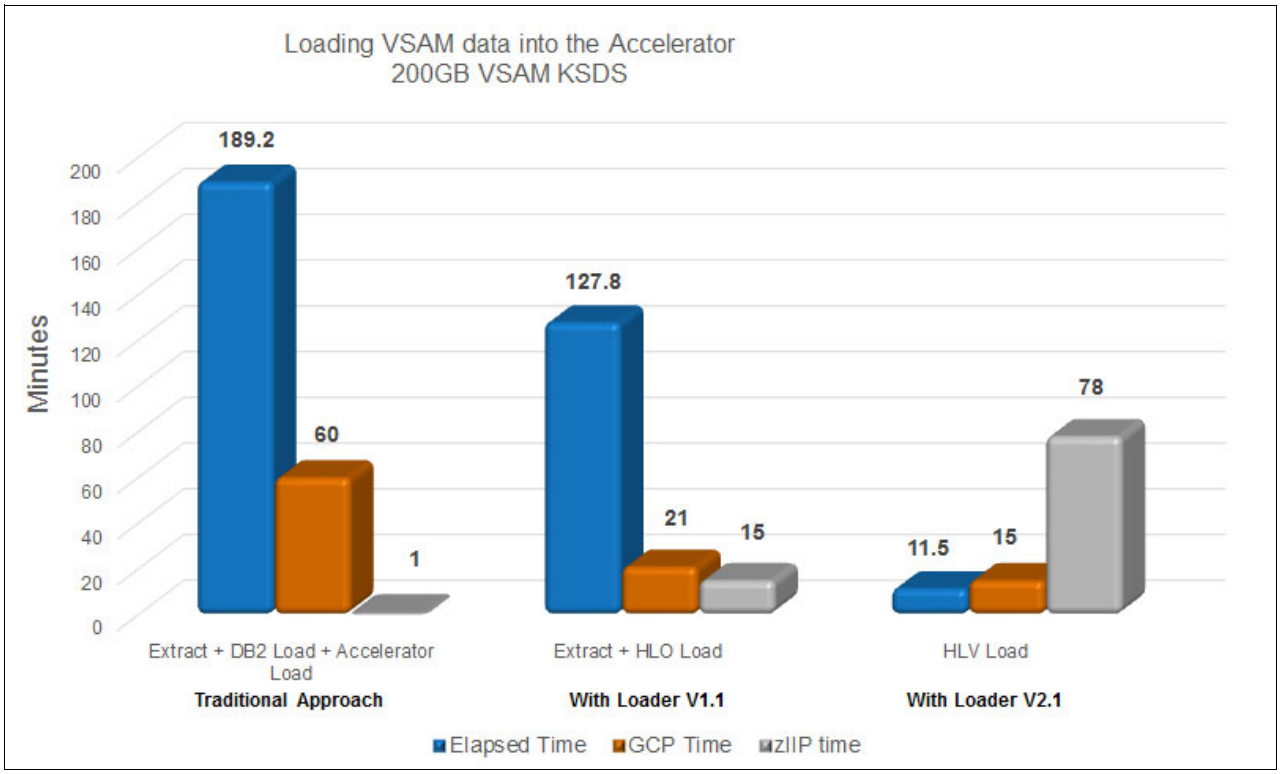


Figure 3-21 Value of Loader V21 for direct load from VSAM into Accelerator

Traditional extract measurement

The first set of bars in Figure 3-21 represent what was considered a traditional approach without the benefit of Loader. Data was unloaded from the VSAM KSDS to a sequential file using a COBOL application.

After a sequential file was created by the COBOL program, that file was used as input to the DB2 11 LOAD utility to load a DB2 table defined to “match” the VSAM KSDS.

Finally, the DB2 table data was loaded into the Accelerator using the Accelerator stored procedure `SYSPROC.ACCEL_LOAD_TABLES`. For the 200 GB VSAM KSDS used, this three-step process took over 3 hours to complete, and having to extract the data to a sequential file added a layer of complexity. Also for this measurement, a COBOL program had to be created to unload the VSAM KSDS to a flat file. The assumption was that using a program written specifically to unload the VSAM KSDS would be more representative of a real world scenario.

The three-step process also used a fair amount of CPU time (60 minutes). However, of that 60 minutes of CPU, only 1 minute was redirected to zIIPs. None of the three processes, required to get the data from VSAM to the Accelerator, took advantage of zIIPs.

Extract with batch Loader (HLO) measurement

The second measurement, represented by the second set of bars in Figure 3-21 on page 76, reduced to two steps the processing required to load the VSAM data into the Accelerator by removing the need to run the DB2 11 LOAD utility. The data was again extracted from the VSAM KSDS using a COBOL program to an external file. That extracted file was then loaded into the Accelerator using only the HLO portion of Loader. The combined two steps to extract the data and load it into the Accelerator still took more than 2 hours to complete using 21 minutes of CPU time. However, because Loader is designed to take advantage of zIIPs, the zIIP redirect portion was 15 minutes. That means that of the total CP + zIIP time of 36 minutes, the zIIP redirect accounted for an appreciable 15 minutes or 42% of the total CPU consumed.

The COBOL program used to extract the data from the VSAM file to a sequential file for this measurement was the same COBOL program used in “Traditional extract measurement” on page 76.

Accelerator Loader Server (HLV)

The third measurement took advantage of Loader v2.1 Accelerator Loader Server (HLV) performing a direct load from the VSAM KSDS directly into the Accelerator. No extraction step needed to be performed. HLV was responsible for mapping the data in the VSAM KSDS to the table in the Accelerator. Of the total CPU consumed, 15 CP plus 78 zIIP, approximately 82% percent was redirected to zIIP processors. This significant redirect is a direct result of the zIIP usage of HLV. This also resulted in an elapsed time drop from 189 minutes for the three step process to only 11.5 minutes for the one-step direct load using HLV.

3.4 System Management Facility (SMF)

Loader has the inherent capability to load SMF records from an SMF dump file directly into the Accelerator. A set of virtual tables representing a subset of SMF records is provided in Loader so creating a virtual table is unnecessary. However, associating an SMF file to the Loader process is still necessary.

Figure 3-22 on page 78 shows a sample list of the SMF virtual tables provided with Loader. SMF processing must first be enabled in Loader before any of the virtual SMF tables are available for display. To enable SMF virtual table processing, modify the SEFVTBEVENTS parameter to YES in data set `hlq.SHLVEXEC`, PDS member `hlvidIN00`:

```
MODIFY PARM NAME(SEFVTBEVENTS) VALUE(YES)
```

The parameter is invoked by the HLS process when the started task begins. The IN00 file is invoked in the procedure by the following DD statement:

```
//SYSEXEC  DD  DISP=SHR,DSN=&hlq.EXECFB in the member PROCLIB(HLV1PROC)
```

The virtual tables provided by Loader are by SMF record type and subtype. There is one table per SMF record type and subtype. Combining the information from multiple SMF virtual tables into a single source using Loader’s virtual views might be necessary in order to complete processing of the SMF data.

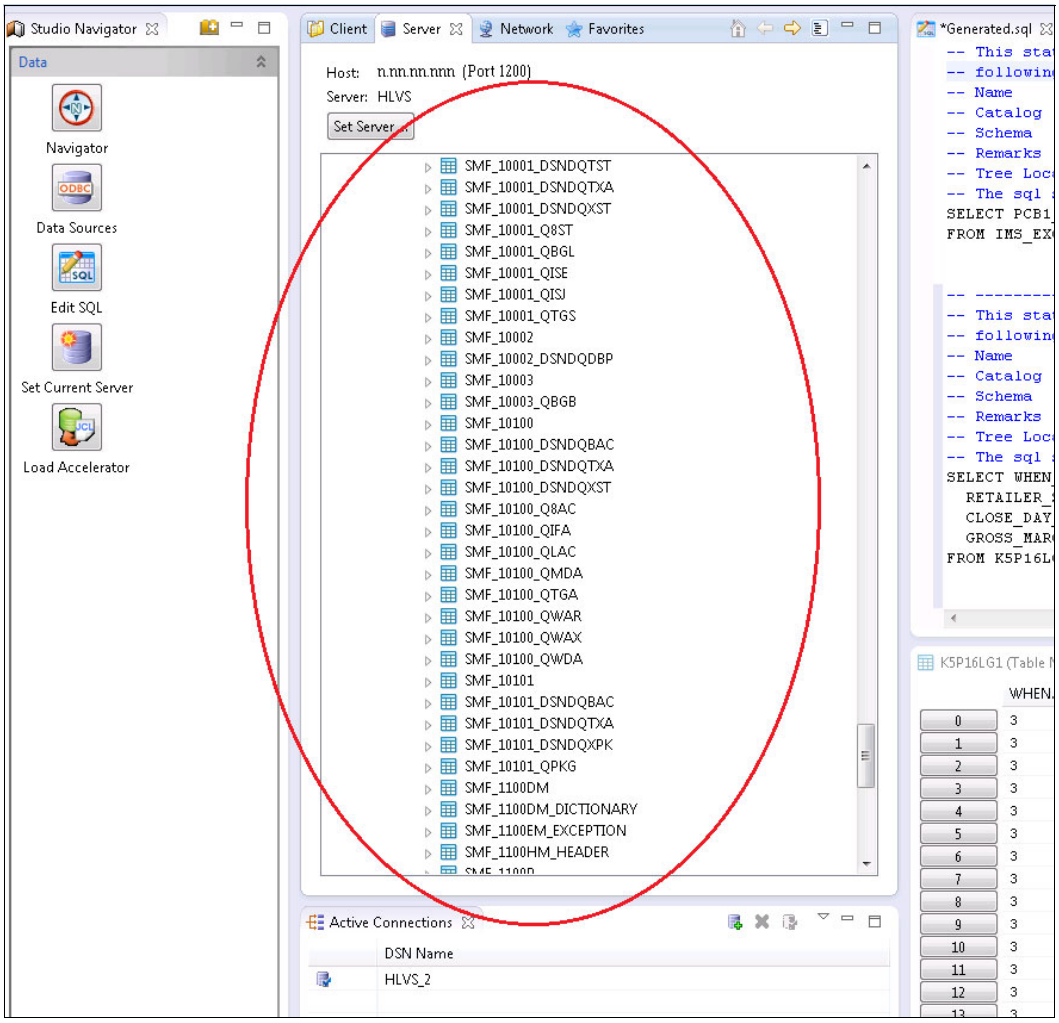


Figure 3-22 SMF virtual table example

The Loader SMF mapping data set must also be added to the HLV started task procedure before the SMF virtual tables can be used. If no SMF virtual tables are presented in the Accelerator Loader Studio plug-in, then this data set was not added to the procedure. Add `hlq.SMFMAP` to the `HLVMAPP DD` concatenation.

SMF virtual table processing must also be enabled before it can be used. Using the **B** or **E** line command, enable the SMF member HLVSMFT2 that is in the Event Facility (SEF) Event Procedure List (Figure 3-23).

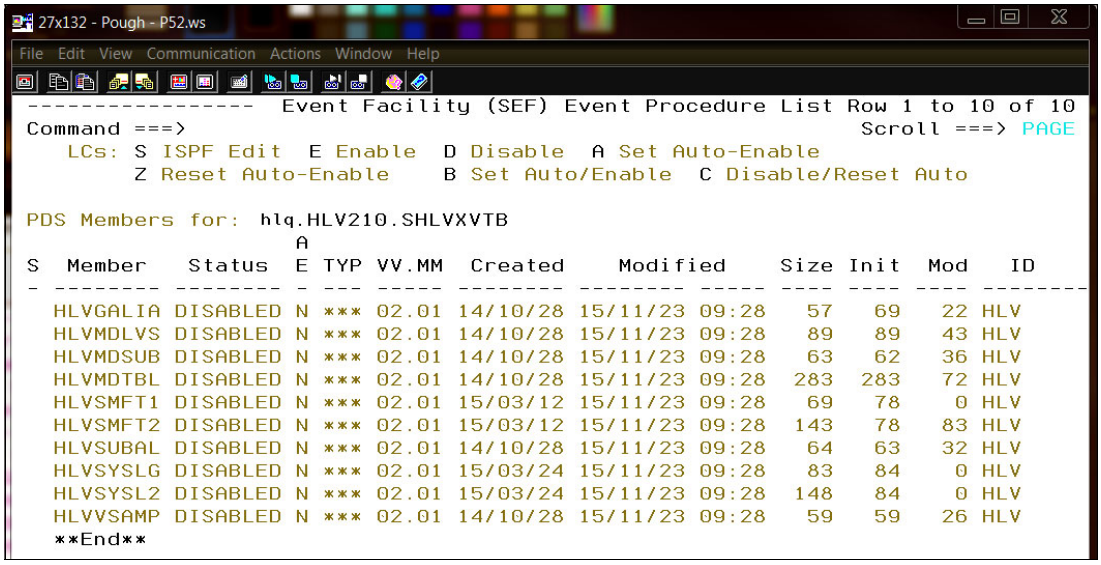


Figure 3-23 SMF_Event_Facility_menu.png

The Event Facility (SEF) Event Procedure List is reached by using the following steps:

1. Open the Loader ISPF interface. This might be an option on an ISPF menu panel, or as in this case, invoked directly as a command from ISPF option 6 (Figure 3-24).

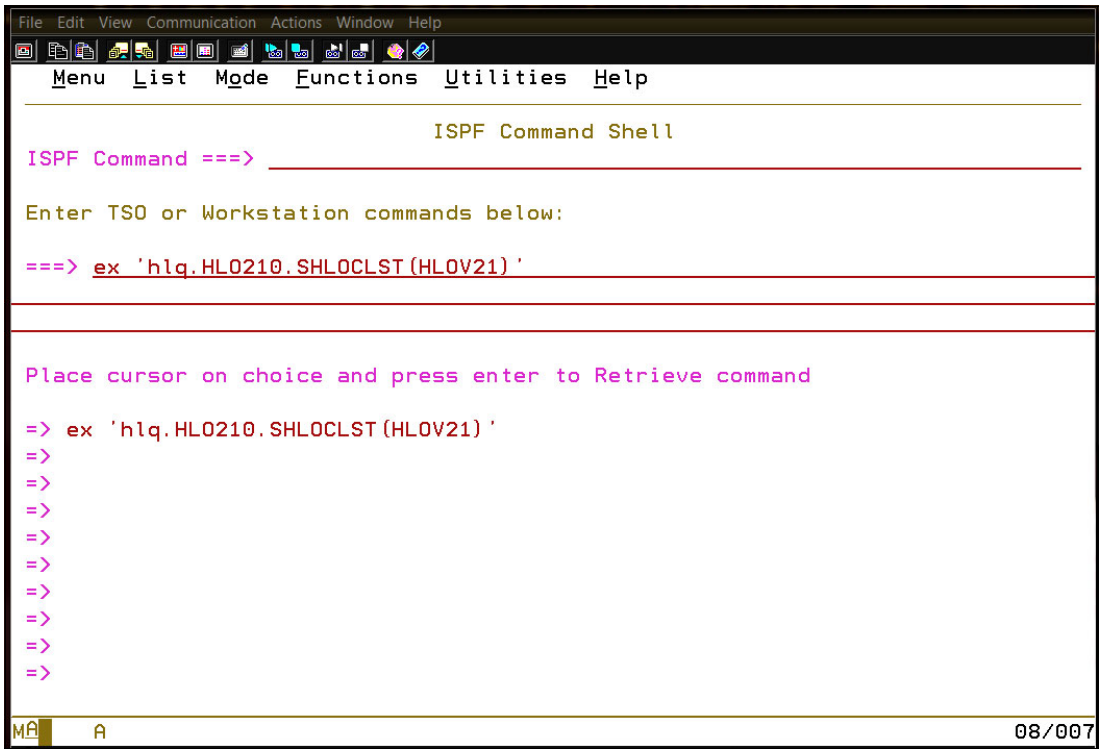


Figure 3-24 ISPF Option 6 Command Shell

2. For the SMF setup, selecting option **1, Server administration** (Figure 3-25), will invoke the three server admin functions (Figure 3-26). However, *before* selecting option 1, complete these two fields on the right side:
   - DB2 SSID: Provide the name of the DB2 for z/OS subsystem being used.
   - Server ID: Provide the name of Loader's HLS server.

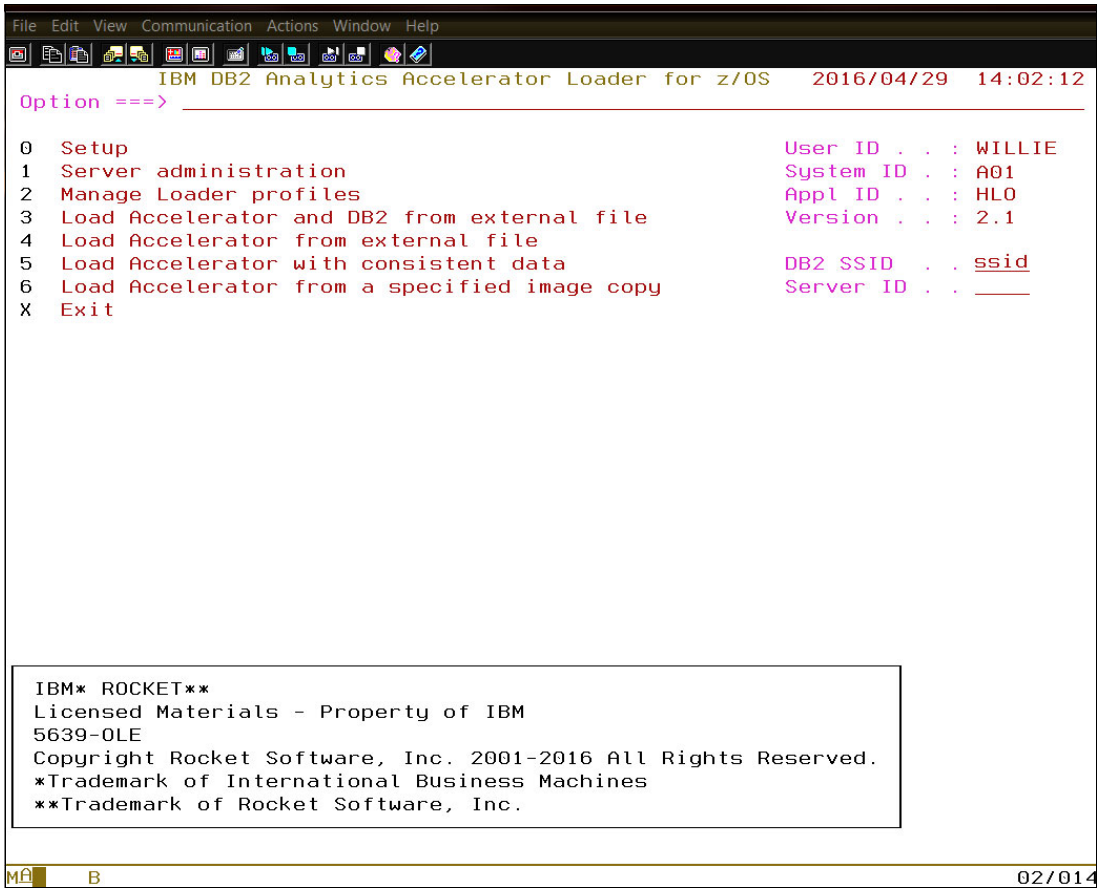


Figure 3-25 ISPF Loader Admin Main Menu

3. From the Administer Accelerator Loader Server menu (Figure 3-26), server traces can be displayed, the server can be configured, or rules can be managed. To complete the SMF setup, select option **3, Manage rules**.

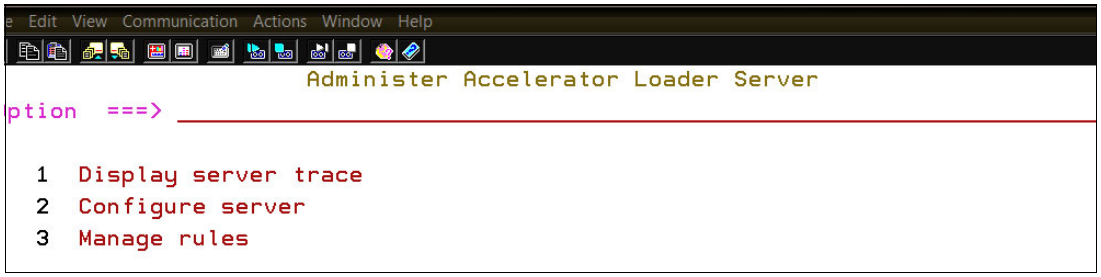


Figure 3-26 Server admin menu

Loader's ISPF Event Facility (SEF) Control menu opens (Figure 3-27).

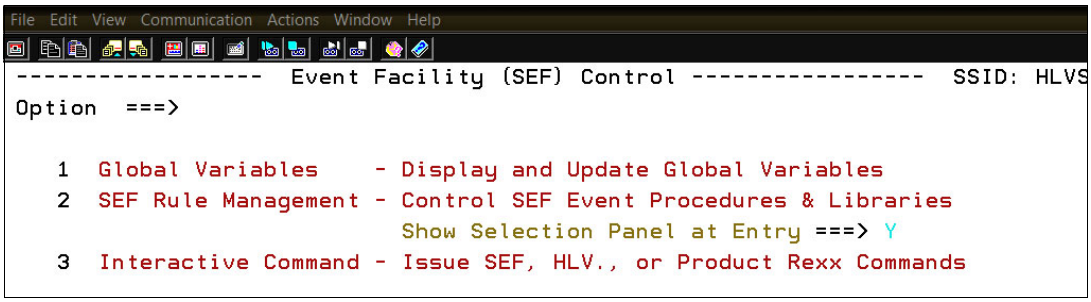


Figure 3-27 Loader Event Facility Control menu

4. Using the configuration list shown Figure 3-28, chose **VTB** (virtual table rules) to manage the rules for the SMF virtual tables.

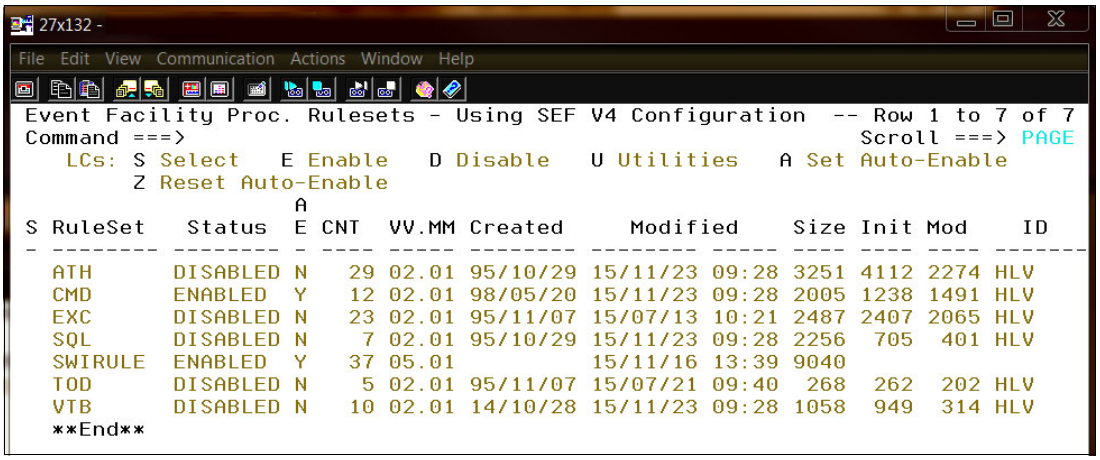


Figure 3-28 SEF Rules Management

Finally, Loader must be made aware of the name of the SMF dump data set. The Loader user guide describes how to modify `GLOBAL.SMFTBL2.GDGBASE` and `GLOBAL.SMFTBL2.DEFAULT` to make Loader aware of the location of the SMF data. In our member, the global variable was actually `GLOBAL2.SMFTBL2.DEFAULT`. From the menu in Figure 3-27, option **1, Global Variables**, could be used.

The SMF dump date sets used during our testing were created using IFASMFDP.

4

Data transformation

This chapter explains in-database transformation (IDT), part of the real-time analytics solution using SPSS Modeler and IBM DB2 Analytics Accelerator for z/OS capabilities.

This chapter contains the following topics:

- Introduction
- SQL pushback in SPSS Modeler
- Nodes supporting SQL generation for DB2 Accelerator
- In-database analytics Processing effort by components
- SPSS data transformation stream
- Stream messages
- Data transformation using DataStage

Note: IBM DB2 Analytics Accelerator for z/OS, DB2 Appliance, and the term *Accelerator* are used interchangeably through out this book.

© Copyright IBM Corp. 2016. All rights reserved.

4.1 Introduction

Data scientists need to work with good data to ensure their predictive models are accurate. However, data used to create models is often not readily consumable by data scientists. The data might contain missing or invalid entries, thus requiring some form of transformation to prepare the data before a model can be built. Data aggregation or generic transformation might be required before a data mining algorithm can be used. Accordingly, a data scientist will apply an appropriate method to evaluate the data and transform the data as required for the later steps in model creation and in model scoring.

After all the source data is loaded in to the Accelerator, the data is denormalized, cleansed, and prepared for modeling by using in-database transformation.

Figure 4-1 depicts the schematic of data load and data transformation processes. The DB2 Analytics Accelerator Loader for z/OS tool is discussed in Chapter 3, “Data integration using IBM DB2 Analytics Accelerator Loader for z/OS” on page 41.

Figure 4-1 does not show DataStage, which can also be used to perform extract, transform and load (ETL) and extract, load, and transform (ELT) from external data sources and use in-database transformation on DB2 Analytics Accelerator.

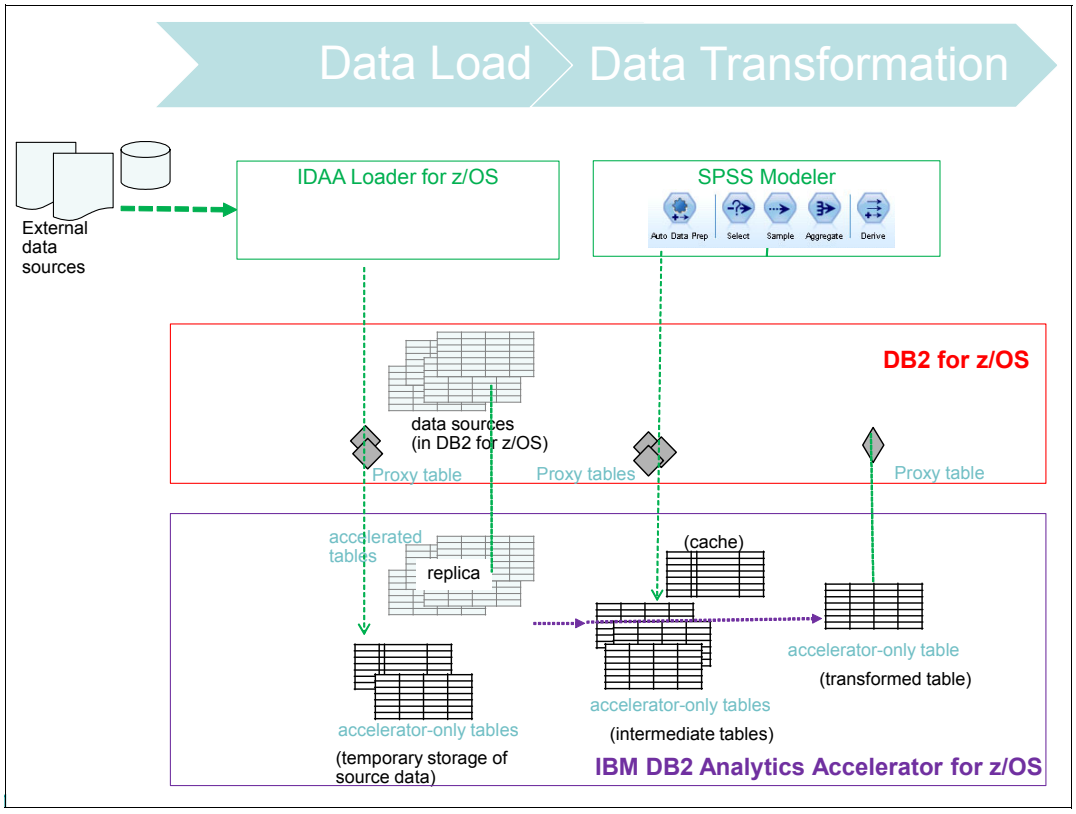


Figure 4-1 Data load and data transformation processes

This chapter explains how the data transformation phase of the analytics lifecycle can take advantage of the in-database transformation on IBM DB2 Analytics Accelerator for z/OS to transform data from one or more DB2 for z/OS tables (and external data through one or more intermediate tables to an accelerator-only table (AOT)) that are optimized for analytical processing.

4.1.1 Accelerator-only table (AOT)

The data preparation phase of the analytics lifecycle requires the presence of an analytics work environment, in which the data scientists can work with data and perform training and validation of predictive analytics models.

On platforms other than z/OS, the data preparation phase usually involves extracting data from OLTP data sources (often on z/OS), then loading and transforming (ELT or ETL) the data into the work environment. But, in a z/OS environment, performing in-database transformation (IDT) in the database accelerator (if the data source is on DB2 for z/OS) is possible. IBM DB2 Analytics Accelerator for z/OS allows data scientists to create the work environment on the DB2 appliance through the AOTs. This is done transparently and the data scientists can be blissfully ignorant of this work environment.

AOTs are tables that do not originate from DB2 base tables. Their data exists only on the Accelerator and not in DB2 for z/OS.

As Figure 4-1 on page 84 shows the following information:

- Data from other sources including VSAM and IMS on z/OS platform and also Social media, Oracle, SQL Server and other databases on other platforms can be extracted and loaded into the AOTs with the use of DB2 Analytics Accelerator Loader for z/OS tool.
- The data load and data transformation steps involved in our proposed solution, where data from external sources can be loaded into the AOT with the help of DB2 Analytics Accelerator Loader for z/OS tool. Those AOTs can then be joined with a replica of DB2 for z/OS tables, which coexists in the Accelerator (also called as Accelerated Tables). The joined result set can also go through the data preparation and data transformation on the Accelerator and eventually results in a single AOT that can be used as the input to the analytics modeling algorithms.

AOT versus accelerated replica tables

When a new AOT is created, an entry is added to the DB2 for z/OS catalog table `SYSIBM.SYSTABLES` to distinguish an AOT from regular tables and views by setting the TYPE column value to “D” (definition-only). The term *proxy table* on DB2 for z/OS is used to signify that only the definition exists on DB2 for z/OS and the corresponding data exists only on the Accelerator.

An AOT 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. If the `IN ACCELERATOR` clause is not specified, then that table is created only in DB2 for z/OS. If you need to accelerate a table that is only in DB2, you need to add it to the accelerator and then load it up with the data on the base table. This accelerated replica table on DB2 Analytics Accelerator can be kept in sync with the base DB2 for z/OS table using change data capture (CDC) based replication, which is part of the DB2 Analytics Accelerator solution. But, the AOT table has no matching data on DB2 for z/OS so you are not able to use CDC-based replication on AOTs.

For additional information, see Chapter 2 of *Accelerating Data Transformation with IBM DB2 Analytics Accelerator for z/OS*, SG24-8314.

The AOTs are used by SPSS Modeler in the data transformation phase in performing the following tasks:

- In-database caching, which can be used to improve the performance in certain data transformation scenarios
- Storing the transformed data (temporarily), which can then be used as an input to build the predictive model

Note: The temporary AOTs created during the data transformation phase are automatically deleted as soon as the predictive model is built successfully.

4.1.2 Enabling in-database processing on SPSS Modeler client

Completing the user preference setup on the SPSS modeler client to enable in-database analytics takes about 30 seconds. By changing the `EnableIDAA` parameter to `true` in the `user.prefs` (PREFS file), you can enable the in-database transformation and in-database modeling features on your SPSS Modeler.

A sample user preferences listing with the `EnableIDAA=true` setting on the `user.prefs` PREFS file (located in the users home directory in the directory path shown) is in 2.2.3, “SPSS Modeler client” on page 25.

This way allows the SPSS Modeler to execute all qualifying nodes directly on IBM DB2 Analytics Accelerator for z/OS. How to check whether the nodes in your SPSS stream can qualify for in-database execution are described in 4.5.1, “SQL Preview option to verify in-database processing” on page 100.

4.2 SQL pushback in SPSS Modeler

Wherever an IBM SPSS Modeler stream reads data from DB2 Accelerator to process the data, it can improve the efficiency of this operation by pushing back the SQL statements to execute in the Accelerator database.

SQL pushback feature of the SPSS Modeler basically generates SQL statements that can be pushed back to (that is, executed in) the DB2 for z/OS Accelerator database.

The DB2 Accelerator can then perform data sampling, cleansing, filtering, aggregating, joining, and so on. This is commonly referred to as in-database transformation (IDT) or in this case, accelerated in-database transformation.

4.2.1 How SQL generation works

The initial fragments of a stream leading from the database source nodes are the main targets for SQL generation. When a node is encountered that cannot be compiled to SQL, the data is extracted from the database and subsequent processing is performed by IBM SPSS Modeler server.

During stream preparation and prior to execution, the SQL generation process happens as follows:

1. The server reorders streams to move downstream nodes into the “SQL zone” where it can be proven safe to do so. (This feature can be disabled on the server.)
2. Working from the source nodes toward the terminal nodes, SQL expressions are constructed incrementally. This phase stops when a node is encountered that cannot be converted to SQL or when the terminal node is converted to SQL. At the end of this phase, each node is labeled with an SQL statement if the node and its predecessors have an SQL equivalent.
3. Working from the nodes with the most complicated SQL equivalents back toward the source nodes, the SQL is checked for validity. If the database does not return an error code on the prepare operation, the SQL is executed. Otherwise, the SPSS Modeler might try different SQL. If the SPSS Modeler is not able to generate SQL that can be processed by the database, it gives up by enumerating the source tables (or intermediate results from accelerator-only tables, in case up-stream nodes were already processed correctly) and then perform the operation, which it failed to do in the SQL directly in the SPSS Modeler server.
4. Nodes for which all operations have generated SQL are highlighted in purple on the stream canvas. For more information, read 4.5.1, “SQL Preview option to verify in-database processing” on page 100.
5. Based on the results, you might want to further reorganize your stream where appropriate to take full advantage of database execution. For instance, you might have to enable node caching in some cases to take advantage of in-database transformation.

SQL generation starts from source node and moves downstream; if any node in middle of the stream fails to generate SQL, the SQL generation stops for all the downstream nodes beginning from that unsuccessful node. The processing continues on the SPSS Modeler server for those downstream nodes.

4.2.2 Where improvements can occur with IDT using Accelerator

Products like DataStage and Cognos can use in-database processing using DB2 Analytics Accelerator to accelerate data transformation and report visualization queries respectively. In addition, SQL optimization in SPSS improves performance in a number of data operations by taking advantage of in-database transformation in the following operations:

- Joins (merge by key): Join operations can increase optimization within databases.
- Aggregation: The aggregate nodes use aggregation to produce their results. Summarized data uses considerably less bandwidth than the original data.
- Selection: Choosing records based on certain criteria reduces the quantity of records.
- Sorting: Sorting records is a resource-intensive activity that is performed more efficiently in a database.
- Field derivation: New fields are generated more efficiently in a database.
- Field projection: IBM SPSS Modeler server extracts only fields that are required for subsequent processing from the database, which minimizes bandwidth and memory requirements. The same is also true for superfluous fields in flat files: although the server must read the superfluous fields, it does not allocate any storage for them.
- Scoring: SQL can be generated from decision trees, linear regression models and so on.

In general, the fact that IDT allows minimizing data movement leads to the result that IDT streams outperform non-IDT (that is, traditional) streams. It allows pushing the transformation

operations to the database, and on top of it; IDT takes advantage of the massively parallel architecture of the Accelerator. But, if most of your stream does not qualify for IDT then the transformation performance might not be as good as one expects.

4.3 Nodes supporting SQL generation for DB2 Accelerator

The nodes are displayed at the bottom of the stream canvas on the SPSS Modeler. Each palette tab contains a collection of related nodes used for the phases of stream operations. This section describes all the nodes that might qualify for in-database execution on DB2 for z/OS with DB2 Analytics Accelerator. The nodes are tabulated for each palette tab.

4.3.1 Source palette tab

The nodes under this palette bring data into IBM SPSS Modeler. Table 4-1 lists the only source palette node that may qualify for in-database execution on DB2 for z/OS with DB2 Analytics Accelerator for z/OS.

Table 4-1 Source palette node that supports in-database transformation

Node	Node description
	Database node: This node is for specifying tables and views to be used for further analysis. It is the only node on the sources palette that can enable SQL pushback into DB2 Analytics Accelerator for in-database processing.

If you want to use any other data source, consider using other external tools like DataStage or DB2 Analytics Accelerator Loader to load that source data into an AOT and then use this Database node to perform your analysis. Also possible is to add generic SQL to this node.

Database source node: SQL query as input

With the SPSS Modeler Database source node, the only possibility is to add external database tables to the stream that are later on processed using a graphical, flow-oriented programming style. Also possible is to add a SQL SELECT statement directly, as shown in Figure 4-2 on page 89.

The steps are as follows:

1. Double-click **Database Source Node**.
2. Click the **Data** tab.
3. Select **SQL Query** as the mode.
4. Type (or paste) the SQL `SELECT` statement into the box below the Data source area. You can then act on that query using the buttons shown in Figure 4-2 on page 89, Such actions include Save Query, Apply, or OK.

Perhaps you are more comfortable expressing something in SQL directly instead of using multiple SPSS Modeler nodes, which can be handy. Also, if you have a third-party tool that can generate SQL, pasting such generated SQL might be a useful integration point with SPSS Modeler.

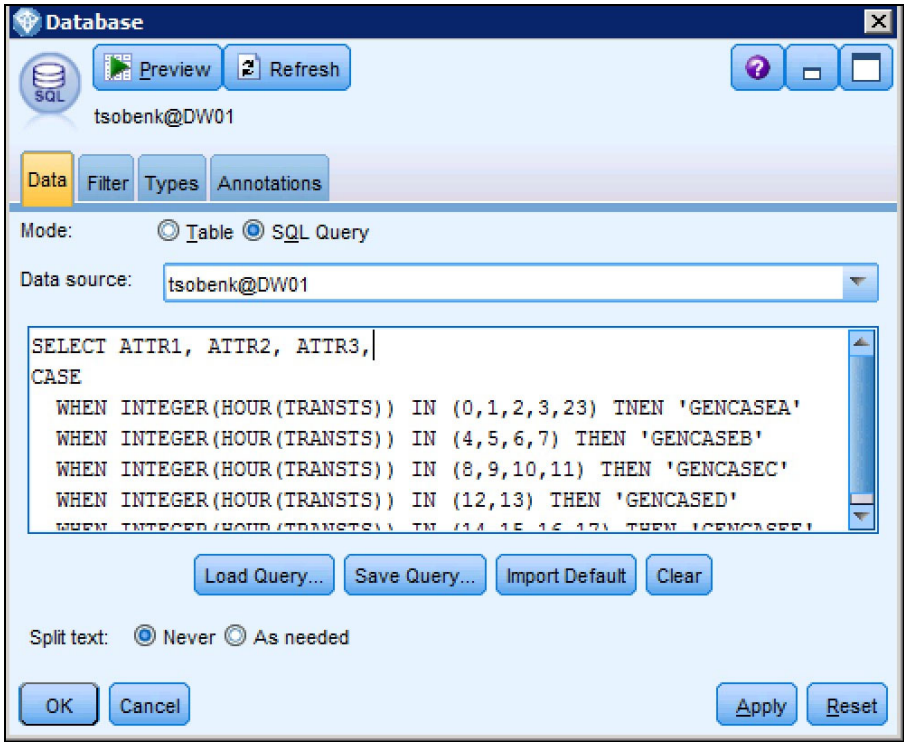


Figure 4-2 Database source node with SQL Query input

4.3.2 Record Ops palette tab

The nodes under this palette perform operations on data records, such as selecting, merging, sampling, and appending. Table 4-2 lists the Record Ops palette nodes that may qualify for in-database execution on DB2 for z/OS with DB2 Analytics Accelerator.

Table 4-2 Record Ops nodes that support in-database transformation

Node	Node description
	Select node: This node supports generation only if SQL generation for the select expression itself is supported.
	Sample node: This node is used to sample the input rows randomly. It supports SQL generation in DB2 Analytics Accelerator by using RAND() function in the WHERE clause of the generated SQL.
	Sort node: This node is used to sort the input rows. It supports SQL generation in DB2 Analytics Accelerator by using the ORDER BY clause on the generated SQL.

Node	Node description
	Distinct node: This node is used to sort the input rows. It supports SQL generation in DB2 Analytics Accelerator by using the DISTINCT clause on the SELECT statement. A Distinct node with (default) mode of *Create a composite record for each group selected* does not support SQL optimization. **Note:** If you have a Sort node followed by the Distinct node, then the Distinct node cannot be accelerated.
	Aggregate node: SQL generation support for aggregation depends on the data storage type.
	RFM Aggregate node: SQL optimization is only enabled when 'Read from Bin Values tab if available' is used. Does not support generation if saving the date of the second or third most recent transactions, or if only including recent transactions. However, including recent transactions does work if the `datetime_date(YEAR,MONTH,DAY)` function is pushed back.
	Merge node: Supports SQL generation for merge by condition. It generates a JOIN construct. Non-matching input fields can be renamed by means of a Filter node, or the Filter tab of a source node.
	Append node: Supports SQL generation if inputs are unsorted. **Note:** SQL optimization is possible only when your inputs have the same number of columns.

4.3.3 Field Ops palette tab

The nodes under this palette perform operations on data fields, such as filtering, deriving new fields, and determining the measurement level for given fields. Table 4-3 lists all Field Ops palette nodes that may qualify for in-database execution on DB2 for z/OS with DB2 Analytics Accelerator. All these nodes can generate SQL, if some restrictions are met and the SQL is accepted by DB2.

Table 4-3 Field Ops palette nodes that support in-database transformation

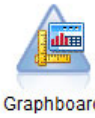
Node	Node description
	Auto Data Prep node: Before you can build a model, you need to specify which fields you want to use as targets and as inputs. This node prepares a field for analysis, it creates a new field containing the adjustments or transformations, rather than replacing the existing values and properties of the old field. The old field is not used in further analysis; its role is set to None. This node automatically creates Accelerator-only tables to accomplish the accepted data preparation changes. For more details read “Auto Data Prep Node” on page 98.
	Filter node: Allows fields to be renamed or removed.

Node	Node description
	Derive node: Allows new fields to be generated based on existing fields. Supports SQL generation only if SQL generated for the derive expression is supported.
	Filler node: Allows values in existing fields to be replaced by new values. For instance, if NULL is replaced with average.
	Reclassify node: Recategorizes set values. Restrictions might apply.
	Binning node: Use this to automatically create new nominal fields based on the values of one or more existing continuous (numeric range) fields. For example, you can transform a continuous income field into a new categorical field (naive Bayes algorithm requires categorical inputs) containing income groups of equal width, or as deviations from the mean. Sensitive personal information, such as salaries, may be reported in ranges rather than actual salary figures in order to protect data privacy.
	RFM Analysis node: The Recency, Frequency, Monetary (RFM) Analysis node enables you to determine quantitatively which customers are likely to be the best ones by examining how recently they last purchased from you (recency), how often they purchased (frequency), and how much they spent over all transactions (monetary).
	Partition node: Used to generate a partition field that splits the data into separate subsets or samples for the training, testing, and validation stages of model building. By using one sample to generate the model and a separate sample to test it, you can get a good indication of how well the model will generalize to larger data sets that are similar to the current data. Partitioning must be enabled on the Model Options tab.
	Set to Flag node: Used to derive flag fields based on the categorical values defined for one or more nominal fields. For example, your data set might contain a nominal field, BP (blood pressure), with the values High, Normal, and Low. For easier data manipulation, you might create a flag field for high blood pressure, which indicates whether or not the patient has high blood pressure.
	Restructure node: Although similar to that of the Set to Flag node, it offers more flexibility. It allows you to create fields of any type (including numeric flags), using the values from another field. You can then perform aggregation or other manipulations with other nodes downstream. The Set to Flag node lets you aggregate fields in one step, which might be convenient if you are creating flag fields.

4.3.4 Graphs palette tab

The nodes under this palette graphically display data before and after modeling. Graphs include plots, histograms, web nodes, and evaluation charts. Table 4-4 lists all the Graphs palette nodes that may qualify for in-database execution on DB2 for z/OS with DB2 Analytics Accelerator.

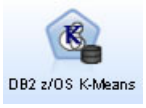
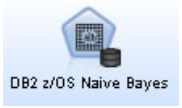
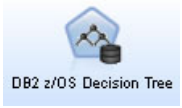
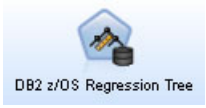
Table 4-4 Graphs palette nodes that support in-database transformation

Node	Node description
	Graphboard node: SQL generation is supported for the following graph types: Area, 3-D Area, Bar, 3-D Bar, Bar of Counts, Heat map, Pie, 3-D Pie, Pie of Counts. For Histograms, SQL generation is supported for categorical data only. SQL generation is not supported for Animation in Graphboard.
	Distribution node: Shows the distribution of symbolic values in a data set. This node is frequently used before manipulation nodes to explore the data and correct any imbalances. The Distribution node is unusual in that it produces both a graph and a table to analyze your data.
	Web node: Shows the strength of relationships between values of two or more symbolic fields. The graph displays connections using varying types of lines to indicate connection strength.
	Evaluation node: Offers an easy way to evaluate and compare predictive models to choose the best model for your application. Evaluation charts show how models perform in predicting particular outcomes.

4.3.5 Database Modeling (Nuggets) palette tab

By definition all the Database Modeling nodes in the Database Modeling palette qualify for in-database processing. The modeling nuggets on other Modeling palette may not qualify for offloading to the Accelerator. Each Database Modeling node listed in Table 4-5 will invoke a set of IBM Netezza Analytics stored procedures, which can run in DB2 Analytics Accelerator to build the requested model. This can be considered as an extension to SQL pushback in the sense that the stored procedures are pushed down to the DB2 Analytics Accelerator. Read more about these notes in “IBM Netezza Analytics stored procedures” on page 190.

Table 4-5 Database Modeling palette nodes that support in-database modeling

Node	Node description
	K-Means is a clustering algorithm to partition a set of entities into a number of clusters.
	Naive Bayes is a supervised learning classification algorithm applying Bayes theorem.
	Decision Tree is a supervised learning algorithm to create a decision tree model. Decision trees usually have the advantage that the found rules are easy to understand by humans.
	Regression Tree is similar to Decision Tree except the target is not a discrete value; instead it is a continuos real number.
	TwoStep Cluster node provides a form of cluster analysis. It can be used to cluster the data set into distinct groups when you do not know what those groups are at the beginning.

4.3.6 Output palette tab

The nodes under this palette produce a variety of output for data, charts, and model results that can be viewed in IBM SPSS Modeler.

4.3.7 Export palette tab

The nodes under this palette produce a variety of output that can be viewed in external applications, such as IBM Cognos for z/OS, IBM SPSS Data Collection or Excel. Table 4-6 lists all the Export palette nodes that may qualify for in-database execution on DB2 for z/OS with DB2 Analytics Accelerator.

With a database export node, you can export data to either a persistent table in DB2 z/OS or a persistent table on the Accelerator. If the used data source has an associated accelerator, a table on that accelerator is extended or created. If the used data source does not have an associated accelerator, the data is persisted in DB2 for z/OS. For example, this is useful if some of the transformation done for predictive model creation is also required for predictive model scoring, and the scoring should be done in-database, in DB2 z/OS, with same qualities of service as everything else in DB2 for z/OS transaction processing (IBM GDPS®, WLM, DB2 backup, and so on).

Table 4-6 Export palette nodes that support in-database transformation

Node	Node description
	Database node: Used to specify tables and views to be used for further analysis. This can be used to either insert new rows into an existing AOT or create a new AOT and insert rows.
	IBM Cognos BI Export node: Used to export data to Cognos BI Server.

4.4 In-database analytics Processing effort by components

Assuming that the accelerated tables are loaded on DB2 Analytics Accelerator, Table 4-7 gives an overview of the technical components involved and what processing effort they incur in general.

Table 4-7 In-database analytics processing effort by component tasks

Component	In-database tasks	In-database processing requirement
SPSS Modeler client	GUI to generate SPSS stream.	Minimal processing that occurs only on client workstation.
SPSS Modeler server	1. Convert SPSS stream to SQL. 2. Monitor SQL execution.	Low server CPU resource consumption (if everything is processed in-database).
DB2 for z/OS (with Analytics Accelerator)	1. Check security. 2. Maintain metadata (catalog). 3. Query rewrite. 4. Create wrapper stored procedures.	Minimal, mainly for security checking and some simple forwarding of slightly rewritten SQL.
Netezza-based DB2 Analytics Accelerator for z/OS	1. Do the actual data transformation. 2. Perform the actual queries. 3. Create the analytical models.	High massively parallel architecture with FPGAs and so on; optimized for exactly that but practically zero MIPS usage on the z/OS side.

The process interaction between the components, listed in Table 4-7, and others involved in in-database transformation is also shown in Figure 2-23 on page 37.

4.4.1 SPSS Modeler client (running on Windows)

The SPSS Modeler client is the GUI used by the data scientist to create SPSS streams. From the Modeler client, you connect to the Modeler server in order to perform SQL Preview and other activities.

4.4.2 SPSS Modeler server

Regardless of whether you are deploying the SPSS Modeler server on Linux on z Systems or a Windows environment, if the streams are executed 100% in-database then all that Modeler server does is transform the stream into optimized SQL and supervise the execution, trying to slightly change the generated SQL if the underlying DBMS returns an error condition.

ODBC driver

SPSS Modeler uses the locally installed ODBC driver to connect to the z/OS DB2 database subsystem over TCP/IP network.

4.4.3 DB2 for z/OS with Analytics Accelerator

The DB2 optimizer understands whether a given SQL statement can be pushed down to the Accelerator, and does a SQL statement rewrite in case acceleration is possible. For the sample stream in this book, everything can run in the Accelerator. When executing a remote procedure like IBM Netezza Analytics stored procedures, DB2 Analytics Accelerator guarantees in conjunction with DB2 that the IBM Netezza Analytics procedure on the Accelerator accesses only data the caller is authorized to access in DB2. The privileges are verified with DB2 z/OS.

4.4.4 Netezza based DB2 Analytics Accelerator for z/OS

In short, the Accelerator does all the real work. The DBMS inside the Accelerator must do all the real transformation on the accelerated and accelerator-only tables. For model creation, the installed IBM Netezza Analytics libraries are used (massively parallel).

4.5 SPSS data transformation stream

This section describes the design of the SPSS data transformation streams that can be used in a real-time analytics solution.

Figure 4-3 shows the data transformation stream used on the stream canvas pertaining to our sample scenario.

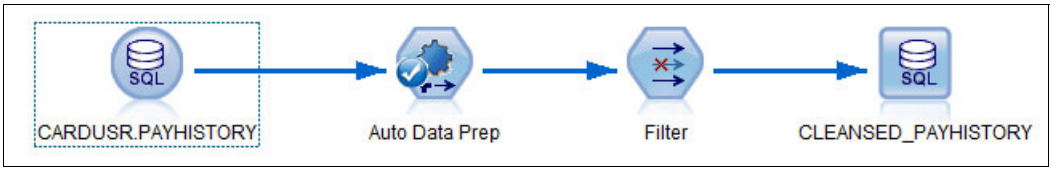


Figure 4-3 Sample data transformation stream

The first node in the data transformation stream is the Source Database node, which is pointed to by the `CARDUSR.PAYHISTORY` table stored in the DB2 Analytics Accelerator (Accelerated DB2 for z/OS table).

Double-click the Database node icon, then select the cataloged DSN source from the Data source drop-down list. It is listed as your *TSOUSERID@DSNALIAS*, shown in Figure 4-4 as `name@Data Source`.

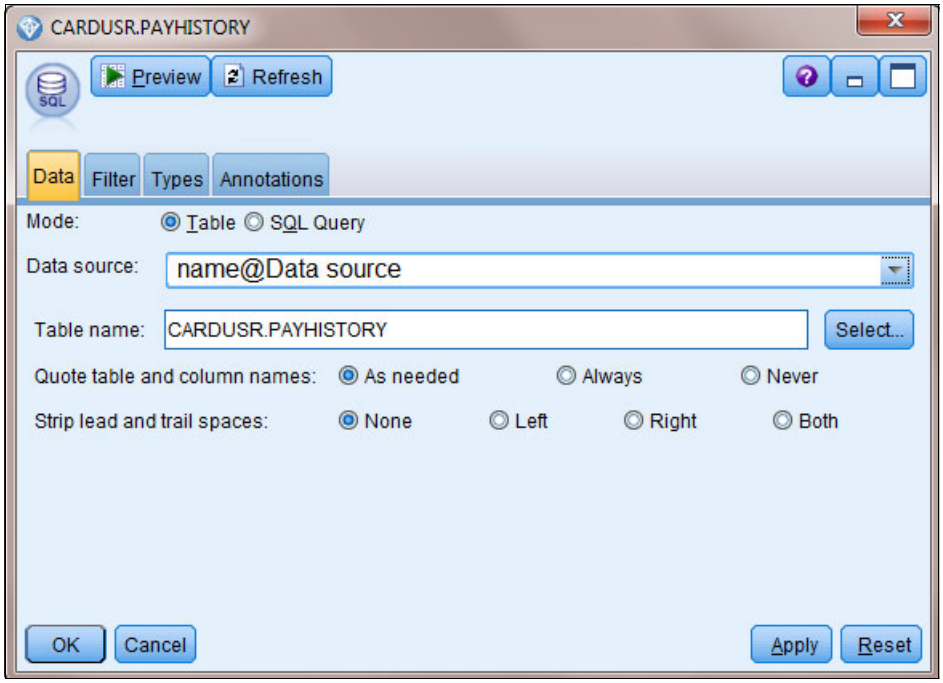
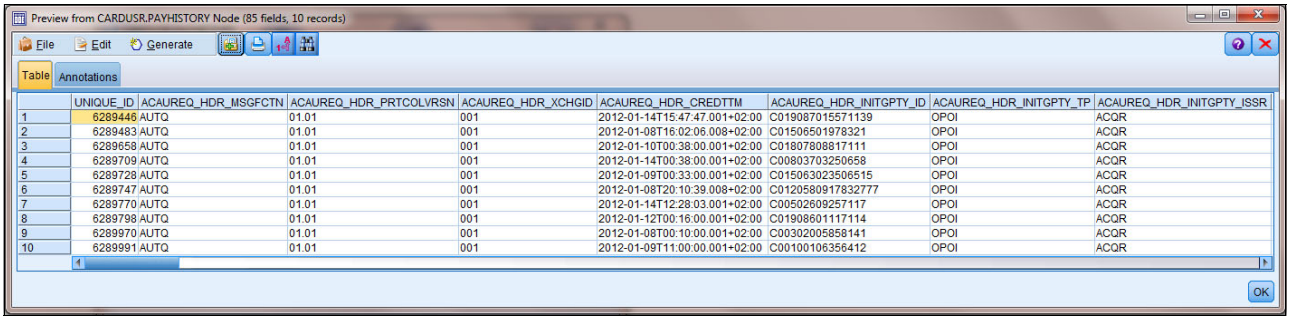


Figure 4-4 Source database node edit panel

From the database node panel shown in Figure 4-4, you can click **Preview** to view a sample of the data contained in the `CARDUSR.PAYHISTORY` table in DB2 Analytics Accelerator. The sample data is displayed in a new window (with 10 rows) as shown in Figure 4-5.



	UNIQUE_ID	ACAUREQ_HDR_MSGFCTN	ACAUREQ_HDR_PRTCOLVRSN	ACAUREQ_HDR_XCHGID	ACAUREQ_HDR_CREDTTM	ACAUREQ_HDR_INITGPTY_ID	ACAUREQ_HDR_INITGPTY_TP	ACAUREQ_HDR_INITGPTY_ISSR
1	6289446	AUTQ	01.01	001	2012-01-14T15:47:47.001+02:00	C019087015571139	OPOI	ACQR
2	6289483	AUTQ	01.01	001	2012-01-08T16:02:06.008+02:00	C01506501978321	OPOI	ACQR
3	6289658	AUTQ	01.01	001	2012-01-10T00:38:00.001+02:00	C01807808817111	OPOI	ACQR
4	6289709	AUTQ	01.01	001	2012-01-14T00:38:00.001+02:00	C00803703250658	OPOI	ACQR
5	6289728	AUTQ	01.01	001	2012-01-09T00:33:00.001+02:00	C01506302350651 5	OPOI	ACQR
6	6289747	AUTQ	01.01	001	2012-01-08T20:10:39.008+02:00	C0120580917832777	OPOI	ACQR
7	6289770	AUTQ	01.01	001	2012-01-14T12:28:03.001+02:00	C00502609257117	OPOI	ACQR
8	6289798	AUTQ	01.01	001	2012-01-12T00:16:00.001+02:00	C01908601117114	OPOI	ACQR
9	6289970	AUTQ	01.01	001	2012-01-08T00:10:00.001+02:00	C00302005858141	OPOI	ACQR
10	6289991	AUTQ	01.01	001	2012-01-09T11:00:00.001+02:00	C00100106356412	OPOI	ACQR

Figure 4-5 Data set Node Preview result set with 10 rows from the DB2 Analytics Accelerator table

Auto Data Prep Node

The second node in the stream, shown in Figure 4-3 on page 96, is the Auto Data Prep (ADP) node, which can help cleanse the data. Several options are offered on the Objectives tab, as shown in Figure 4-6.

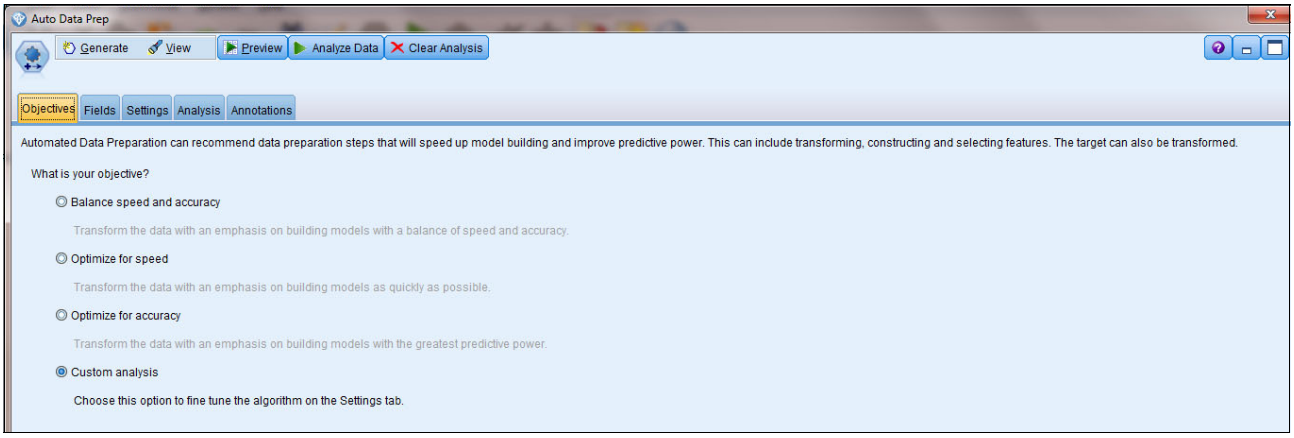
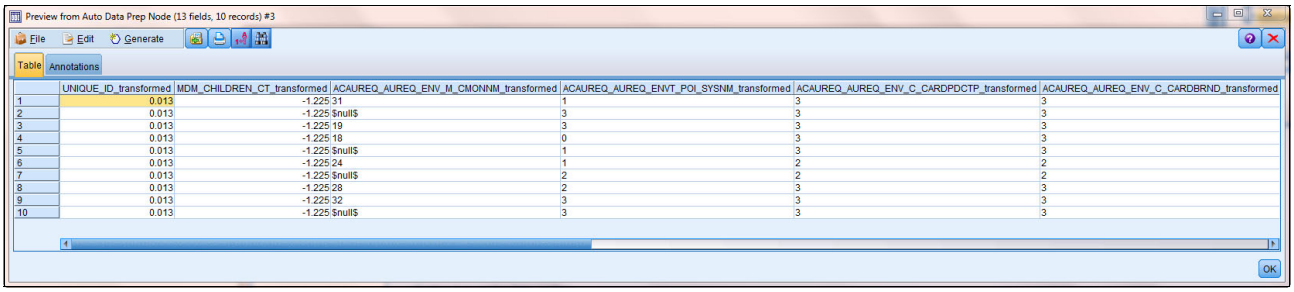


Figure 4-6 Objectives tab of the Auto Data Prep Node

The Auto Data Prep node was not initialized before it was used for further SQL generation. The two ways to initialize the ADP node are as follows:

- Open the node, click **Analyze Data**, select the **Analysis** tab, and make sure the summaries were generated.
- Run the stream twice, which forces the ADP node to be initialized.

The Auto Data Prep operation adds the suffix `_transformed` to each affected field name, as shown in Figure 4-7. This suffix might cause the length of the field name to be extended beyond the 30-byte limit allowed in IBM DB2 11 for z/OS.



	UNIQUE_ID_transformed	MDM_CHILDREN_CT_transformed	ACAUREQ_AUREQ_ENV_M_CMONNM_transformed	ACAUREQ_AUREQ_ENVT_POI_SYSNM_transformed	ACAUREQ_AUREQ_ENV_C_CARDPDCTP_transformed	ACAUREQ_AUREQ_ENV_C_CARDBRND_transformed
1	0.013	-1.225	31	1	3	3
2	0.013	-1.225	$null$	3	3	3
3	0.013	-1.225	19	3	3	3
4	0.013	-1.225	18	0	3	3
5	0.013	-1.225	$null$	1	3	3
6	0.013	-1.225	24	1	2	2
7	0.013	-1.225	$null$	2	2	2
8	0.013	-1.225	28	2	3	3
9	0.013	-1.225	32	3	3	3
10	0.013	-1.225	$null$	3	3	3

Figure 4-7 Preview window showing the transformed field names from the Auto Data Prep node

Because of the extended length, another data cleansing step is needed to shorten the names of the transformed fields. To rename the transformed field, use a filter and then edit the field names:

1. Double-click the **Filter** node icon (shown as the third node in Figure 4-3 on page 96).
2. The Filter window opens (Figure 4-8). Double-click the field names in the right column to edit them.

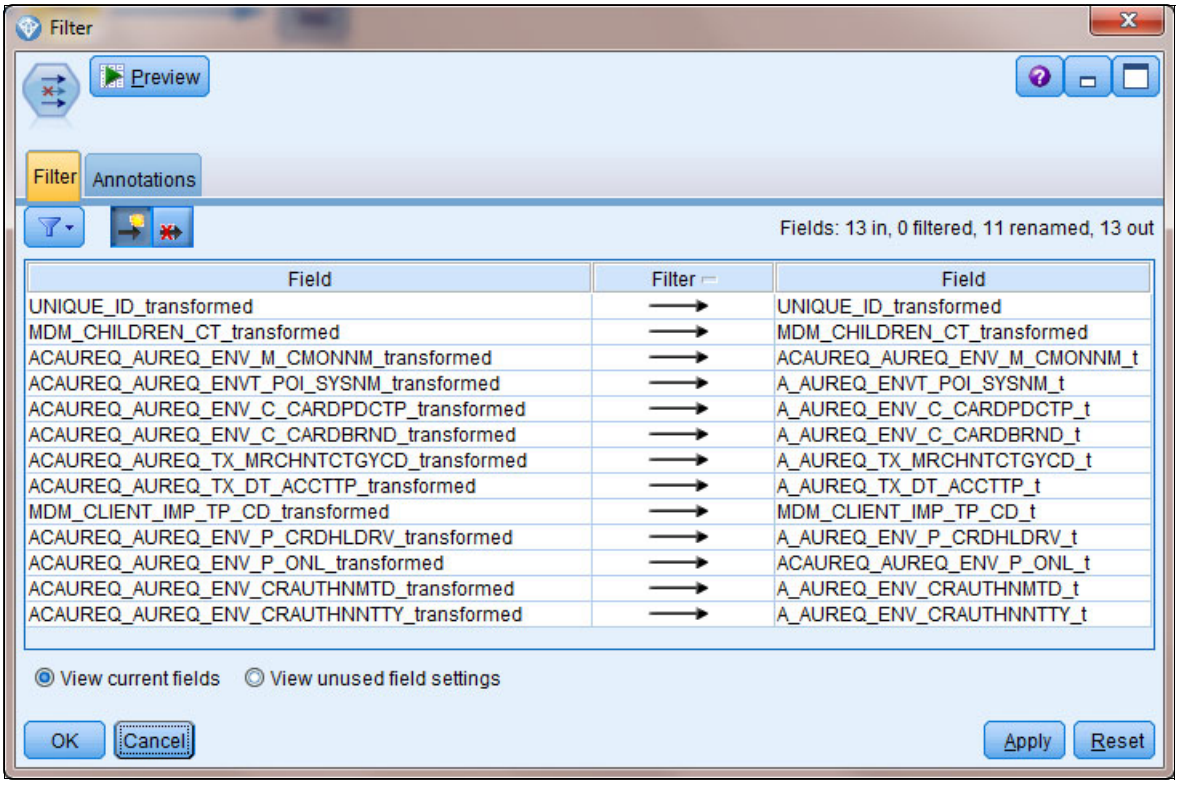


Figure 4-8 Filter node edit panel to rename the transformed fields

The last node is from the Export palette and in our case it is the Database node, which is the `CLEANSED_PAYHISTORY` table, and is also stored in the DB2 Analytics Accelerator. Double-click the **Export** node icon, then select the source from the Data source Drop-down list as shown in Figure 4-4 on page 97. The CLEANSED_PAYHISTORY table contains the output of the data transformation stream.

4.5.1 SQL Preview option to verify in-database processing

To verify whether your stream will run in-database or not without executing the stream, you can use the SQL Preview icon (purple hexagon) on your SPSS Modeler client main window, as shown in Figure 4-9.

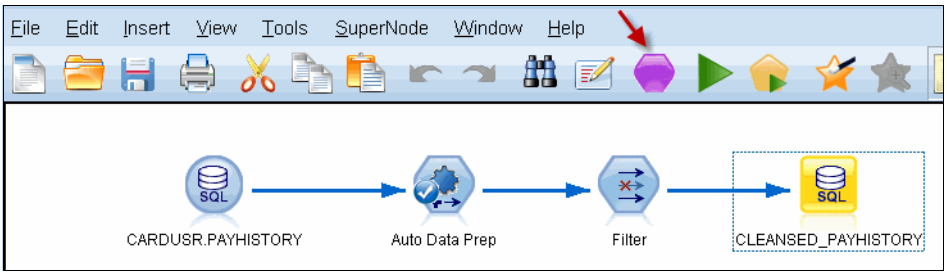


Figure 4-9 SQL Preview icon to check in-database processing prior to stream execution

This hexagonal shaped icon is enabled automatically when you select a terminal node on your stream canvas, otherwise is disabled. Terminal node can be any node from output, graph, Database Modeling, or export palette. In the sample stream shown in Figure 4-9, the Database Node (CLEANSED_PAYHISTORY table) is taken from the Export palette. Because this table is an AOT, the entire stream qualifies for in-database transformation (Figure 4-10).

4.5.2 Why are my nodes purple

The nodes that qualify for in-database execution will change their color to purple during the execution of the stream (or during SQL Preview operation as discussed in 4.5.1, “SQL Preview option to verify in-database processing” on page 100). The nodes in Figure 4-3 on page 96 changes its color to purple as shown in Figure 4-10 when the underlying processes can run or are executed in-database (that is, in DB2 Analytics Accelerator in this case).

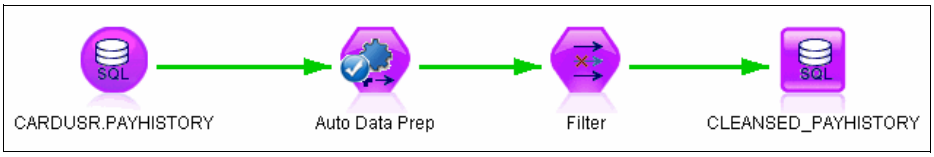


Figure 4-10 Stream execution in DB2 Analytics Accelerator highlighting in-database nodes

For high level information about what those underlying processes (or SQL operations) are that are performed in-database for the sample stream, see 4.6, “Stream messages” on page 101.

4.5.3 Enable cache at node level

For streams run in a database, data can be cached midstream to a temporary table in the Accelerator (AOT). When combined with SQL optimization, this can result in significant gains in performance. For example, the output from a stream that merges multiple tables to create a data mining view can be cached and reused as needed. By automatically generating SQL for all downstream nodes, performance can be further improved.

To take advantage of database caching, both SQL optimization and database caching must be enabled. Note that server optimization settings override those on the client.

For more information about setting optimization options for streams, select **Help** → **Help Topics** in the IBM SPSS Modeler Tool bar. A web browser opens to the local IBM SPSS Modeler Help page (no Internet connection is necessary). From there, go to the Table of Contents on the left side and select **User's Guide** → **Building Streams** → **Building Data Streams** → **Working with Streams** → **Setting Options for Streams**.

With database caching enabled, right-click any non-terminal node to cache data at that point, and the cache will be created automatically directly in the database the next time the stream is run. If database caching or SQL optimization is not enabled, the cache will be written to the file system instead, which is not desirable with in-database analytics.

In some situations, caching of a node can enable SQL generation with complex streams. For example, DB2 z/OS does not support more than 500 tables in a single query but we have successfully run client streams with several thousand tables using in-database caching in a proof of concept.

Regardless of whether CDC replication is used on the accelerated tables (source data) in a given stream, any subsequent data changes to the underlying tables in the upstream are not propagated to the cached AOTs.

4.6 Stream messages

The stream messages will give you a clear idea of what is happening when in-database analytics is in effect.

The stream messages can be viewed from the SPSS Modeler client through the Tools menu by navigating through **Stream Properties** as shown in Figure 4-11.

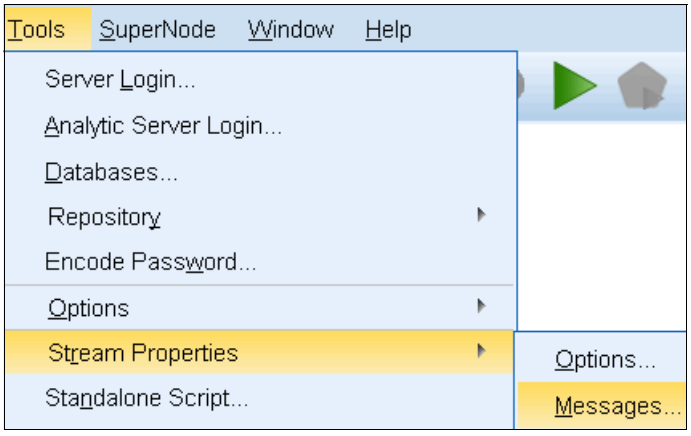


Figure 4-11 Tools menu navigation to view stream execution messages

The Message section of the Messages tab lists the steps processed behind the scenes when a stream is executed. For illustration, the stream shown in Figure 4-10 on page 100 has a message summary shown in Figure 4-12.

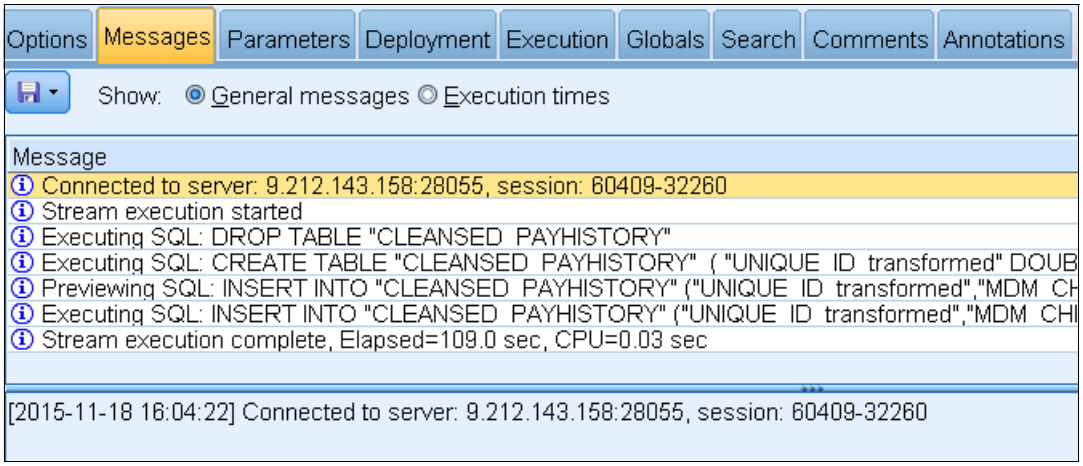


Figure 4-12 Sample stream messages tab from SPSS client

The letter “i” to the left of each message means that these messages are informational. The message steps are as follows:

1. Message one shows the I/P address of the SPSS server that you are connected to and the automatically generated session identifier.
2. Message two says that the stream execution started.
3. Message three through six indicate that the SQL Modeler connected to the database (in our case, DB2 for z/OS with DB2 Analytics Accelerator) is doing these SQL operations:
   a. Execute dropping (DROP) the output table, which is an AOT, essentially to make sure that the structure and data on the output table at the end of stream execution will be that of this stream’s.
   b. Execute creating (CREATE) a table.
   c. Preview inserting (INSERT).
   d. Execute inserting (INSERT).
4. Message four indicates the creation of the output table as an AOT using the DDL shown in the Example 4-1, which you can view on the bottom part of the Messages tab when you click the summary message.

Example 4-1 CREATE AOT statement

```
CREATE TABLE "CLEANSED_PAYHISTORY"  ( "UNIQUE_ID_transformed"
DOUBLE,"MDM_CHILDREN_CT_transformed" DOUBLE,"ACAUREQ_AUREQ_ENV_M_CMONNM_t"
VARCHAR(8),"A_AUREQ_ENVT_POI_SYSNM_t" VARCHAR(4),"A_AUREQ_ENV_C_CARDPDCTP_t"
VARCHAR(4),"A_AUREQ_ENV_C_CARDBRND_t" VARCHAR(4),"A_AUREQ_TX_MRCHNTCTGYCD_t"
VARCHAR(8),"A_AUREQ_TX_DT_ACCTTP_t" VARCHAR(4),"MDM_CLIENT_IMP_TP_CD_t"
VARCHAR(4),"A_AUREQ_ENV_P_CRDHLDRV_t" VARCHAR(4),"ACAUREQ_AUREQ_ENV_P_ONL_t"
VARCHAR(4),"A_AUREQ_ENV_CRAUTHNMTD_t" VARCHAR(4),"A_AUREQ_ENV_CRAUTHNNTTY_t" VARCHAR(4)
)  IN ACCELERATOR "SBSDEVV4" CCSID UNICODE
```

5. Message five indicates that the SQL preview is done for the consolidated transformed data to be inserted into the output AOT. The sample SQL statement for this step, as generated by the SPSS Modeler is shown in Example 4-2.

Example 4-2 Sample SQL statement that creates and populates an AOT

```
INSERT INTO "CLEANSED_PAYHISTORY"
("UNIQUE_ID_transformed","MDM_CHILDREN_CT_transformed","ACAUREQ_AUREQ_ENV_M_CMONNM_t","A
_AUREQ_ENVT_POI_SYSNM_t","A_AUREQ_ENV_C_CARDPDCTP_t","A_AUREQ_ENV_C_CARDBRND_t","A_AUREQ
_TX_MRCHNTCTGYCD_t","A_AUREQ_TX_DT_ACCTTP_t","MDM_CLIENT_IMP_TP_CD_t","A_AUREQ_ENV_P_CRD
HLDRV_t","ACAUREQ_AUREQ_ENV_P_ONL_t","A_AUREQ_ENV_CRAUTHNMTD_t","A_AUREQ_ENV_CRAUTHNNTTY
_t")
   SELECT ((DOUBLE(1.9030898725523110e-19) * DOUBLE((CAST(TO."UNIQUE_ID" AS FLOAT) -
-6.9156586044167104e+16))) + 0.0000000000000000e+00) AS
"UNIQUE_ID_transformed",((DOUBLE(7.3055302837382196e-08) * DOUBLE(({fn
CONVERT(TO."MDM_CHILDREN_CT",SQL_FLOAT)} - 1.6773860575300001e+07))) +
0.0000000000000000e+00) AS "MDM_CHILDREN_CT_transformed",(CASE WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'Pins and Balls') THEN '00' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'One More Drink') THEN '01' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'Don''t be alone') THEN '02' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'Fair Rides') THEN '03' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'Motor Parts') THEN '04' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'FourSeasons') THEN '05' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'Allegro') THEN '06' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'Donna Affascinante') THEN '07' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'Donna Elegante') THEN '08' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'Shoes') THEN '09' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'Uomo Elegante') THEN '10' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'Rinascimento') THEN '11' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'M and W') THEN '12' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'Family Clothing') THEN '13' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'LittleAdult') THEN '14' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'OminodiFerro') THEN '15' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'SmartKids') THEN '16' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'Restaurant') THEN '17' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'Eat and Go') THEN '18' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'ToBeOrNotTobe') THEN '19' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'Books & Paper') THEN '20' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'Musicae') THEN '21' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'LP and CD') THEN '22' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'House of Drapery') THEN '23' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'All for your House') THEN '24' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'Home of Crystal') THEN '25' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'Hardware and Equipment') THEN '26' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'My Beatiful House') THEN '27' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'Plumbing and Heating') THEN '28' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'Electric Parts Equ') THEN '29' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'Masonry and Plaster') THEN '30' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'AllFood') THEN '31' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'Incrocio') THEN '32' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'TuttoePiu') THEN '33' WHEN
(TO."ACAUREQ_AUREQ_ENV_M_CMONNM" = 'Movies') THEN '34' ELSE NULL END) AS
"ACAUREQ_AUREQ_ENV_M_CMONNM_t",(CASE WHEN (TO."ACAUREQ_AUREQ_ENVT_POI_SYSNM" = 'POI13')
THEN '0' WHEN (TO."ACAUREQ_AUREQ_ENVT_POI_SYSNM" = 'POI11') THEN '1' WHEN
(TO."ACAUREQ_AUREQ_ENVT_POI_SYSNM" = 'POI14') THEN '2' WHEN
(TO."ACAUREQ_AUREQ_ENVT_POI_SYSNM" = 'POI12') THEN '3' ELSE NULL END) AS
"A_AUREQ_ENVT_POI_SYSNM_t",(CASE WHEN (TO."ACAUREQ_AUREQ_ENV_C_CARDPDCTP" = '0006') THEN
'0' WHEN (TO."ACAUREQ_AUREQ_ENV_C_CARDPDCTP" = '0005') THEN '1' WHEN
(TO."ACAUREQ_AUREQ_ENV_C_CARDPDCTP" = '0004') THEN '2' WHEN
(TO."ACAUREQ_AUREQ_ENV_C_CARDPDCTP" = '0003') THEN '3' ELSE NULL END) AS
"A_AUREQ_ENV_C_CARDPDCTP_t",(CASE WHEN (TO."ACAUREQ_AUREQ_ENV_C_CARDBRND" = 'Amex
```

```
Platinum') THEN '0' WHEN (TO."ACAUREQ_AUREQ_ENV_C_CARDBRND" = 'Visa Gold') THEN '1' WHEN
(TO."ACAUREQ_AUREQ_ENV_C_CARDBRND" = 'Credit Card') THEN '2' WHEN
(TO."ACAUREQ_AUREQ_ENV_C_CARDBRND" = 'Debit Card') THEN '3' ELSE NULL END) AS
"A_AUREQ_ENV_C_CARDBRND_t",(CASE WHEN (TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '7933') THEN
'00' WHEN (TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '5813') THEN '01' WHEN
(TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '7273') THEN '02' WHEN
(TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '7996') THEN '03' WHEN
(TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '5031') THEN '04' WHEN
(TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '5631') THEN '05' WHEN
(TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '5621') THEN '06' WHEN
(TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '5611') THEN '07' WHEN
(TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '5661') THEN '08' WHEN
(TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '5651') THEN '09' WHEN
(TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '5812') THEN '10' WHEN
(TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '5814') THEN '11' WHEN
(TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '7922') THEN '12' WHEN
(TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '5311') THEN '13' WHEN
(TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '5192') THEN '14' WHEN
(TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '5733') THEN '15' WHEN
(TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '5735') THEN '16' WHEN
(TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '5641') THEN '17' WHEN
(TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '5714') THEN '18' WHEN
(TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '5722') THEN '19' WHEN
(TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '5950') THEN '20' WHEN
(TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '5072') THEN '21' WHEN
(TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '5719') THEN '22' WHEN
(TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '1711') THEN '23' WHEN
(TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '5065') THEN '24' WHEN
(TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '1740') THEN '25' WHEN
(TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '7832') THEN '26' WHEN
(TO."ACAUREQ_AUREQ_TX_MRCHNTCTGYCD" = '5711') THEN '27' ELSE NULL END) AS
"A_AUREQ_TX_MRCHNTCTGYCD_t",(CASE WHEN (TO."ACAUREQ_AUREQ_TX_DT_ACCTTP" = 'CRDT') THEN
'0' WHEN (TO."ACAUREQ_AUREQ_TX_DT_ACCTTP" = 'CHCK') THEN '1' ELSE NULL END) AS
"A_AUREQ_TX_DT_ACCTTP_t",(CASE WHEN (TO."MDM_CLIENT_IMP_TP_CD" = '1') THEN '0' WHEN
(TO."MDM_CLIENT_IMP_TP_CD" = '2') THEN '1' WHEN (TO."MDM_CLIENT_IMP_TP_CD" = '3') THEN
'2' WHEN (TO."MDM_CLIENT_IMP_TP_CD" = '4') THEN '3' ELSE NULL END) AS
"MDM_CLIENT_IMP_TP_CD_t",(CASE WHEN ((CASE WHEN (TO."ACAUREQ_AUREQ_ENV_P_CRDHLDRV" IS
NULL) THEN 'MNSG' ELSE TO."ACAUREQ_AUREQ_ENV_P_CRDHLDRV" END) = 'MNVR') THEN '0' WHEN
((CASE WHEN (TO."ACAUREQ_AUREQ_ENV_P_CRDHLDRV" IS NULL) THEN 'MNSG' ELSE
TO."ACAUREQ_AUREQ_ENV_P_CRDHLDRV" END) = 'MNSG') THEN '1' ELSE NULL END) AS
"A_AUREQ_ENV_P_CRDHLDRV_t",(CASE WHEN ((CASE WHEN (TO."ACAUREQ_AUREQ_ENV_P_ONL" IS NULL)
THEN 'OFLN' ELSE TO."ACAUREQ_AUREQ_ENV_P_ONL" END) = 'SMON') THEN '0' WHEN ((CASE WHEN
(TO."ACAUREQ_AUREQ_ENV_P_ONL" IS NULL) THEN 'OFLN' ELSE TO."ACAUREQ_AUREQ_ENV_P_ONL"
END) = 'OFLN') THEN '1' ELSE NULL END) AS "ACAUREQ_AUREQ_ENV_P_ONL_t",(CASE WHEN ((CASE
WHEN (TO."ACAUREQ_AUREQ_ENV_CRAUTHNMTD" IS NULL) THEN 'BYPS' ELSE
TO."ACAUREQ_AUREQ_ENV_CRAUTHNMTD" END) = 'MERC') THEN '0' WHEN ((CASE WHEN
(TO."ACAUREQ_AUREQ_ENV_CRAUTHNMTD" IS NULL) THEN 'BYPS' ELSE
TO."ACAUREQ_AUREQ_ENV_CRAUTHNMTD" END) = 'BYPS') THEN '1' ELSE NULL END) AS
"A_AUREQ_ENV_CRAUTHNMTD_t",(CASE WHEN ((CASE WHEN (TO."ACAUREQ_AUREQ_ENV_CRAUTHNNTTY" IS
NULL) THEN 'AGNT' ELSE TO."ACAUREQ_AUREQ_ENV_CRAUTHNNTTY" END) = 'MERC') THEN '0' WHEN
((CASE WHEN (TO."ACAUREQ_AUREQ_ENV_CRAUTHNNTTY" IS NULL) THEN 'AGNT' ELSE
TO."ACAUREQ_AUREQ_ENV_CRAUTHNNTTY" END) = 'AGNT') THEN '1' ELSE NULL END) AS
"A_AUREQ_ENV_CRAUTHNNTTY_t"
    FROM CARDUSR.PAYHISTORY TO
```

6. Message six indicates the SQL shown in the preview step is executed to create and populate the output AOT.

   Note: If you enable cache on any of the data transformation nodes in your stream then it would essentially result in creation and population of an AOT. For more information about caching, see 4.5.3, “Enable cache at node level” on page 100.

7. Message seven shows you the run-time details like total elapsed time and CPU time consumption on the SPSS Server to execute the stream.

4.7 Data transformation using DataStage

If you are using a non z/OS distributed platform to perform your predictive analytics modeling on DB2 for z/OS operational data now, and the process to build your external tables already exists as DataStage jobs, you can use the Accelerator to eliminate this data sprawl.

To use in-database transformation as efficiently as possible in DataStage, 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.

5

Data modeling using SPSS and DB2 Analytics Accelerator

The third phase in the data lifecycle is data modeling. This chapter describes the design of the SPSS data model stream used in this solution. The design supports the solution to the business problem discussed in the beginning of this book. For more information about IBM SPSS Modeler with Scoring Adapter for DB2 z/OS, see the following web page:

http://www.ibm.com/software/analytics/spss/products/modeler/

This chapter contains the following topics:

- Introduction
- SPSS data modeling stream design
- Adding record IDs to input tables
- Combining in-database transformation on Accelerator with traditional model creation

© Copyright IBM Corp. 2016. All rights reserved.

5.1 Introduction

IBM SPSS Modeler with Scoring Adapter for z Systems platform is a software package designed for building, validating, and deploying accurate predictive data models and has a powerful predictive analytics engine. The graphical user interface provides an efficient way to connect data sources to models, run advanced analyses, and deliver results. For example, data scientists can develop a model to analyze existing patterns and trends in customer purchases, evaluate current transactions, and calculate resulting scores. Ultimately, the aggregation of data and results can be used to predict a customer's future purchasing behavior. IBM SPSS Modeler can also deploy a Predictive Model Markup Language (PMML) model to the scoring adapter or some other scoring component. Furthermore, it can store results like transformed data needed in transactional systems or batch score results back to the OLTP system.

If a data scientist interactively works with source data to do model creation accelerated in-database, the advantages are that all the real work must be done by the Accelerator, that no data access or data movement cost exists in z/OS DB2, and that the overall elapsed time, exploiting the massively parallel architecture of the Netezza based accelerator with dedicated FPGAs and analytics-optimized system architecture is significantly shorter, especially if the amount of data to be processed is significant.

Note that for small models, accelerated in-database modeling does not always lead to shortened elapsed time as the IBM Netezza Analytics models used have a startup time of approximately 1 - 2 minutes. In our observations, IBM Netezza Analytics models are usually faster if the input table has at least one million rows. For smaller tables, SPSS Modeler alone is usually able to create a model on one Integrated Facility for Linux (IFL) on z13 in less than the mentioned startup time of IBM Netezza Analytics (if it is connected using HiperSockets and z/OS DB2 delivers the requested data at full speed).

5.2 SPSS data modeling stream design

The end-to-end data lifecycle phases are divided into smaller streams. The following sections describe the design of the SPSS in-database modeling stream used in this solution. The nodes are selected from those available in the tables presented in 4.3, "Nodes supporting SQL generation for DB2 Accelerator" on page 88.

5.2.1 SPSS data modeling stream

Figure 5-1 on page 109 shows the data modeling stream used in this solution. This figure is a representation of the basic components of a modeling stream. Simple scenarios might require only a Source node and a Model node; more complex scenarios might require additional nodes for further filtering beyond what was performed in the data transformation stream.

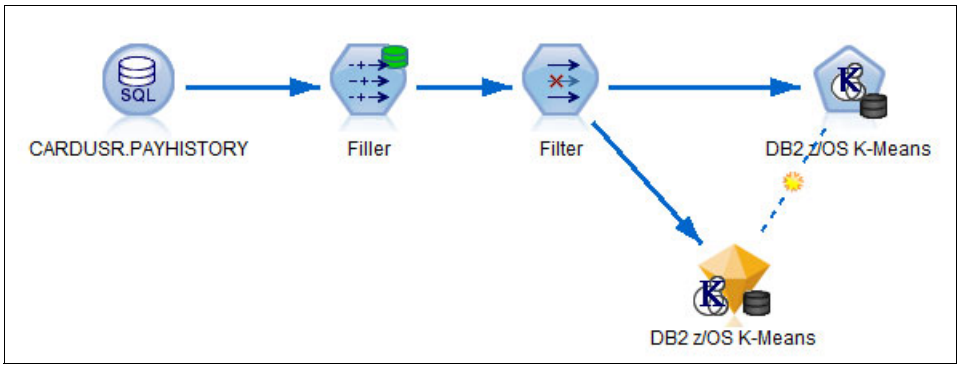


Figure 5-1 SPSS data modeling stream

Figure 5-1 shows the following nodes (from left to right) in the data modeling stream:

- The first node is a Source node. In this solution, the data source is an IBM DB2 table named `CARDUSR.PAYHISTORY`. In practice, however, this node can be replaced with the output of the data transformation stream.
- The second node is a Filler node. The Filler node is used to help cleanse the data by replacing null values for any field in the table with a choice of a value or an expression. Double-click the **Filler** node icon and click the **Settings** tab to access this feature. An example of the logic added for this solution is shown in Figure 5-2 on page 110.

  The green cylindrical database symbol in the upper-right corner of the Filler node icon indicates that caching is enabled. By selecting this feature, cache is created in the database automatically, and data is stored in a temporary table rather than in the file system. This feature can improve performance in streams with tables that are used frequently.

  With SPSS Modeler, you can enable caching for individual nodes. For in-database modeling, in-database caching is required, which means that in the stream where the cache is enabled, SPSS Modeler creates a temporary table (in-database if accelerator is associated to the used data source) and fills it using an INSERT FROM SELECT statement, caching everything up to that node. This process can help DB2 with Accelerator process complex queries.

  Note: While testing our prototype at a client, we found that the existing SPSS streams of that client lead to SQL with about 1200 tables. Because z/OS DB2 has a limitation that a query cannot contain more than 500 tables, those client streams were able to be processed only by introducing in-database caching. Nevertheless, the client was satisfied with the reduction of stream execution time from days (with heavily used DB2) to minutes (running completely within DB2 Accelerator).

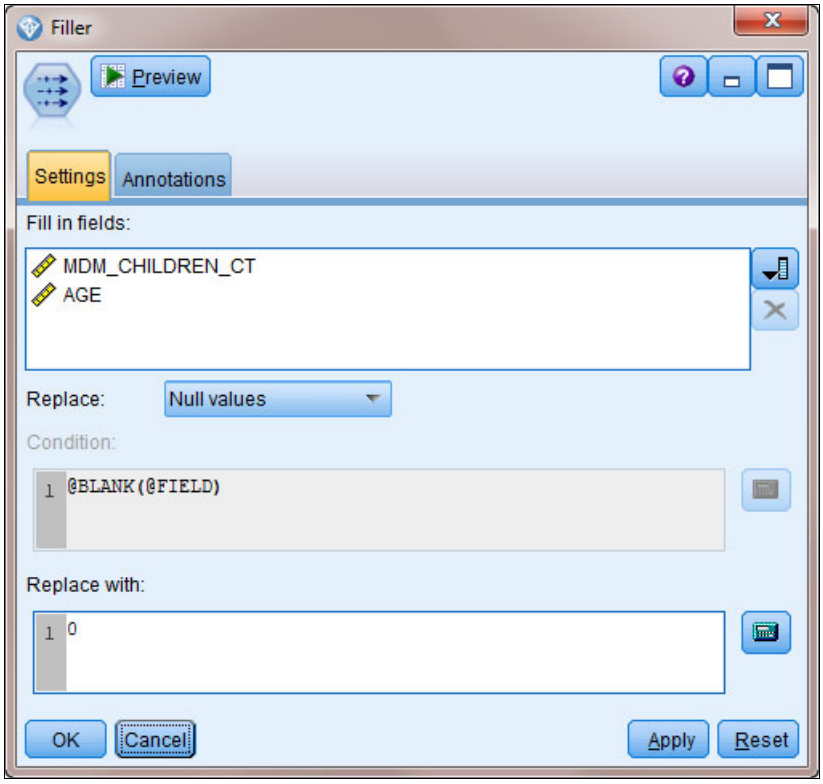


Figure 5-2 Filler node settings

- The third node is a Filter node. The Filter node is used to filter the fields that are required to build the model. Double-click the **Filter** node icon, then double-click the field names in the right column to select or deselect the desired fields. An example of the fields selected for this solution is shown in Figure 5-3.

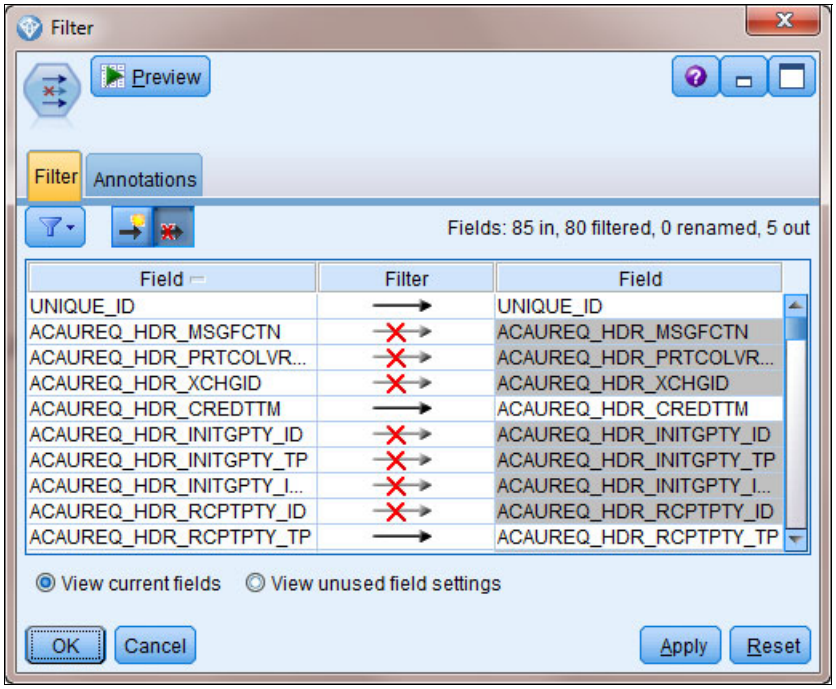


Figure 5-3 Filter node settings

- The last node is the Database Model node. The Database Model palette contains five model options: DB2 z/OS K-Means, DB2 z/OS Naive Bayes, DB2 z/OS Decision Tree, DB2 z/OS Regression Tree, and DB2 z/OS TwoStep.

  A DB2 z/OS K-Means model is used to perform a cluster analysis on the data in the z/OS DB2 table for this solution. Figure 5-4 shows the fields included in this model.

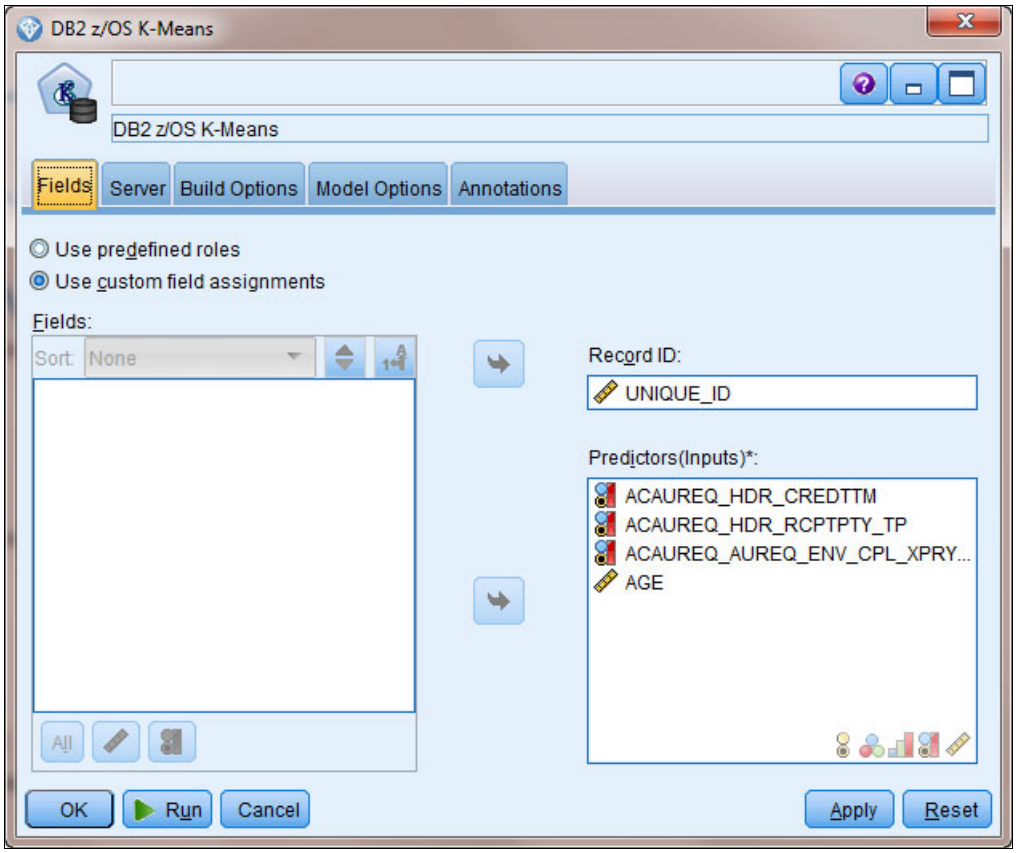


Figure 5-4 DB2 z/OS K-Means model fields

After running the data modeling stream, the model nugget icon is created (shown in Figure 5-1 on page 109 and Figure 5-5). The model is now ready to publish to DB2 and is also temporarily stored in the DB2 Analytics Accelerator. More information about publishing the model to DB2 is in 6.2.1, “Real-time scoring and near real-time scoring” on page 119.

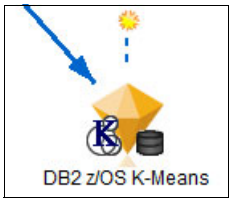


Figure 5-5 The K-Means model nugget

5.2.2 Example of how to optimize pruning a decision tree

Especially for a decision tree model, one of the problems to avoid is *overfitting*, which means that the predictive model does not describe the real-world relationships but rather random error and noise. In an extreme case, the model can describe the used training data set perfectly well however this does not guarantee that scoring of future items has a high precision. In addition and if possible, a favorable step is to reduce the complexity of a model in general by replacing sub-trees with leaves.

SPSS Modeler decision tree nodes offer an option to use a given percentage of input data for model training and the rest for tree pruning, as shown Figure 5-6.

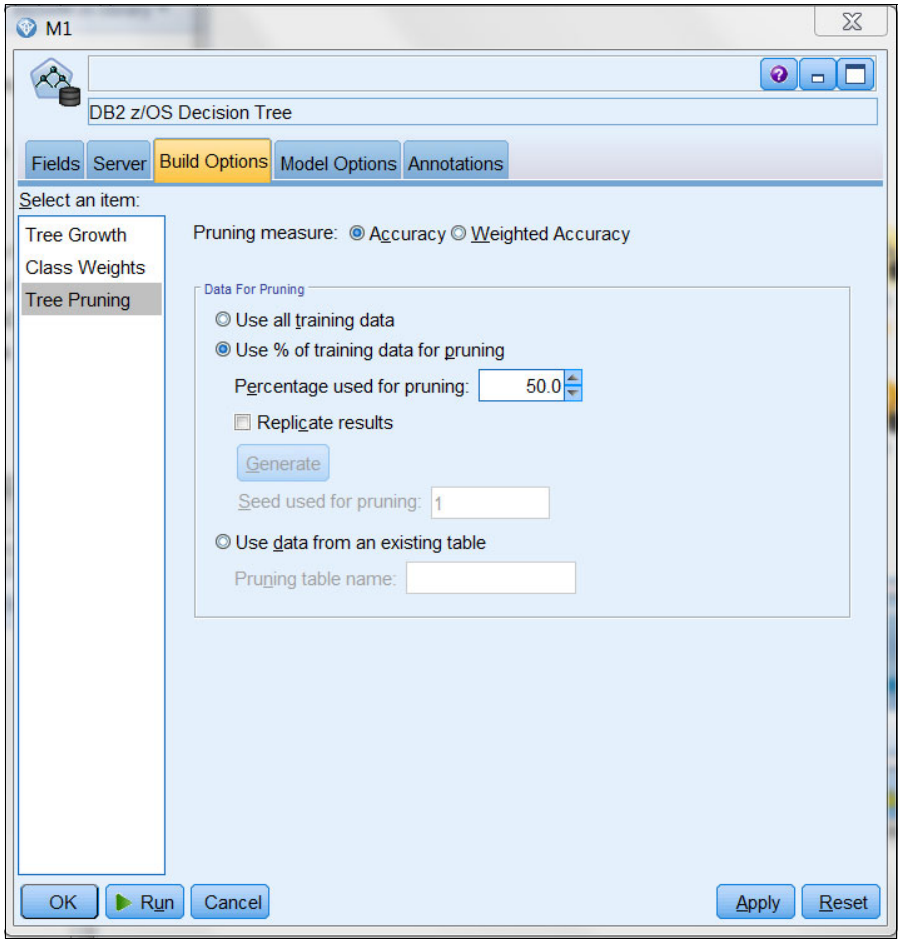


Figure 5-6 DB2 for z/OS Decision Tree build model options with tree pruning

This means that first, SPSS Modeler calls an IBM Netezza Analytics stored procedure to randomly split the accelerated input table into two accelerator-born tables containing different subsets of the original input:

- One subset is used for original training of the decision tree model.
- Another subset is used for model pruning.

The example stream (shown in Figure 5-7) uses split data and decision tree pruning.

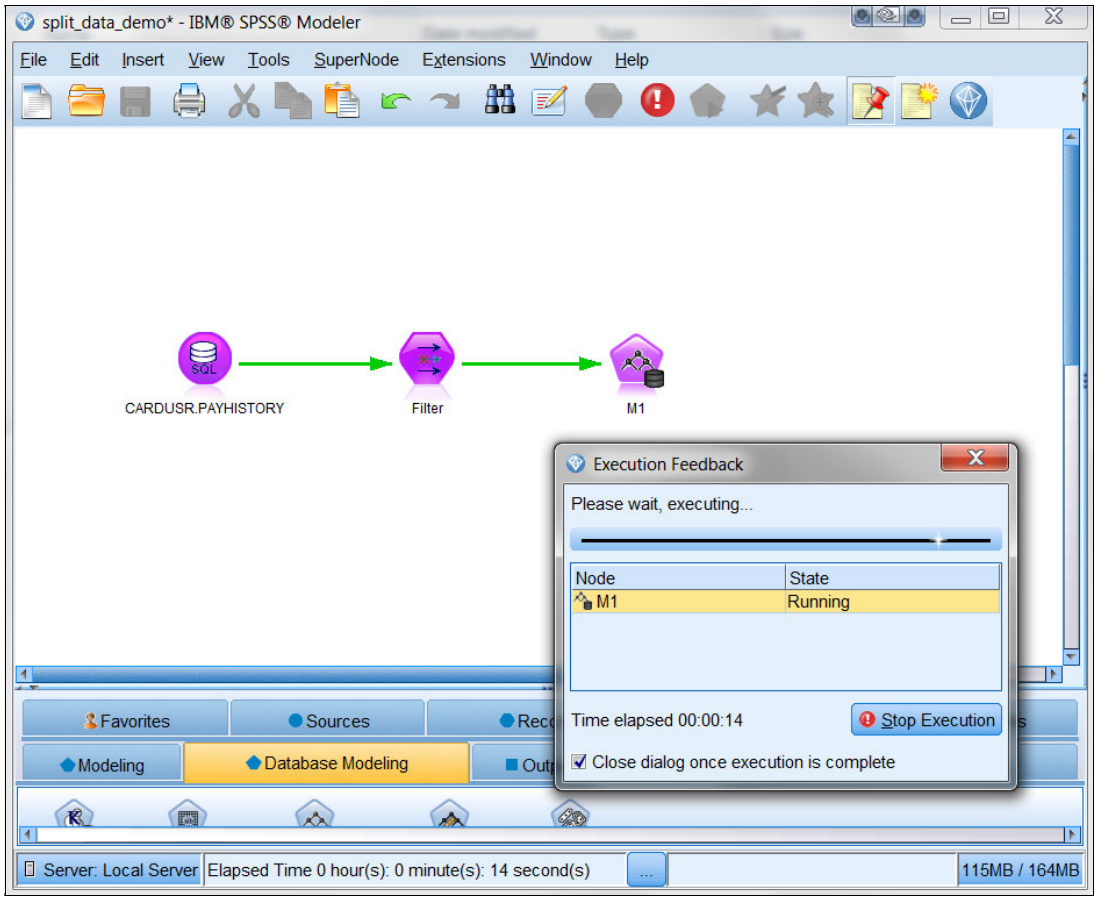


Figure 5-7 Simple SPSS stream using split data and decision tree pruning

If running this stream, the SQL (Figure 5-8) is generated by the SPSS Modeler.

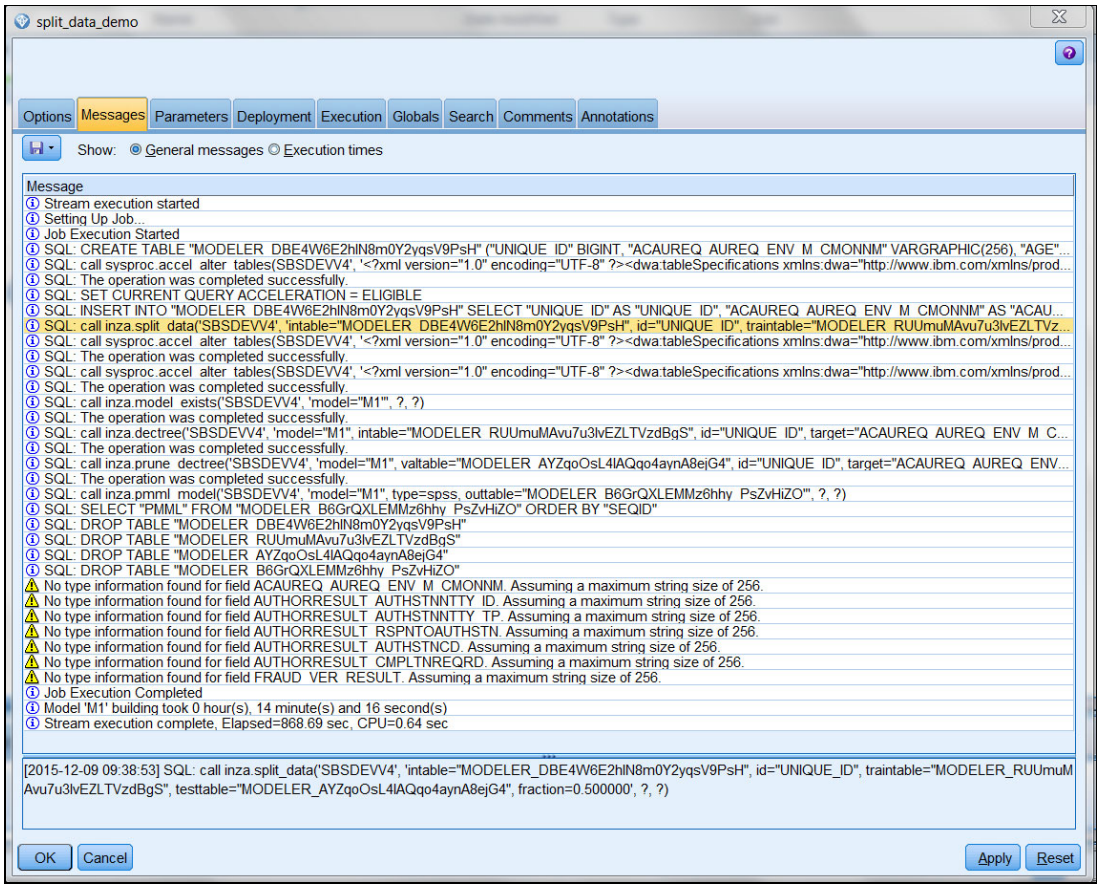


Figure 5-8 SQL generated by SPSS Modeler for the example stream

First, the example input table (*intable*) is split into two tables: *traintable* and *testable*.

Although the traintable is used by the `inza.dectree()` stored procedure, the table that is created as testable by `inza.split_data()` stored procedure is later consumed by the `inza.prune_dectree()` stored procedure as *valtable* for decision tree pruning, as shown in Figure 5-9.

```
[2015-12-09 09:45:57] SQL: call inza.prune_dectree('SBSDEVV4', 'model="M1", valtable="MODELER_AYZqoOsL4lAQqo4aynA8ejG4", id="UNIQUE_ID", target="ACAURE
Q_AUREQ_ENV_M_CMONNM", qmeasure=Acc', ?, ?)
```

Figure 5-9 Invocation of inza.prune_dectree() stored procedure

After everything finishes correctly, the model applier node is added to the stream. In it, the original decision tree model is replaced by the pruned decision tree model (Figure 5-10 on page 115). A *model applier node* defines a container for a generated model for use when the model is added to the IBM SPSS Modeler canvas from the Models tab of the manager pane.

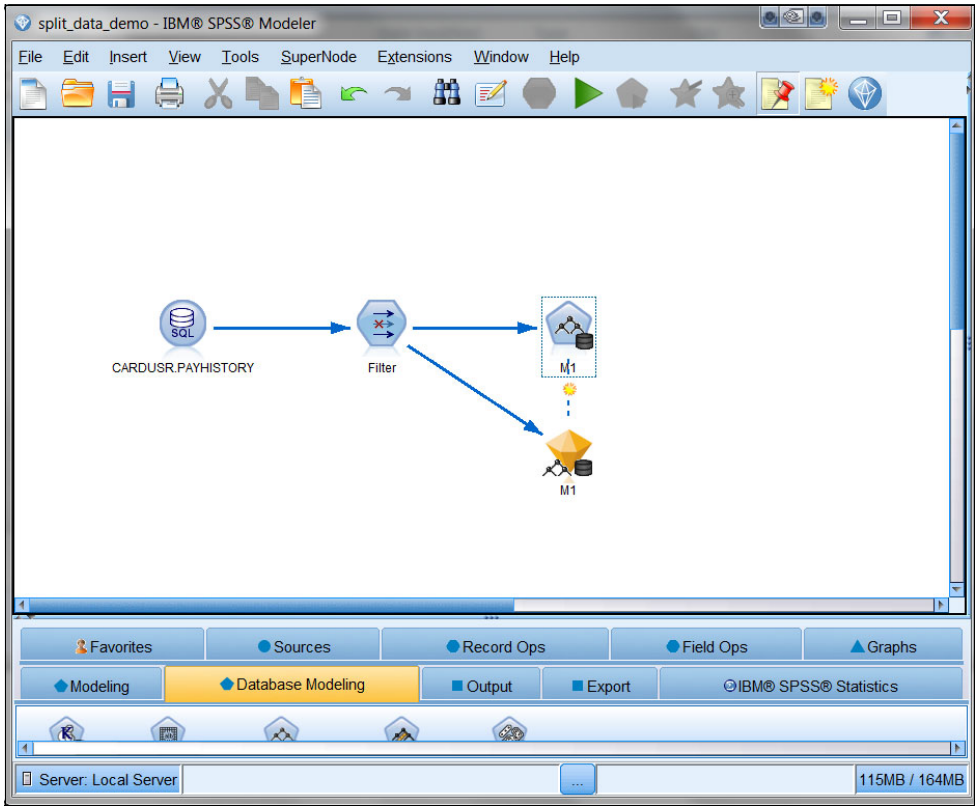


Figure 5-10 Stream with pruned decision tree model apply node

5.3 Adding record IDs to input tables

All IBM Netezza Analytics models require a a record ID, which must be unique. Ideally, you already have attributes in your data that are unique.

If you do not have any unique attributes in your input data, adding a unique attribute using the SPSS Modeler Derive node unfortunately does not perform well. The problem is that SPSS Modeler cannot push this down to the accelerated database but does that processing in the Modeler server. Then, it inserts the values into accelerated tables, using single-row inserts that do not perform well in DB2 Analytics Accelerator 4.x or DB2 Analytics Accelerator 5.1.

If you are able to add an attribute to the table, one possible solution is to add an attribute to the z/OS DB2 base table and then insert unique values based on a sequence, as shown in Example 5-1.

Example 5-1 Inserting unique values

```
\sampleBegin
ALTER TABLE myTab ADD COLUMN UNIQUE_ID BIGINT ;
CREATE SEQUENCE mySeq AS BIGINT START WITH 1 INCREMENT BY 1 CYCLE CACHE 1000 NO ORDER ;
UPDATE TESTOB SET UNIQUE_ID = NEXT VALUE FOR mySeq;
\sampleEnd
```

Remember to reload the extended table to the Accelerator after running these SQL commands.

5.4 Combining in-database transformation on Accelerator with traditional model creation

In many cases, the most significant resource consumption is in the data transformation and preparation step, where all of the input data is analyzed, aggregated, restructured and combined. The result of this transformation might go directly to a predictive modeling algorithm used to generate a new scoring model that can predict, for example, whether a transaction is fraud or whether a given person might be interested in some offer.

SPSS Modeler offers many more model creation algorithms than what is currently available based on IBM Netezza Analytics. The best approach might be to do everything within the Accelerator, so no additional data movement is required. However, combining in-database transformation with “traditional” modeling might make sense, especially if the Modeler server is running on Linux on z, connected using fast HiperSockets within the mainframe. At our customer proof of concept (POC) we have seen that many streams the customer used today are data intensive for the preparation, with actual customer pain points to work with all that data. However, at the point in time when the data scientist prepared the data for model creation, aggregated and reduced it to the principal components for modeling, the model itself on the SPSS Modeler was not the customer’s pain point. That only took minutes (however, as it only took minutes (however of course, this depends on SPSS Modeler server connectivity, amount of data to transfer and the used modeling node).

6

Model scoring

In predictive analytics, understanding the difference between model creation and model apply or model scoring is important.

In model creation, typically large parts of prepared data from various data sources are analyzed using data mining or machine learning algorithms in order to find patterns in that data. Training models on fraud is possible if enough data exists and if the attribute combinations that were fraud in the past are known. The algorithm might detect that the probability for fraud is high whenever a credit card is used to purchase jewelry just after a drink was purchased at a gas station. Or, that in order to properly segment customers with similar behavior, having the highest education level as an input parameter makes sense.

After the learning is done, a model is created that can then be deployed to the production system. Now for model apply or scoring, what is typically done is that the model function is applied on a new set of input parameters. If a new transaction is received, the score function for fraud detection will calculate the fraud risk based on the given input parameters. Or, if a new customer registers with a given company, that company might calculate the cluster of that customer based on data available at that time.

This chapter discusses model scoring with regards to in-transactional scoring, real-time scoring, near real-time scoring, and batch scoring. It provides a more in-depth discussion of batch scoring. This chapter contains the following topics:

- In-transactional scoring (single-record scoring)
- In-database scoring and scoring adapter
- Batch scoring
- DMG and PMML
- Scoring and smart decisions
- Examples of integration for scoring solutions using ODM on z/OS

It also has an overview of Operational Decision Manager (ODM) and SPSS Collaboration and Deployment Services (C&DS), which are higher-level offerings to better manage and automate all analytical assets and to associate business rules to the score results in order to better drive decisions in the operational environment.

© Copyright IBM Corp. 2016. All rights reserved.

6.1 In-transactional scoring (single-record scoring)

In-transactional scoring, which is also known as single-record scoring, means you score one record at a time. Having the invocation effort of the scoring function be minimized is important. Often architecture is implemented to establish a (distributed) server dedicated for scoring and to connect it using network-based interfaces like web services. In pseudocode, transaction code might look like Example 6-1.

Example 6-1 Pseudocode

```
select customer_name, product_name from tab1 inner join tab2 on ...;

invoke web service;
wait until the request has been transferred through the network and the
distributed box has queued and later on processed the request;
wait until the response from the distributed box finds its way back to your z/OS;
```

This has no advantages but significant disadvantages:

- The web services overhead on the z/OS side is typically higher than the execution of a complete score operation.
- Scalability is typically a problem, especially for high transaction rates.
- Network issues in distributed environments can bring down the mainframe transactional workload. So if the scoring user-defined function (UDF) is essential, overall availability of the business application is no longer the good number of the mainframe but just the number of the distributed computers.
- WLM cannot help, it cannot donate resources to distributed computers.
- Your data leaves z/OS control. A hacker can choose to attack either z/OS or your distributed computers if the hacker wants to attack the service in question.

For some models, creating SQL for the scoring is possible. Depending on the model, this can be a good or a bad alternative. For example, if the model is simple enough, it might be faster than the scoring adapter and still have all of the advantages from scoring in-database. If it contains many switch expressions in SQL, it might be slow in relation. And in some cases, you might not be able to get a score function in pure SQL.

Imagine you created a predictive model that takes a number of input parameters from your database you consolidated on z/OS and that is used for online transaction processing (OLTP). The database is free from redundancy (“single version of the truth”), and with atomicity, consistency, isolation, and durability (ACID) transactions operating on the data, DB2 ensures that new transactions always see the most current data (real real-time data). Now your predictive model can determine the probability that a given transaction is fraudulent.

For those requirements, a better alternative is if the score function is invoked from within the transaction code (which can be written in any language, but it must be able to access the database using an interface like ODBC, JDBC, CICS DB2 attachment facility, or whatever is most appropriate for you).

In pseudocode, the transaction might do something similar to Example 6-2.

Example 6-2 Pseudo-ode of score function invoked from within the transaction code

```
select customer_name, product_name, score_UDF( fraudModel, customer_number,
merchant_number) as fraud_probability from table inner join tab2 on ... ;
```

This example means you prefer to invoke your predictive model named `fraudModel`, with input parameters `customer_number` and `merchant_number` and an output value for fraud probability, plus additional parameters from the database. All this is running in the database, with the `score_UDF` function being a generic implementation which can score predictive models.

Deploying the model to your production OLTP system is triggered by a user that is capable of deploying it from perhaps SPSS Modeler to DB2. In the case of the scoring adapter, the models, which need to be highly available for business-critical transactional workloads, are stored in DB2 tables. Regarding existing backup, disaster recovery, security, and availability related procedures, nothing changes.

Although this example is oversimplified, it outlines the SPSS Modeler Scoring Adapter for z Systems platform. In reality, it uses several steps with the use of **`PACK`** and **`UNPACK`** commands for efficient parameter movement but in principle it works as outlined.

Another valid alternative is Zementis with in-application scoring, assuming you are running Java workload so you can invoke Zementis scoring functions (which are implemented by Zementis PMML scoring API in Java) from within your own application environment.

6.2 In-database scoring and scoring adapter

Real-time data is usually found in the OLTP system, in z/OS DB2. Most, if not all, transactions obtain input data from z/OS DB2. So for real-time in-transactional scoring, it works well if the database is able to provide the dynamically calculated score value in addition to the retrieved static values.

6.2.1 Real-time scoring and near real-time scoring

With SPSS Modeler Scoring Adapter for DB2 z/OS, you can score a Predictive Model Markup Language (PMML) model, which is deployed to the scoring adapter from SPSS Modeler client, using scoring UDFs.

With the scoring adapter, there is no need to move data in real-time (or near real-time) from z/OS to the system running the scoring engine. Actually, an easy approach is to use the latest available, real-time data available in DB2. In a performance study, we found that the invocation of a web service is typically more expensive (with regard to elapsed time and to consumed z/OS MSUs) than doing the score UDF including the actual scoring function within z/OS altogether. Predictive model creation is resource consuming, but doing an individual model score is not if compared to the overhead of invoking a standard-based web service call. In addition, getting a distributed scoring engine that can scale is challenging. If using the scoring adapter score UDFs, scaling occurs with DB2 so that achieving a WLM-enforced SLAs becomes easier.

To publish the model to z/OS DB2, complete the following steps:

1. double-click the model nugget icon and click the **File** tab. On the **file** tab page, select **Publish for Server Scoring Adapter** as shown in Figure 6-1.

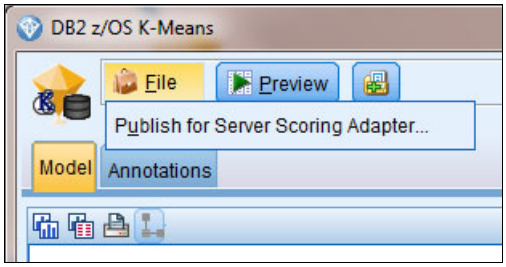


Figure 6-1 Publish for Server Scoring Adapter

2. Enter the required connection information, as shown by the example in Figure 6-2.

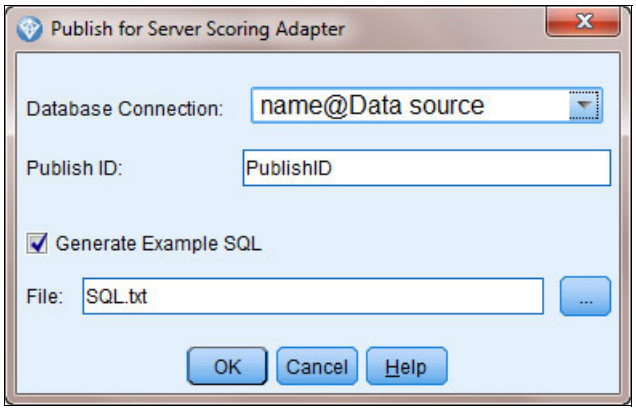


Figure 6-2 Publish for Server Scoring Adapter connection information

Create a text file containing SQL code by selecting the **Generate Example SQL** check box. Provide a file name, and click **OK**.

An example of the SQL code generated from this stream is provided in Example 6-3.

Example 6-3 Sample generated SQL code

```
SELECT TO."UNIQUE_ID" AS "UNIQUE_ID", TO."ACAUREQ_HDR_CREDTTM" AS
"ACAUREQ_HDR_CREDTTM", TO."ACAUREQ_HDR_RCPTPTY_TP" AS
"ACAUREQ_HDR_RCPTPTY_TP", TO."ACAUREQ_AUREQ_ENV_CPL_XPRYDT" AS
"ACAUREQ_AUREQ_ENV_CPL_XPRYDT", TO.AGE AS AGE, UNPACK
(HUMSPSS.SCORE_COMPONENT('P','PlayKMeans',PACK(CCSID
208,TO."UNIQUE_ID",TO."ACAUREQ_HDR_CREDTTM",TO."ACAUREQ_HDR_RCPTPTY_TP
",TO."ACAUREQ_AUREQ_ENV_CPL_XPRYDT",TO.AGE))).* AS (C5 VARCHAR(1024)
CCSID UNICODE, C6 DOUBLE) FROM (SELECT TO."UNIQUE_ID" AS
"UNIQUE_ID",TO."ACAUREQ_HDR_CREDTTM" AS
"ACAUREQ_HDR_CREDTTM",TO."ACAUREQ_HDR_RCPTPTY_TP" AS
"ACAUREQ_HDR_RCPTPTY_TP",TO."ACAUREQ_AUREQ_ENV_CPL_XPRYDT" AS
"ACAUREQ_AUREQ_ENV_CPL_XPRYDT",TO.AGE AS AGE FROM ${TABLEO} TO) AS TO
```

The highlighted text is the SQL code that can be incorporated in OLTP application code used to score transactions in real time.

After the model is published to DB2, the PublishID entered in the panel shown in Figure 6-2 on page 120 is stored in the PUBLISHED_COMPONENTS table and the actual model is stored in the COMPONENTS table. These two tables are then used by the UDF to perform scoring.

Figure 6-3 shows the messages displayed while the data modeling stream is executing.

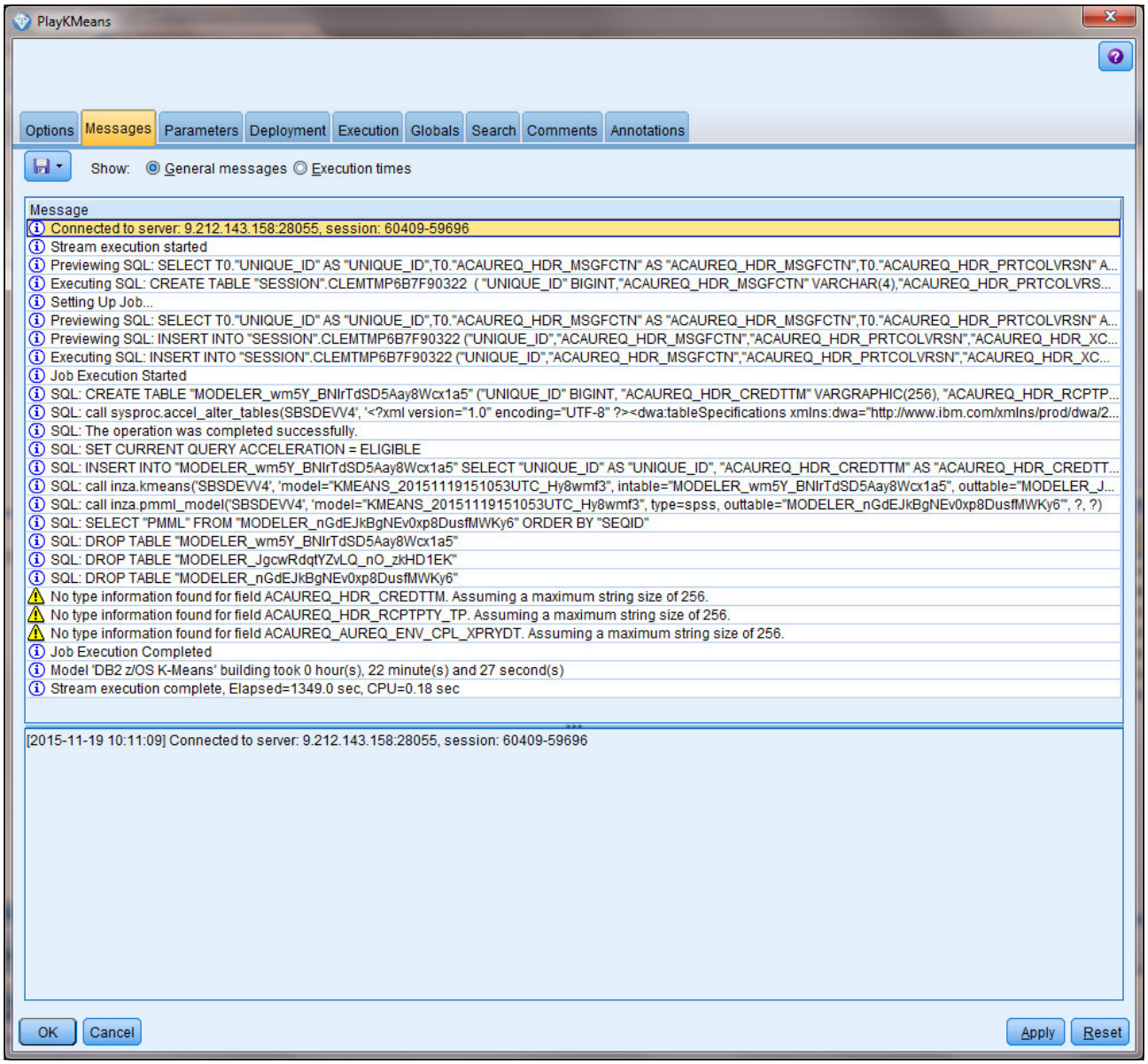


Figure 6-3 Data modeling stream messages

6.2.2 Export PMML from SPSS model apply node

After a new model is created, you can either publish it to the SPSS Modeler Scoring Adapter for DB2 z/OS or export it in standardized PMML format. If you want to get the PMML, double-click the model nugget icon and click the **File** tab. On the File tab, select **Export PMML** as shown in Figure 6-4.

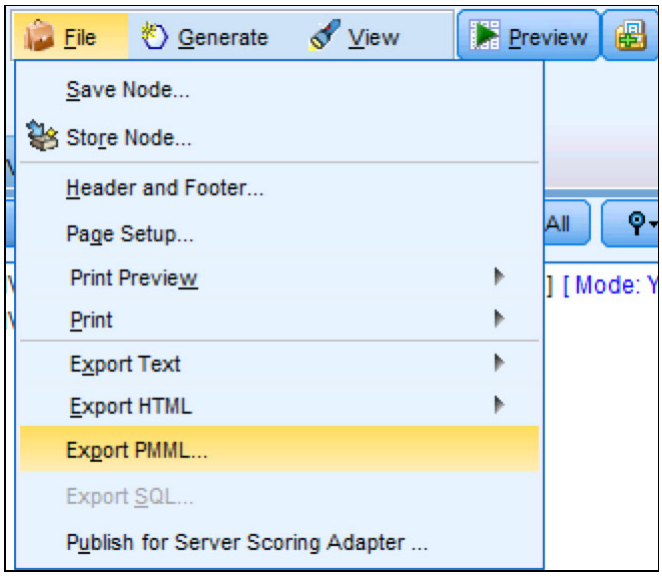


Figure 6-4 Export PMML

As explained previously, this PMML can now be used by other vendor tools, such as Zementis, which offers in-application scoring for SPSS generated models and for models generated by SAS Enterprise Miner or other tools. For in-transactional scoring, in-application scoring can show similar advantages like in-database scoring.

Alternatively, consider the generated PMML, which is a well-defined XML format. Example 6-4 is from one of our streams.

Example 6-4 Sample generated PMML

```
<?xml version="1.0" encoding="UTF-8"?><Parameters debug="false" version="8.0"
xmlns="http://xml.spss.com/pasw/extension"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xml.spss.com/pasw/extension
http://xml.spss.com/pasw/extension/parameters.xsd">
<Parameter name="scoring.kmLabelPrefix" value=""/>
<Parameter name="scoring.time_interval" value="1.0"/>
<Parameter name="scoring.adSupressNonAnomalousRecords" value="false"/>
<Parameter name="scoring.resultType">
<StructuredValue>
<Attribute name="prediction" value=""/>
<Attribute name="probability" value="P"/>
<Attribute name="probcategory" value="P"/>
<Attribute name="confidence" value="C"/>
<Attribute name="stddev" value="S"/>
<Attribute name="clustering-kohonen-x" value="X"/>
<Attribute name="clustering-kohonen-y" value="Y"/>
<Attribute name="clustering-kohonen-xy" value="XY"/>
<Attribute name="clustering-kmeans-distance" value="D"/>
<Attribute name="association_confidence" value="C"/>
```

```
<Attribute name="selflearning_confidence" value="C"/>
</StructuredValue>
</Parameter>
<Parameter name="scoring.confidenceBasedOn" value="onProbability"/>
<Parameter name="scoring.labelPattern">
<StructuredValue>
<Attribute name="prediction" value="kSCORING_PredictedValue_0"/>
<Attribute name="probability" value="kSCORING_UndefinedLabel_0"/>
<Attribute name="probcategory" value="kSCORING_Probability_2"/>
<Attribute name="confidence" value="kSCORING_ConfidencePP_0"/>
<Attribute name="stddev" value="kSCORING_UndefinedLabel_0"/>
<Attribute name="tree-nodeid" value="kSCORING_UndefinedLabel_0"/>
<Attribute name="cox-cumhazard" value="kSCORING_UndefinedLabel_0"/>
<Attribute name="knn-caseid" value="kSCORING_UndefinedLabel_0"/>
<Attribute name="knn-distance" value="kSCORING_UndefinedLabel_0"/>
<Attribute name="knn-caselabel" value="kSCORING_UndefinedLabel_0"/>
<Attribute name="adjustedpropensity" value="kSCORING_UndefinedLabel_0"/>
<Attribute name="rawpropensity" value="kSCORING_UndefinedLabel_0"/>
</StructuredValue>
</Parameter>
<Parameter name="scoring.resolveNameConflict" value="true"/>
<Parameter name="scoring.isRawPropensity" value="false"/>
<Parameter name="scoring.assocCheckBasket" value="isOutsideBasket"/>
<Parameter name="scoring.time_field" value=""/>
<Parameter name="scoring.isStandardDeviation" value="false"/>
<Parameter name="scoring.encoding" value="UTF-8"/>
<Parameter name="scoring.selflearningTargetFields">
<ListValue/>
</Parameter>
<Parameter name="scoring.selflearningRandomization" value="0.0"/>
<Parameter name="scoring.isEnsemble" value="true"/>
<Parameter name="scoring.isPrediction" value="true"/>
<Parameter name="scoring.assocIsOnlyOutputValue" value="false"/>
<Parameter name="scoring.scoreValue" value=""/>
<Parameter name="scoring.predicateList">
<StructuredValue>
<Attribute name="closeTo" value="kSCORING_Predicate_CloseTo"/>
<Attribute name="crosses" value="kSCORING_Predicate_Crosses"/>
<Attribute name="intersect" value="kSCORING_Predicate_Intersect"/>
<Attribute name="touch" value="kSCORING_Predicate_Touch"/>
<Attribute name="overlap" value="kSCORING_Predicate_Overlap"/>
<Attribute name="within" value="kSCORING_Predicate_Within"/>
<Attribute name="contain" value="kSCORING_Predicate_Contain"/>
<Attribute name="northOf" value="kSCORING_Predicate_NorthOf"/>
<Attribute name="southOf" value="kSCORING_Predicate_SouthOf"/>
<Attribute name="eastOf" value="kSCORING_Predicate_EastOf"/>
<Attribute name="westOf" value="kSCORING_Predicate_WestOf"/>
</StructuredValue>
</Parameter>
<Parameter name="scoring.isStandardDeviationRequired" value="false"/>
<Parameter name="scoring.isNodeID" value="false"/>
<Parameter name="scoring.assocTransactionalID" value=""/>
<Parameter name="pasw.annotation_toolTipText" value=""/>
<Parameter name="scoring.isPropensity" value="false"/>
<Parameter name="scoring.assocIsOnlyTrueForFlag" value="true"/>
<Parameter name="scoring.isOutputIntNodeId" value="false"/>
<Parameter name="pasw.annotation_keywords" value=""/>
<Parameter name="scoring.modelDll">
```

```
<StructuredValue>
<Attribute name="TreeModel" value="TreeModel"/>
<Attribute name="discriminant-model" value="discriminant-model"/>
<Attribute name="GeneralRegressionModel" value="GeneralRegressionModel"/>
<Attribute name="ClusteringModel" value="ClusteringModel"/>
<Attribute name="NeuralNetwork" value="NeuralNetwork"/>
<Attribute name="AssociationModelLattice" value="AssociationModelLattice"/>
<Attribute name="SeqAssociationLattice" value="SeqAssociationLattice"/>
<Attribute name="NaiveBayesModel" value="NaiveBayesModel"/>
<Attribute name="AssociationModel" value="AssociationModel"/>
<Attribute name="RuleSetModel" value="RuleSetModel"/>
<Attribute name="BayesNetModel" value="BayesNetModel"/>
<Attribute name="SupportVectorMachineModel" value="SupportVectorMachineModel"/>
<Attribute name="NearestNeighborModel" value="NearestNeighborModel"/>
<Attribute name="MiningModel" value="MiningModel"/>
<Attribute name="RegressionModel" value="RegressionModel"/>
<Attribute name="TimeSeriesModel" value="TimeSeriesModel"/>
</StructuredValue>
</Parameter>
<Parameter name="scoring.selflearningRandomSeed" value="876547"/>
<Parameter name="scoring.nameList">
<StructuredValue>
<Attribute name="prediction" value=""/>
<Attribute name="probability" value=""/>
<Attribute name="probcategory">
<ListValue/>
</Attribute>
<Attribute name="confidence" value=""/>
<Attribute name="stddev" value=""/>
<Attribute name="tree-nodeid" value=""/>
<Attribute name="cox-cumhazard" value=""/>
<Attribute name="knn-caseid">
<ListValue/>
</Attribute>
<Attribute name="knn-distance">
<ListValue/>
</Attribute>
<Attribute name="knn-caselabel">
<ListValue/>
</Attribute>
<Attribute name="adjustedpropensity" value=""/>
<Attribute name="rawpropensity" value=""/>
</StructuredValue>
</Parameter>
<Parameter name="scoring.assocRepeatPredications" value="false"/>
<Parameter name="scoring.maxCategories" value="25"/>
<Parameter name="scoring.adScoreMethod" value="FLAG_AND_SCORE"/>
<Parameter name="scoring.enableSQL" value="false"/>
<Parameter name="scoring.isRuleID" value="false"/>
<Parameter name="scoring.selflearningReliability" value="true"/>
<Parameter name="scoring.cumulative_hazard" value="false"/>
<Parameter name="scoring.isConfidence" value="true"/>
<Parameter name="scoring.maxPredictions" value="3"/>
<Parameter name="pasw.annotation_description" value=""/>
<Parameter name="scoring.isOutputInputData" value="true"/>
<Parameter name="scoring.past_survival_time" value=""/>
<Parameter name="scoring.scoringDll">
<StructuredValue>
```

```
<Attribute name="SmartScore" value="mcscorersmart"/>
<Attribute name="NeuralNetwork" value="mcscorernnet"/>
<Attribute name="MiningModel" value="mcscorermining"/>
</StructuredValue>
</Parameter>
<Parameter name="scoring.useNameList" value="false"/>
<Parameter name="scoring.adNumAnomalyFields" value="3"/>
<Parameter name="scoring.isBaseModels" value="false"/>
<Parameter name="scoring.modelID">
<StructuredValue>
<Attribute name="TreeModel" value="R"/>
<Attribute name="discriminant-model" value="discriminant-model"/>
<Attribute name="GeneralRegressionModel" value="L"/>
<Attribute name="ClusteringModel" value="ClusteringModel"/>
<Attribute name="NeuralNetwork" value="N"/>
<Attribute name="AssociationModelLattice" value="AssociationModelLattice"/>
<Attribute name="SeqAssociationLattice" value="SeqAssociationLattice"/>
<Attribute name="NaiveBayesModel" value="NaiveBayesModel"/>
<Attribute name="AssociationModel" value="A"/>
<Attribute name="RuleSetModel" value="RuleSetModel"/>
<Attribute name="BayesNetModel" value="B"/>
<Attribute name="SelflearningModel" value="S"/>
<Attribute name="SupportVectorMachineModel" value="S"/>
<Attribute name="NearestNeighborModel" value="KNN"/>
<Attribute name="MiningModel" value="MiningModel"/>
<Attribute name="CLUSTER_KOHONEN" value="K"/>
<Attribute name="ANOMALYDETECTION" value="O"/>
<Attribute name="CLUSTER_KMEANS" value="KM"/>
<Attribute name="RULESET_DL" value="D"/>
<Attribute name="GZLR" value="G"/>
<Attribute name="LR" value="E"/>
<Attribute name="RegressionModel" value="RegressionModel"/>
<Attribute name="TimeSeriesModel" value="TS"/>
<Attribute name="SELF_LEARNING" value="S"/>
<Attribute name="CLUSTER_TWOSTEP" value="T"/>
</StructuredValue>
</Parameter>
<Parameter name="scoring.isDistance" value="false"/>
<Parameter name="scoring.assocRuleCriterion" value="confidence"/>
<Parameter name="scoring.isProbability" value="false"/>
<Parameter name="pasw.annotation_autoLabel" value="true"/>
<Parameter name="scoring.future_time_as" value="None"/>
<Parameter name="scoring.assocIsOutputOrigInterval" value="false"/>
<Parameter name="scoring.namePattern">
<StructuredValue>
<Attribute name="prediction" value="${M}{S}-{T}"/>
<Attribute name="probability" value="${M}{R}{S}-{T}"/>
<Attribute name="probcategory" value="${M}{R}{S}-{C}"/>
<Attribute name="confidence" value="${M}{R}{S}-{T}"/>
<Attribute name="stddev" value="${M}{R}-{T}"/>
<Attribute name="tree-nodeid" value="NodeID"/>
<Attribute name="cox-cumhazard" value="CumHazard"/>
<Attribute name="knn-caseid" value="${M}{S}-neighbor-{G}"/>
<Attribute name="knn-distance" value="${M}{S}-distance-{G}"/>
<Attribute name="knn-caselabel" value="${M}{S}-neighbor-{G}"/>
<Attribute name="adjustedpropensity" value="$AP{S}-{T}"/>
<Attribute name="rawpropensity" value="$RP{S}-{T}"/>
<Attribute name="clustering-clusterid" value="clustering-clusterid"/>
```

```
<Attribute name="clustering-kohonen-x" value="${M}{R}{S}-Kohonen"/>
<Attribute name="clustering-kohonen-y" value="${M}{R}{S}-Kohonen"/>
<Attribute name="clustering-kohonen-xy" value="${M}{R}{S}-Kohonen"/>
<Attribute name="anomaly-flag" value="${M}{S}-Anomaly"/>
<Attribute name="anomaly-index" value="${M}{S}-AnomalyIndex"/>
<Attribute name="anomaly-clusterid" value="${M}{S}-PeerGroup"/>
<Attribute name="anomaly-varname" value="${M}{S}-Field-{G}"/>
<Attribute name="anomaly-avrimpact" value="${M}{S}-FieldImpact-{G}"/>
<Attribute name="clustering-kmeans-clusterid" value="${M}{S}-{N}"/>
<Attribute name="clustering-kmeans-distance" value="${M}{R}{S}-{N}"/>
<Attribute name="association_prediction" value="${M}{S}-{T}-{G}"/>
<Attribute name="association_confidence" value="${M}{R}{S}-{T}-{G}"/>
<Attribute name="association_ruleid" value="${M}{S}-Rule_ID-{G}"/>
<Attribute name="selflearning_prediction" value="${M}{S}-{T}-{G}"/>
<Attribute name="selflearning_confidence" value="${M}{R}{S}-{T}-{G}"/>
<Attribute name="ruleset-dl-probability" value="${M}P{S}-{T}"/>
<Attribute name="ruleset-firstid" value="${M}I{S}-{T}"/>
<Attribute name="timeseries-lowerconfidence" value="${M}LCI{S}-{T}"/>
<Attribute name="timeseries-upperconfidence" value="${M}UCI{S}-{T}"/>
<Attribute name="timeseries-noiseresiduals" value="${M}NR{S}-{T}"/>
</StructuredValue>
</Parameter>
<Parameter name="scoring.isKohonenY" value="true"/>
<Parameter name="scoring.selflearningSort" value="descending"/>
<Parameter name="scoring.isKohonenX" value="true"/>
<Parameter name="scoring.isCumHazard" value="false"/>
<Parameter name="scoring.assocNameTable">
<ListValue/>
</Parameter>
<Parameter name="scoring.isOthers" value="false"/>
<Parameter name="scoring.missingValuePolicy" value="asMissingPredictor"/>
<Parameter name="pasw.annotation_lastSavedBy" value="oliver.benke"/>
<Parameter name="pasw.annotation_id" value="id3K7RSNB1Z5N"/>
<Parameter name="scoring.num_future_times" value="1"/>
<Parameter name="scoring.assocEqualsBasket" value="false"/>
<Parameter name="scoring.isOutputAll" value="false"/>
<Parameter name="scoring.isProbabilityCategory" value="false"/>
<Parameter name="scoring.isCaseLabel" value="false"/>
<Parameter name="scoring.adAnomalyIndexCutoff" value="0.0"/>
<Parameter name="scoring.isReferenceModel" value="false"/>
<Parameter name="scoring.versionSupported" value="4.0"/>
<Parameter name="scoring.isNeighbor" value="false"/>
<Parameter name="pasw.annotation_lastSavedDate" value="1.44854280264E9"/>
<Parameter name="pasw.annotation_label" value=""/>
<Parameter name="override_isDistance" value="true"/>
<Parameter name="scoring.all_probabilities" value="true"/>
</Parameters>
```

6.3 Batch scoring

Batch scoring means scoring complete bulks of data at once. Therefore, be sure that the processing effort per score value is minimized, and that data transfer (if required) is optimized or bulked.

6.3.1 Complete batch scoring example with accelerated in-database predictive modeling

Figure 6-5 shows that SPSS Modeler generates a new K-Means predictive model in-database, in the accelerator. After the model is created in the accelerator (see the `call inza.kmeans()` message in line one of the SPSS SQL messages shown in Figure 6-5), the accelerator retrieves the PMML model through z/OS DB2. After that, all temporary artifacts are dropped by SPSS Modeler on the accelerator, so the data scientist is not required to take care of any house keeping or cleanup activity.

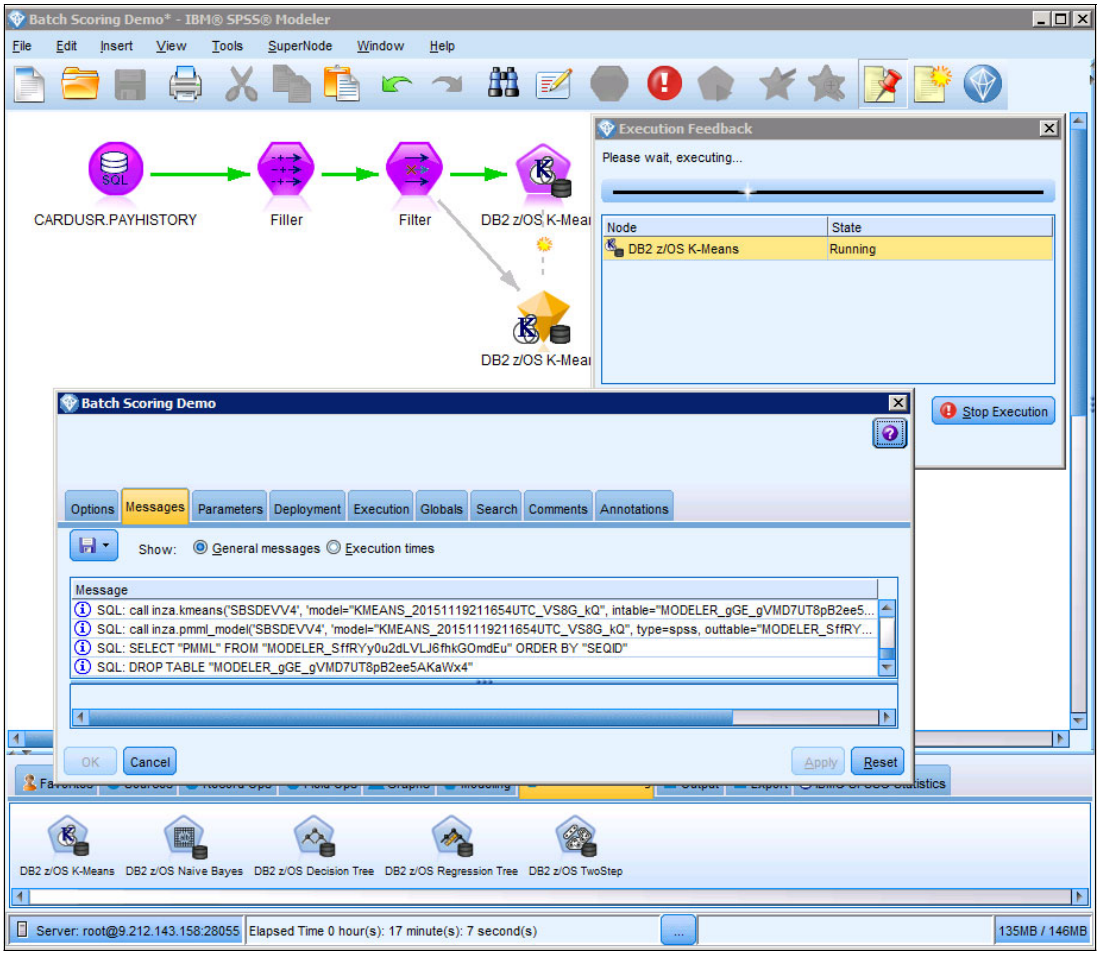


Figure 6-5 K-Means model create, using a source table on the Accelerator and running on the Accelerator

6.3.2 Batch scoring using Scoring Adapter into accelerator-only tables (less efficient method)

For SPSS Modeler to now do batch scoring, it must first add an output table to the SPSS Modeler Scoring Adapter, it must first add an output table to it. The user adds a new accelerator-only table to the stream, with the intention that this table stores the scored results.

First, create a new Export node against Accelerator database connection: DW01ACCL in this installation (see Figure 6-6).

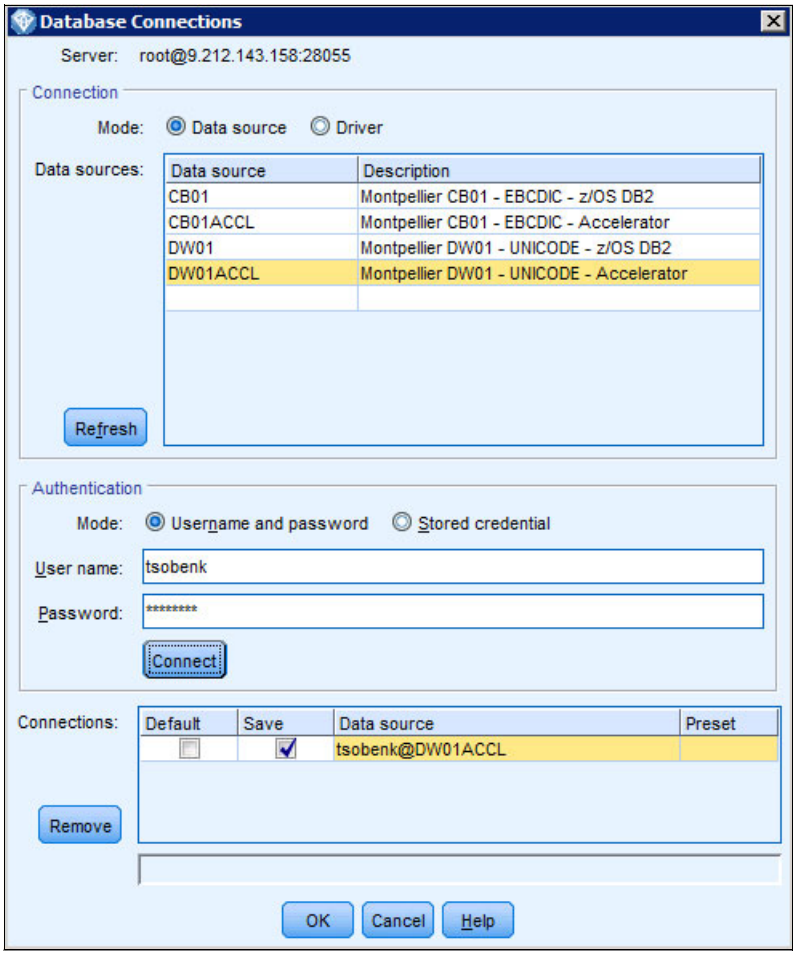


Figure 6-6 Select a database source with or without accelerator (DB2 base table or accelerator-only table)

You can choose a name for your new accelerator-only table in the dialog shown in Figure 6-7.

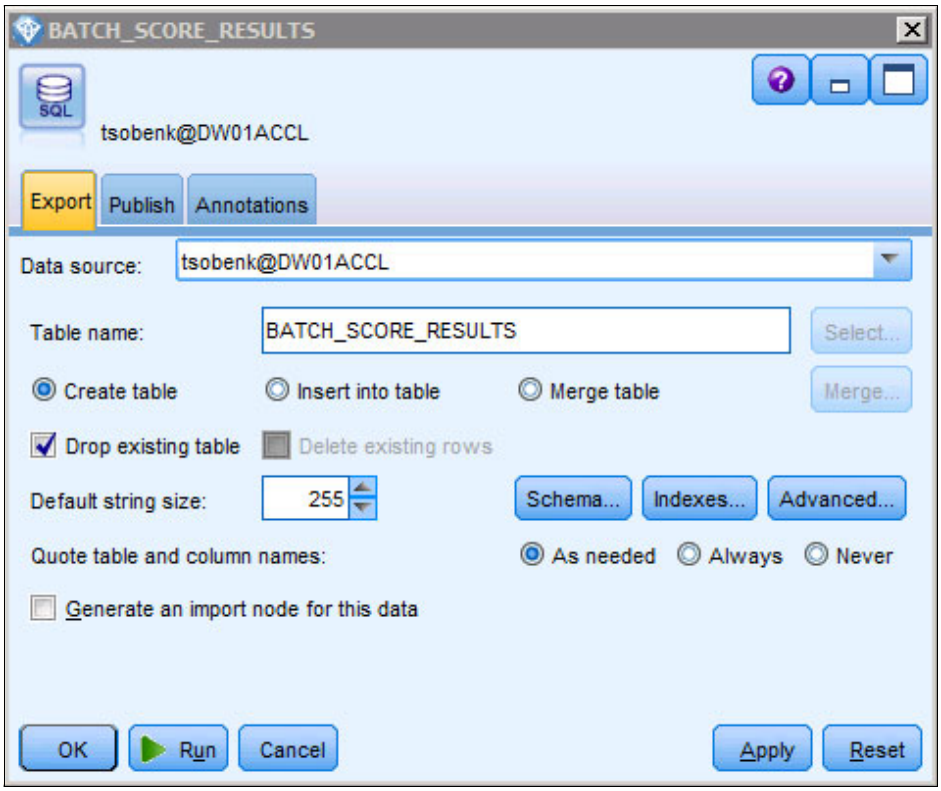


Figure 6-7 Settings of Database Export node in SPSS Modeler Export Database node

Select the **Drop existing table** so that with every new run of the batch scoring stream, the table is completely replaced. Because SPSS Modeler with DB2 Analytics Accelerator is case-sensitive with table names, the best approach is to use all uppercase letters for the table name.

While running the stream, SPSS Modeler changes the color of the icons to purple to indicate everything runs in-database, as shown in Figure 6-8.

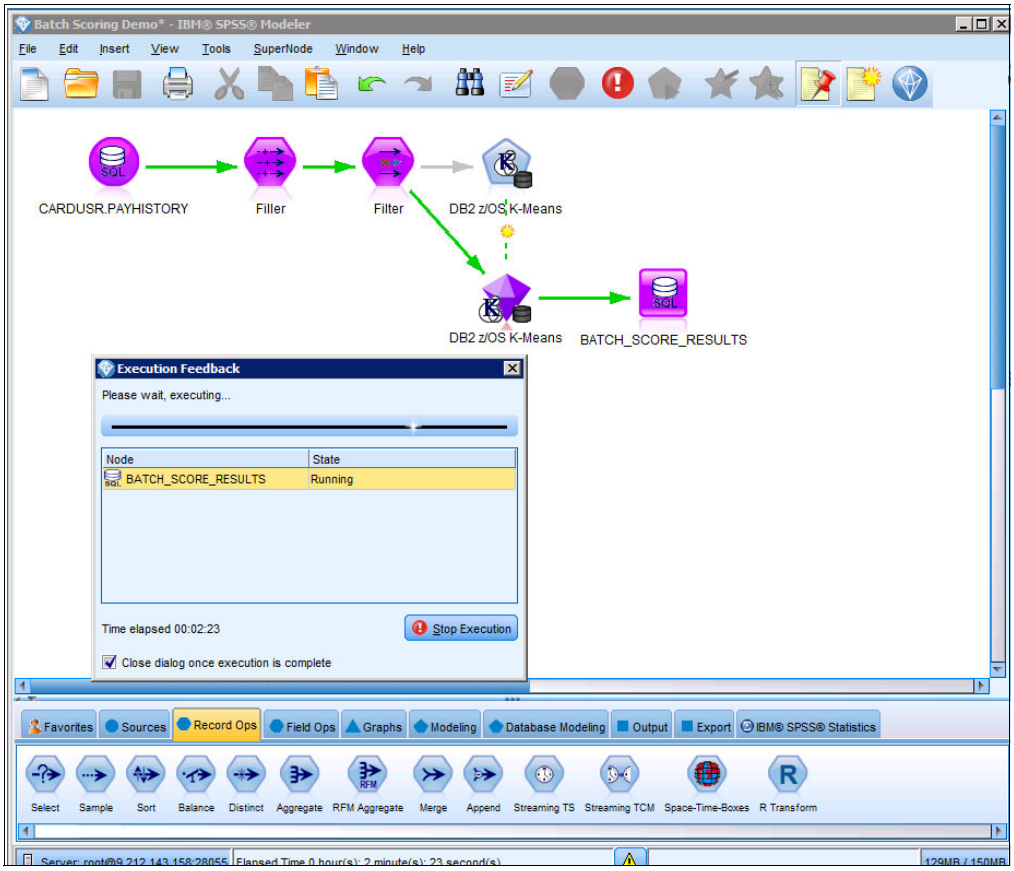


Figure 6-8 Batch scoring with accelerated tables as source and accelerator-only table as target (not recommended)

Using an accelerator-only table for batch scoring results is not a good idea, as described subsequently. First, consider what happens next.

After this stream is executed (by selecting the **Export Database** node and then clicking **Run selection**), SPSS Modeler (shown in Figure 6-9 on page 131) does the following steps:

1. The Export Database node that contains the output table to be dropped and re-created is selected. First it drops and re-creates the `BATCH_SCORE_RESULTS` table by using the following SQL:

   ```
   DROP TABLE "BATCH_SCORE_RESULTS"
      CREATE TABLE "BATCH_SCORE_RESULTS" ( "UNIQUE_ID" BIGINT,
   "ACAUREQ_HDR_CREDTTM" VARCHAR(255),  ..., "$KM-KMEANS_20151119211654UTC_V"
   VARCHAR(255), "$KMD-KMEANS_20151119211654UTC_" DOUBLE ) IN ACCELERATOR
   "SBSDEVV4" CCSID UNICODE
   ```

   The score result table contains all attributes used as input for the model create and for the model apply node, plus generated attributes for the following score results:

   ```
   "$KM-KMEANS_20151119211654UTC_V"
   "$KMD-KMEANS_20151119211654UTC_"
   ```

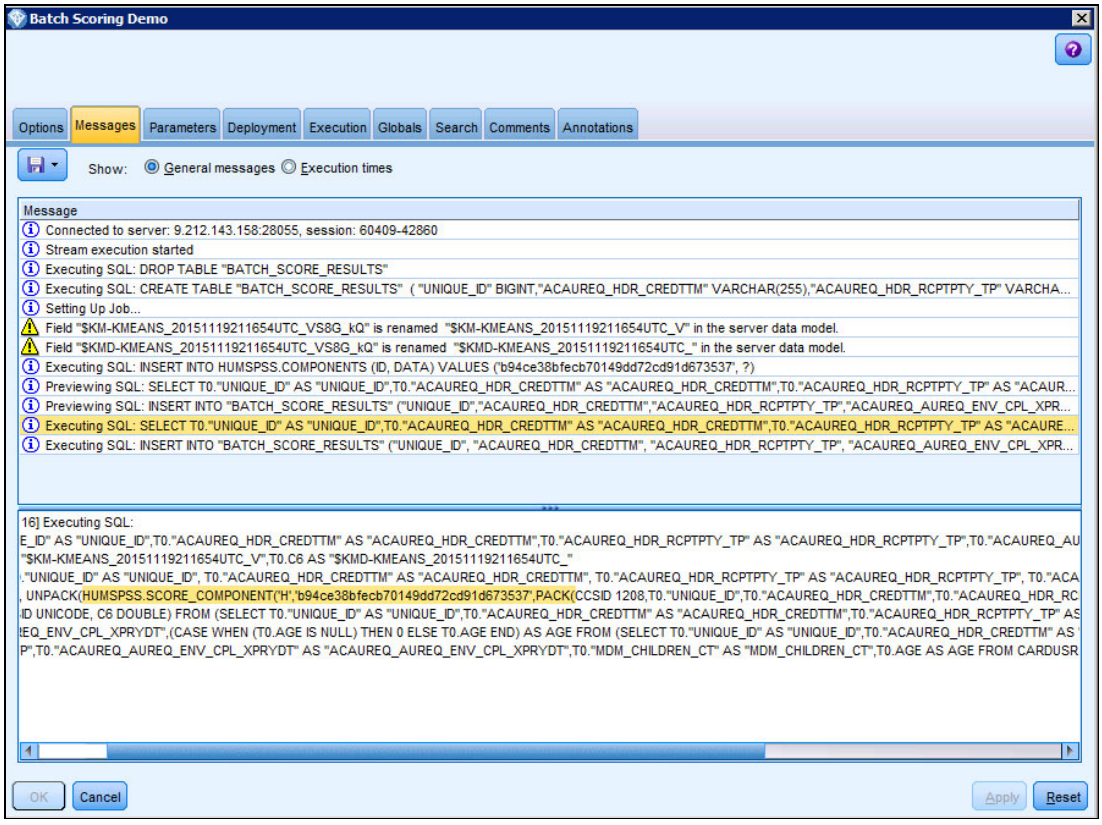


Figure 6-9 Batch scoring to accelerator-only table: first retrieves all score results using DB2 Scoring Adapter on DB2 z/OS, then single-row insert into accelerator-only table

2. If the SPSS Scoring Adapter is installed and enabled but the model is not yet published to the Scoring Adapter, the SPSS Modeler Client first automatically stores the PMML model to the Scoring Adapter on DB2 for z/OS by inserting the model as a BLOB to the Scoring Adapter tables. In the SPSS Modeler client SQL Messages, if the SPSS Modeler automatically publishes a model required for batch scoring to the scoring adapter, a message similar to the following example is displayed:

   ```
   INSERT INTO HUMSPSS.COMPONENTS (ID, DATA) VALUES
   ("B94CE38BFECB70149DD72CD91D673537", ?)
   ```

3. The SPSS Modeler usually tries the SQL first to verify how it works. In such cases, you can see `Previewing SQL` at the beginning of the SQL in question. If the database management system (DBMS) being used indicates that the SQL works, it will issue the same SQL. This then appears in the messages as a line prefixed with `Executing SQL`.

   Note: SPSS Modeler sometimes tries several variations of SQL before “giving up.” SPSS knows which database is connected, so it usually starts with SQL suitable for DB2 for z/OS but it might try alternatives if it sees an error from the DBMS. With regard to “giving up,” if SPSS cannot generate SQL, which is accepted by the database for whatever reason (such as reaching an edge-case restriction on a special data type in the table in use), it usually fails back to enumerating the input tables and doing all complex processing within SPSS Modeler server.

4. The following SQL invokes the SPSS Scoring Adapter:

```
SELECT T0."UNIQUE_ID" AS "UNIQUE_ID", T0." ACAUREQ_HDR_CREDTTM " AS
."ACAUREQ_HDR_CREDTTM", …, T0.C5 AS "$KM-KMEANS_20151119211654UTC_V",  T0.C6 AS
"$KMD-KMEANS_20151119211654UTC_"
 FROM (SELECT T0."UNIQUE_ID" AS "UNIQUE_ID", T0."ACAUREQ_HDR_CREDTTM" AS
."ACAUREQ_HDR_CREDTTM", …,, UNPACK(HUMSPSS.SCORE_COMPONENT('H',
'9f79359f6887124bbb54ce1009941d8a', PACK(CCSID 1208, T0."UNIQUE_ID", T0.
."ACAUREQ_HDR_CREDTTM", …))).* AS (C5 VARCHAR(1024) CCSID UNICODE, C6 DOUBLE)
FROM (SELECT T0."UNIQUE_ID" AS "UNIQUE_ID", T0."ACAUREQ_HDR_CREDTTM" AS
."ACAUREQ_HDR_CREDTTM", …, FROM CARDUSR.PAYHISTORY T0) AS T0) T0
```

   The score UDF function is HUMSPSS.SCORE_COMPONENT. The PACK and UNPACK functions, were introduced in APAR PM56631 for DB2 V10 for z/OS and were added to DB2 to allow for more efficient parameter transfer to the score UDF. The score UDF is a generic C++ implementation; it checks the HUMSPSS tables to determine if the model component is registered.

   The generically returned score values are converted using the UNPACK function to C5 and C6 values.

5. In doing it this way, SPSS Modeler must insert all results into the accelerator-only table. To better understand what SPSS Modeler has done out of the stream created, look at the SQL code. It shows that SPSS inserts all score values retrieved into the accelerator-only table as shown in the SPSS Modeler SQL log:

```
INSERT INTO "BATCH_SCORE_RESULTS" ("UNIQUE_ID", "ACAUREQ_HDR_CREDTTM", … ,
"$KM-KMEANS_20151119211654UTC_V", "$KMD-KMEANS_20151119211654UTC_") VALUES (?.
?. ?. ?. ?. ?. ?)
```

   Because multi-row insert for accelerator-only tables is not yet implemented in DB2 for z/OS, SPSS Modeler must use single-row insert for that. This works if you only want to score something like 100 scores, but it already took more than 2 minutes to score 1000 scores, with almost all of that time spent in this INSERT statement. So if you are about to do batch scoring with millions of scores, a better approach is to do it with DB2 base tables as described in the next section.

6.3.3 Batch scoring using Scoring Adapter into z/OS DB2 base tables (more efficient method)

A better approach is to score into a base table. This works exactly as explained in 6.3.2, “Batch scoring using Scoring Adapter into accelerator-only tables (less efficient method)” on page 128 from a user perspective, except that the tables used must be DB2 base tables (both the source table and the target node table in which the scores are saved.

If you combine base tables and accelerator-only tables in one stream, SPSS Modeler will execute the stream, which is called *database federation*. What happens if you transform data from a base table to an accelerator-only table is that SPSS Modeler reads all data from the source database, performs potential transformation and then stores it to the accelerator-only table. From SPSS Modeler perspective, a database connection with accelerator is an independent DBMS, completely unrelated to a database connection without accelerator, even if the underlying z/OS DB2 subsystem is always the same. This is usually not what is intended, and it does not show acceptable performance.

Therefore, start by adding a new base source table. So far, you have the stream that created the model in accelerator, so all tables used so far are accelerator-only tables as shown in Figure 6-10 on page 133.

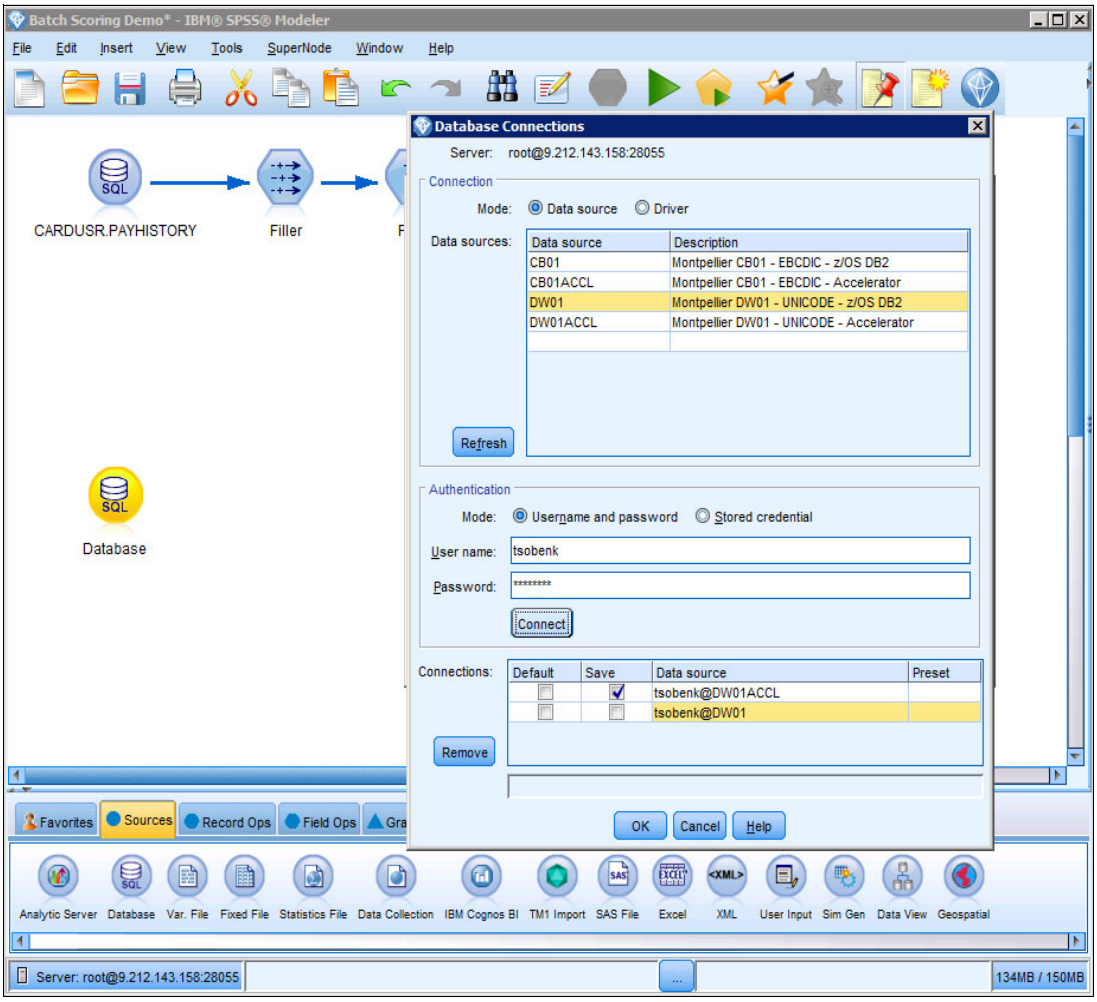


Figure 6-10 Add z/OS DB2 base table corresponding to the accelerated table used for model create

Now the transformations are done before the model create node is required for the model apply node (gold nugget) also. Therefore, you must copy the Filler and Filter nodes, connect them to the DB2 z/OS K-Means gold nugget (model apply node) and then connect it to an export database node with a DB2 base table target, as shown in Figure 6-11.

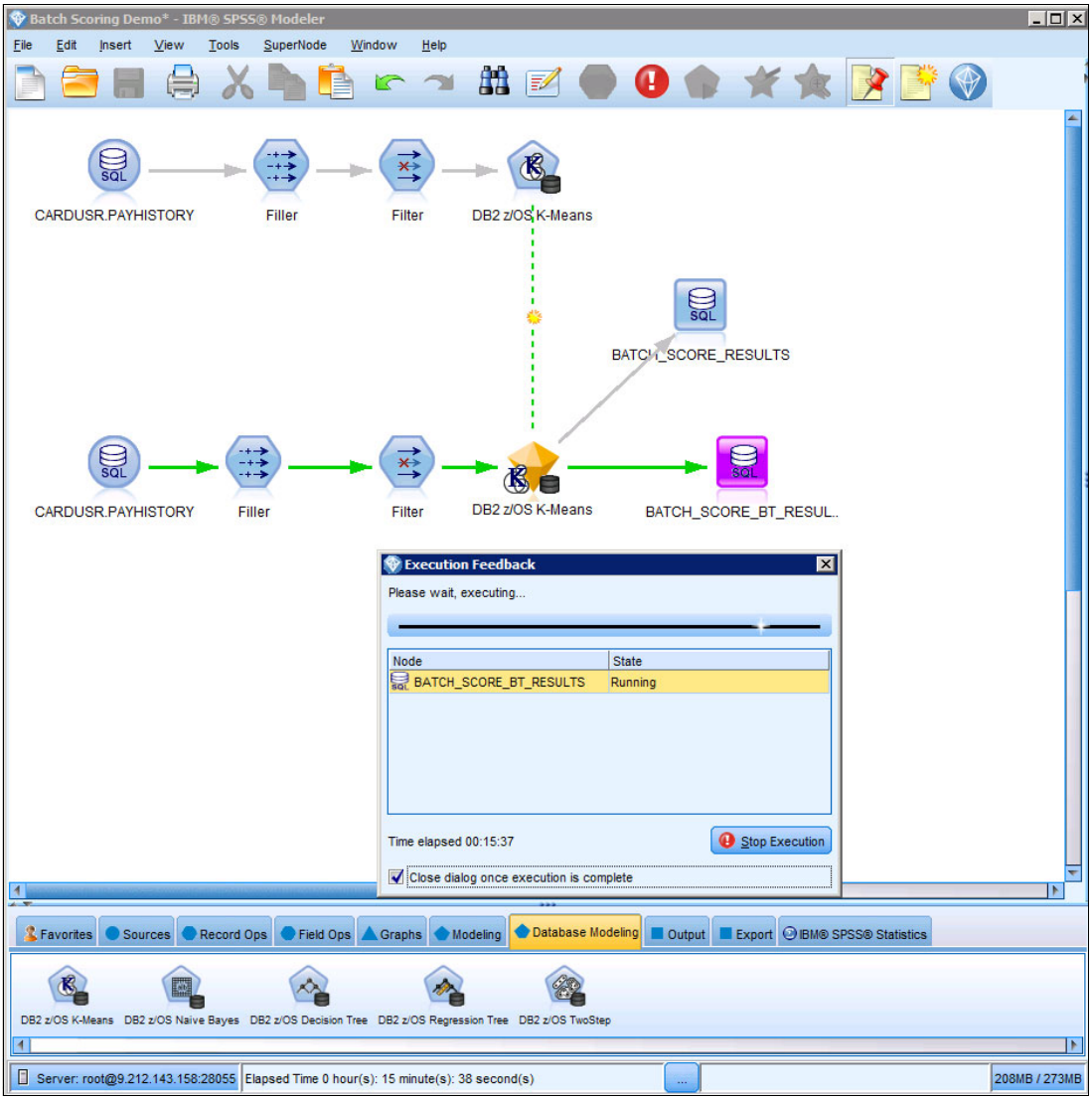


Figure 6-11 Stream where data transformation is replicated on DB2 z/OS, so the model gold nugget gets same input as for model create, then scoring to z/OS DB2 base table

As shown in Figure 6-11, while running that stream, only the database export node turns purple. However that does not mean that it is not running in-database, as you can see by verifying the SQL output of the stream shown in Figure 6-12 on page 135.

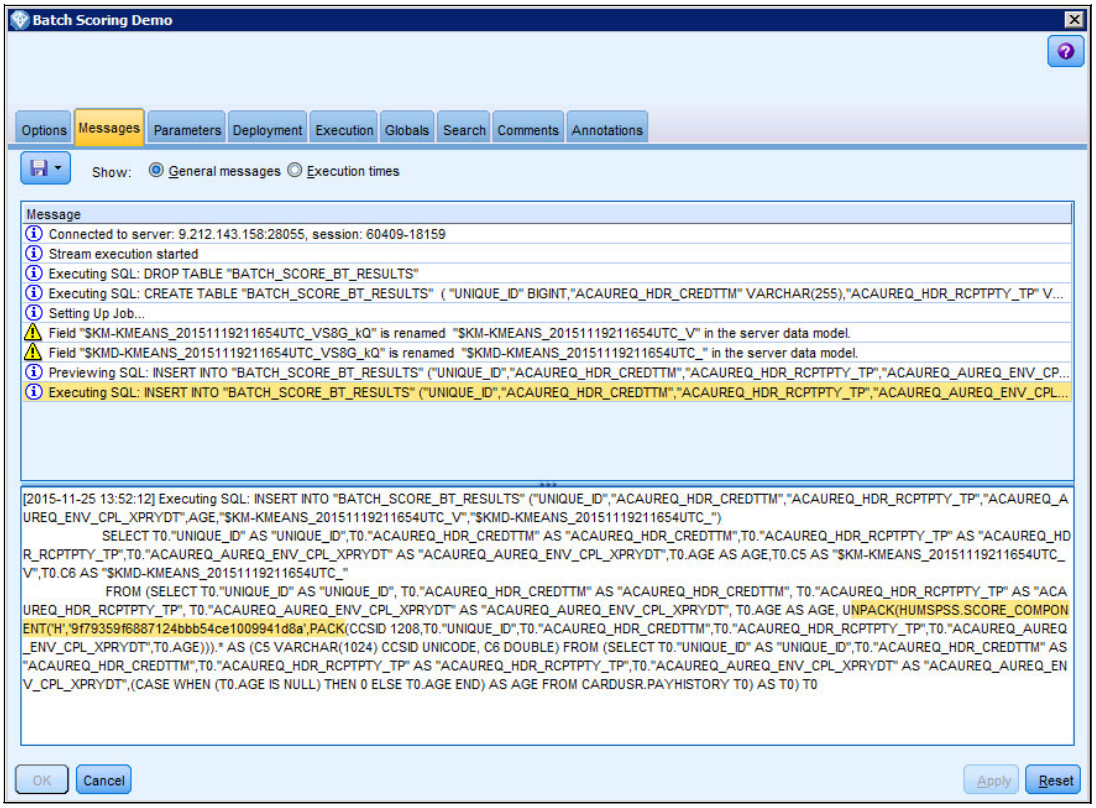


Figure 6-12 Efficient batch scoring from and to DB2 base table, running completely within z/OS, with no data transfer outside and no single-row insert

All SQL that is issued by SPSS Modeler looks similar to the stream above, with target accelerator-only table, however it does not do a SELECT using score UDF followed by single-row INSERT. Rather it does an INSERT FROM SELECT, using the score UDF in the SELECT part of that statement. Everything else is identical.

This way, we were able to score one million rows with that stream in 48 seconds.

If you want the batch score results on the Accelerator, you must load the result tables. That task is one that the z/OS DB2 DBA does, using Data Studio with DB2 Analytics Accelerator plug-ins. SPSS Modeler is intended for the data scientists and does not implement DBA tasks. Data Studio is intended for the DBA.

6.4 DMG and PMML

Predictive Model Markup Language (PMML) is an XML-based standard that is defined by the Data Mining Group (DMG), a non-profit organization:

`http://dmg.org`

The goal is to have an interoperable standard between statistical and data mining model producer and consumer. The PMML model can be produced using tools (such as R, SPSS Modeler, SAS Enterprise Miner, or others) and then deployed for real-time scoring in an OLTP layer using scoring technology (such SPSS Scoring Adapter or Zementis).

6.4.1 Zementis and in-application scoring

Zementis delivers Java libraries that can run in-application if the transaction environment contains a Java Virtual Machine (JVM). Note that on z Systems platform, this workload is eligible to use IBM z Systems Integrated Information Processor (zIIP).

6.5 Scoring and smart decisions

With IBM SPSS Collaboration and Deployment Services, you can manage analytical assets, automate processes, and efficiently share results widely and securely.

Predictive analytics uses historical data to find patterns, and the outcome of such analysis is often mathematical scores that are best applied when combined with an experience human to deliver the smarter decisions.

A smart decision system encapsulates business rules and predictive models. Business rules codify the best practices and human knowledge that a business builds over time. Predictive models use statistics and mathematical algorithms to recommend the best action at any given time.

The combination of those two disciplines enables organizations to optimize their decisions.

A solution that combines technologies such as predictive analytics and business rules can improve customer service, reduce fraud, manage risk, increase agility, and drive growth.

6.5.1 Solution architecture

Figure 6-13 on page 137 shows the relationship between the key components to implement a scoring execution solution for smarter decisions.

The client application running in CICS or IMS on z/OS interacts with a Decision Service Orchestration component. This component is responsible to orchestrate calls to these services:

- Data preparation
- Scoring model execution
- Business rules execution

This Decision Service Orchestration component can be implemented either in the CICS and IMS application or in an integration layer running on z/OS, such as IBM Integration Bus.

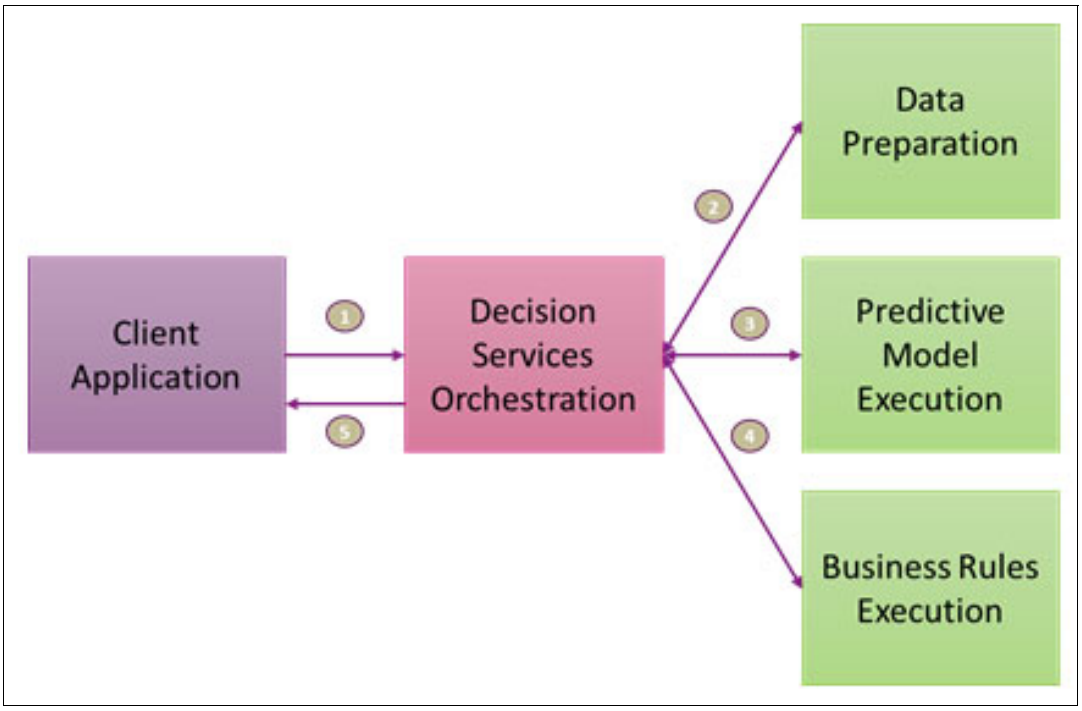


Figure 6-13 Implementing a scoring execution solution for smarter decisions

Data preparation

Before the execution of the scoring model, the data might need to be prepared and formatted before being sent to the scoring execution engine.

In some scenarios, the IBM DB2 Analytics Accelerator can significantly accelerate the preparation of that data by calculating some aggregates in real time and remove many of the burdens that are associated with the traditional batches for data aggregation.

Predictive model execution

The predictive model execution allows the analysis of every individual incoming transaction and assigns each transaction a *score* based on specific factors that are indicators of the quality of the transaction. Those indicators might have been calculated during the data preparation stage.

If using SPSS Collaboration and Deployment Services, two scoring engines for predictive models execution can be used on IBM z Systems platform:

- IBM SPSS Collaboration and Deployment Services Scoring Service
- IBM SPSS Scoring Adaptor for DB2 for z/OS

Business rules execution

The business rules execution step refers to the core logic of how the business decision is made, taking the input of the context and the predictive behavior of the subject of interest. Basically a business rule is a statement that describes a business policy or procedure. Business rules describe, constrain, or control some aspect of the business. IBM Operational Decision Manager can be used to automate the decision-making process.

6.5.2 Operational Decision Manager overview

A business rule management system (BRMS) is an essential technology within operational decision management (ODM). It allows organizational policies, and the repeatable decisions associated with those policies such as claim approvals, cross-sell offer selection, pricing calculations, and eligibility determinations, to be defined, deployed, monitored, and maintained separately from application code. With a BRMS, organizations can be more agile and effective in their decision management processes.

A BRMS manages decisions represented by collections of rules. These rules provide solutions to complex decisions that can be stateless or stateful. When decisions are stateless, the ability of the system to provide a result depends on two items:

- Data provided to the system
- Defined business rules

These business rules are evaluated against provided data and when conditions in the rules are met, actions are taken. Stateless decisions do not keep any history on previous results, so every time that the system is started, all the business rules are evaluated against provided data. These types of decisions are considered request-driven decisions because there is no history of previous invocations. Request-driven decisions are always made in a passive mode, which means you always need a client application to trigger a request for a decision.

IBM Operational Decision Manager is the offering to support complex decisions for both request-driven and situation-driven decisions. It provides a set of components to fully support needs for authoring, storing, and running decisions. The following main components are part of the solution:

- IBM Decision Center

  This component focuses on the rule management process and provides all tools that are needed for authoring and governing business rules. This component provides two main web-based consoles:

  - One for users to author, validate, and maintain rules
  - Oriented toward administrative tasks

- IBM Decision Server

  This component focuses on the execution, design, and monitoring of all executable artifacts.

Operational Decision Manager provides options for running the Decision Server on z/OS:

- zRule Execution Server for z/OS (zRES)

  This option offers a native integration with existing z/OS applications. A supplied stub program provides an API to directly execute decisions in the zRule Execution Server for z/OS. The zRES can run in two modes:

  - Stand-alone mode provides a zRES address space that can be invoked from existing COBOL or PL/I applications by way of the supplied API stub.
  - In IBM CICS Transaction Server for z/OS V4.2 and later, a local execution option uses the Java virtual machine (JVM) server environment to host the zRES within the CICS region.

- Rule Execution Server on WebSphere Application Server for z/OS

  This option brings the full power of WebSphere Application Server for z/OS to the rule execution on z/OS.

6.5.3 SPSS Collaboration and Deployment Services

IBM SPSS Collaboration and Deployment Services provides a flexible, enterprise level foundation for managing and deploying analytics in an enterprise. The product enables collaboration and provides reliable automation of analytical processes for better orchestration and discipline in an organization. It also streamlines deployment of analytical results throughout the enterprise to enable better decision-making.

The IBM SPSS Collaboration and Deployment Services Deployment Portal allows the generation of scores from models deployed in the Collaboration and Deployment Services repository. The SPSS Modeler streams scenario files, and PMML files can be scored. This component facilitates scoring of individual incoming transactions in real time. The scoring is performed directly within the OLTP application.

This real-time capability and the ability to incorporate the scoring into the decision-making process makes this a powerful solution for real-time analytic applications such as fraud detection.

The SPSS Collaboration and Deployment Services platform supports also predictive model governance in the following ways:

- Suitable roles and permissions can be configured to deploy and manage the model, such as an IT admin to schedule jobs or real-time scoring for the models.
- Model performance is automatically monitored and can be configured to provide an alert when a model's accuracy degrades to a predetermined level.
- Models and the related performance metrics are version-controlled.
- The predictive model is periodically trained on new data, and this is done using a model refresh. A model refresh is the process of rebuilding an existing model in a stream using newer data.

6.5.4 SPSS Scoring Adaptor for DB2 z/OS

Scoring data is defined as deploying a predictive model on new data with an unknown outcome. This predictive model processes incoming data and produces a predictive score about the likelihood or probability of an event.

The IBM SPSS Modeler Server Scoring Adapters for DB2 z/OS allows data to be scored by the generated models within the database, which improves performance. DB2 z/OS allows SQL pushback of most of the SPSS Modeler model nuggets. In these cases, model scoring can be performed within the database, avoiding the need to extract the data before scoring.

The Scoring Adapter for DB2 z/OS defines a UDF that applications can start by using SQL to run the scoring models synchronously, inline within their transactions, by using live transaction data as the input for scoring to maximize the effectiveness of scoring results.

SPSS Modeler is used to publish the model nugget to the scoring adapter. When a model is published to a server scoring adapter, it generates a sample SQL statement. This SQL statement uses UDFs to start the SPSS model that was built earlier and generates a predictive score that can be used by a decision management system.

6.6 Examples of integration for scoring solutions using ODM on z/OS

This section describes two integration examples using ODM on z/OS.

6.6.1 Example: Calling the SPSS Scoring Adapter for IBM DB2 for z/OS

Figure 6-14 illustrates an example of an optimized infrastructure solution based on the SPSS Scoring Adapter for DB2 for z/OS for the scoring engine and ODM on z/OS for the business rules management system.

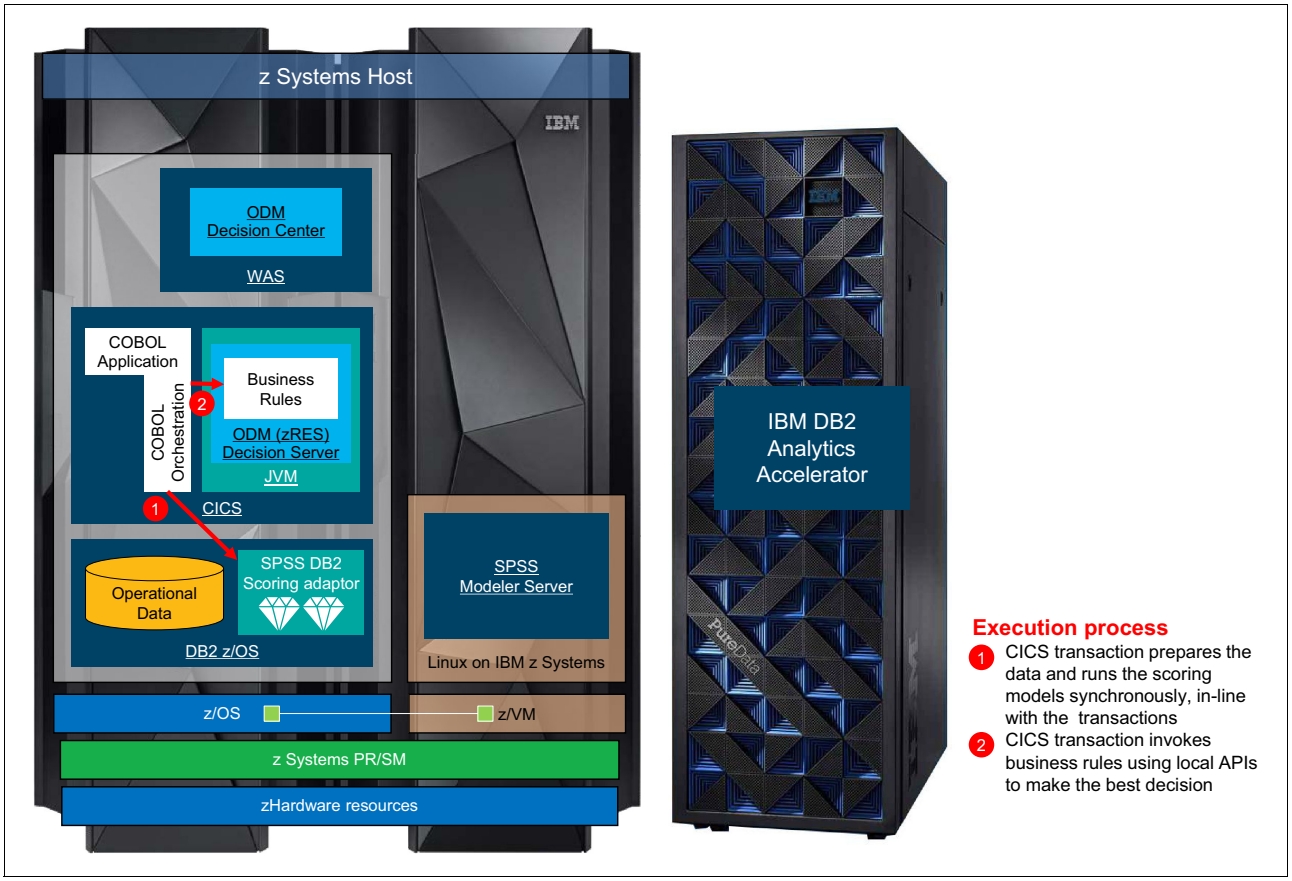


Figure 6-14 Optimized infrastructure

In this example, the decision service orchestration is implemented in COBOL in CICS. The CICS transaction will prepare the data for the scoring and will call the scoring engine running in DB2 z/OS within the same transaction. This data preparation and access to the scoring engine maximizes the effectiveness of scoring results.

For the business rules execution, the CICS transaction calls the zRule Execution Server (zRES) running inside CICS in a JVM using local APIs provided by ODM.

This example is an optimal solution from a performance, high availability, and system management perspective to perform real-time analytics on IBM z Systems platform.

6.6.2 Example: Calling SPSS Collaboration and Deployment Services

In the example discussed in 6.6.1, “Example: Calling the SPSS Scoring Adapter for IBM DB2 for z/OS” on page 140, the model is run and data is processed in the DB2 environment. In the alternative approach, the process occurs in the Collaboration and Deployment Services environment. The data is outside of a Collaboration and Deployment Services environment and must travel from the database to Collaboration and Deployment Services for processing, which causes this approach to be slower than the DB2 scoring adapter approach.

Figure 6-15 shows this alternative scoring method. The predictive model is developed by using IBM SPSS Modeler Client and the model is published to IBM SPSS Collaboration and Deployment Services running on Linux on IBM z Systems platform. First, the CICS transaction calls DB2 z/OS to prepare the data, then the predictive model is called with a web service call from the CICS transaction. The model processes the data and produces a real-time score that will be used to make business decisions in ODM running in CICS.

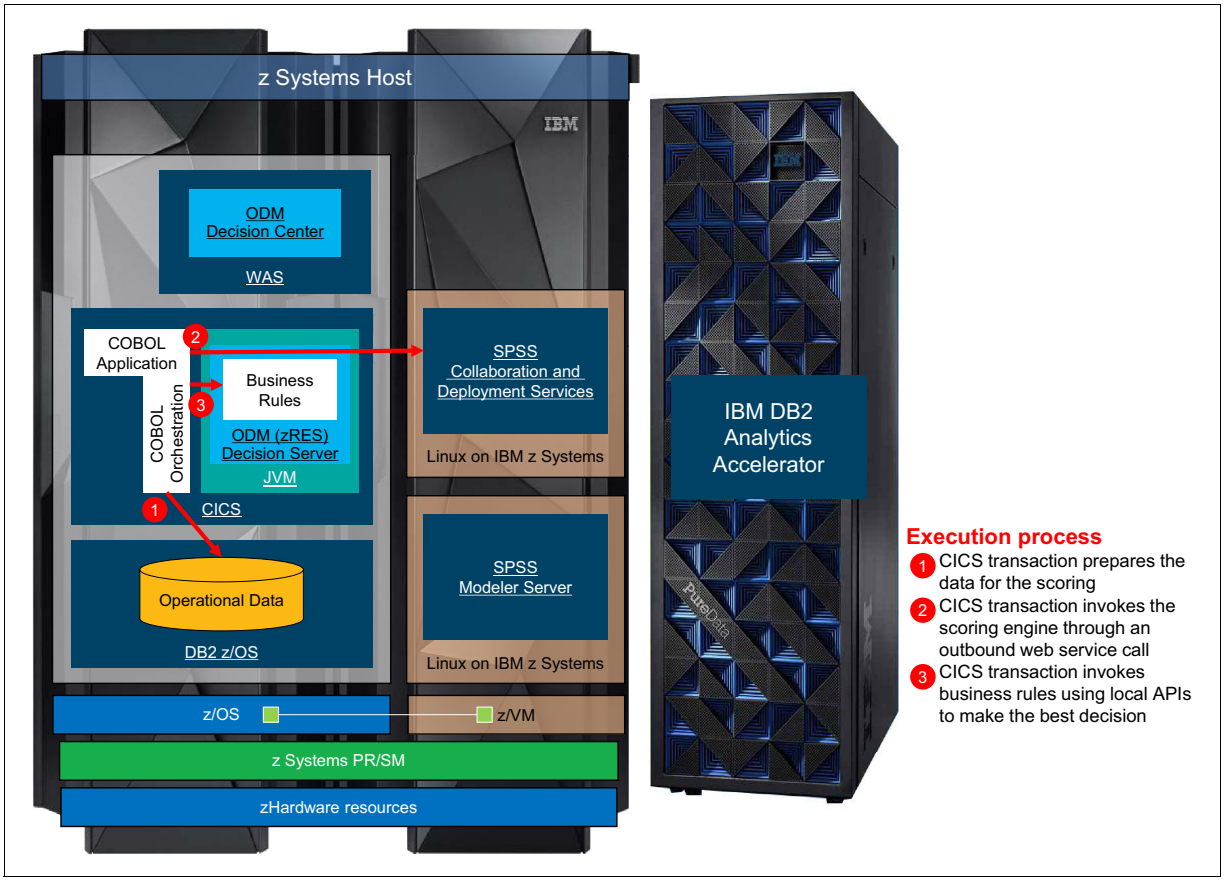


Figure 6-15 Alternative optimized infrastructure

7

Acceleration for descriptive analytics: IBM DB2 Analytics Accelerator value with IBM Cognos BI

Delivering information fast is the objective of putting in place an analytics solution: a set of data reference architecture (or data lake) concepts and methodology, added by software products hosted on a strong infrastructure able to use those sets of technical solutions.

The analytics solution covers several stages of data analysis, such as these:

- Descriptive analytics: What happened?
- Diagnostic analytics: Why did it happen?
- Predictive analytics: What might happen in the future?
- Prescriptive analytics: How should we respond to those potential future events?

These stages are the strong foundation for a company to start their cognitive transformation.

Because a majority of business analytics (most notably social analytics) are descriptive, this chapter focuses on descriptive analytics and demonstrates the value that IBM z Systems platform can bring to a descriptive solution.

This chapter contains the following topics:

- Descriptive analytics
- IBM Cognos Business Intelligence
- IBM Cognos BI 10 and DB2 Analytics Accelerator and z13

© Copyright IBM Corp. 2016. All rights reserved.

7.1 Descriptive analytics

Descriptive analytics are the most commonly used and most well understood type of analytics. The goals are to categorize, characterize, consolidate, and classify data. Descriptive analytics techniques are most commonly applied to structured data, although numerous efforts have been made to extend their reach to unstructured data, often through the creation of structured metadata and indices. Descriptive analytics help provide an understanding of the past and also events occurring in real-time.

The descriptive analytics tool must be a unified workspace for the entire organization that can be used to achieve these results:

- Answer key business questions.
- Explore all types of information from all angles to assess the current business situation.
- Analyze facts and anticipate tactical and strategic implications by shifting from viewing to more advanced, predictive, or what-if analysis.
- Collaborate to establish decision networks to share insights and drive toward a collective intelligence.
- Provide transparency and accountability to drive alignment and consensus.
- Communicate and coordinate tasks to engage the right people at the right time.
- Access information and take action anywhere, taking advantage of mobile devices and real-time analytics.
- Integrate and link analytics in everyday work to business workflow and process.

As the descriptive solutions have democratized the information access, they are really the entry point for information delivery and challenge the host infrastructure every day: scalability, performance, and reliability and the key non-functional requirements that the platform must rely on.

IBM Cognos Business Intelligence is the descriptive solution tool, hosted on IBM z Systems (for Linux on z Systems and z/OS) that fulfills scalability, performance and reliability requirements.

7.2 IBM Cognos Business Intelligence

IBM Cognos Business Intelligence (BI) helps your organization make smarter decisions, achieve better results, and gain a deeper understanding of trends, opportunities, weaknesses, and threats. With Cognos BI, you can explore any data, in any combination, over any time period, and with a broad range of analytics capabilities.

IBM Cognos 10 delivers a revolutionary new experience and expands traditional business intelligence (BI) with planning, scenario modeling, real-time monitoring, and predictive analytics.

Cognos 10 provides the following analytic capabilities:

- Query, reporting, and analysis through a Cognos BI server or interactive offline activity reports
- Scorecarding
- Dashboarding

- Real-time monitoring
- Statistics
- Planning and budgeting
- Collaborative BI

For more information about IBM Cognos Business Intelligence v10.1.1, see this web page:

http://www.ibm.com/support/docview.wss?uid=swg27021353

7.3 IBM Cognos BI 10 and DB2 Analytics Accelerator and z13

The descriptive analytics solution is a huge consumer of data and huge provider of information for the line of business. The IBM DB2 Analytics Accelerator brings value for this typical descriptive analytics solution:

- Dramatically improves query response up to 2000X faster to support time-sensitive decisions.
- The descriptive analytics solution comes at the right time with low latency and provides trusted and accurate results.

Figure 7-1 shows analytics reports and workloads from a typical day.

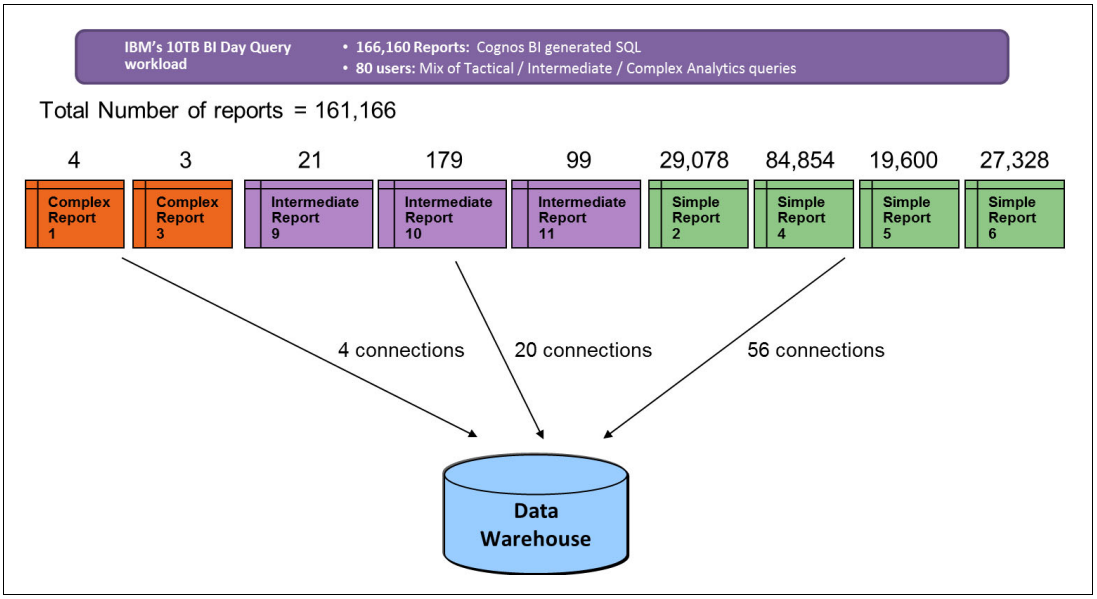


Figure 7-1 Typical analytical reports and workloads

With IBM DB2 Analytics Accelerator, the large workload shown in Figure 7-1 will be accelerated, thus putting less stress on the system and freeing resources for other workloads.

Consider the following information about workload performance improvements shown in Figure 7-1:

- DB2 v11 provided a greater than 30% improvement over DB2 v10.
- Improvement was for the native DB2 portion of the workload because of significant improvements in DB2 access paths, in-memory enhancements, and query parallelism.
- The IBM z13 provided approximately a 30% improvement over the zEC12.

Figure 7-2 shows a comparison of hardware and software and the query workload improvements.

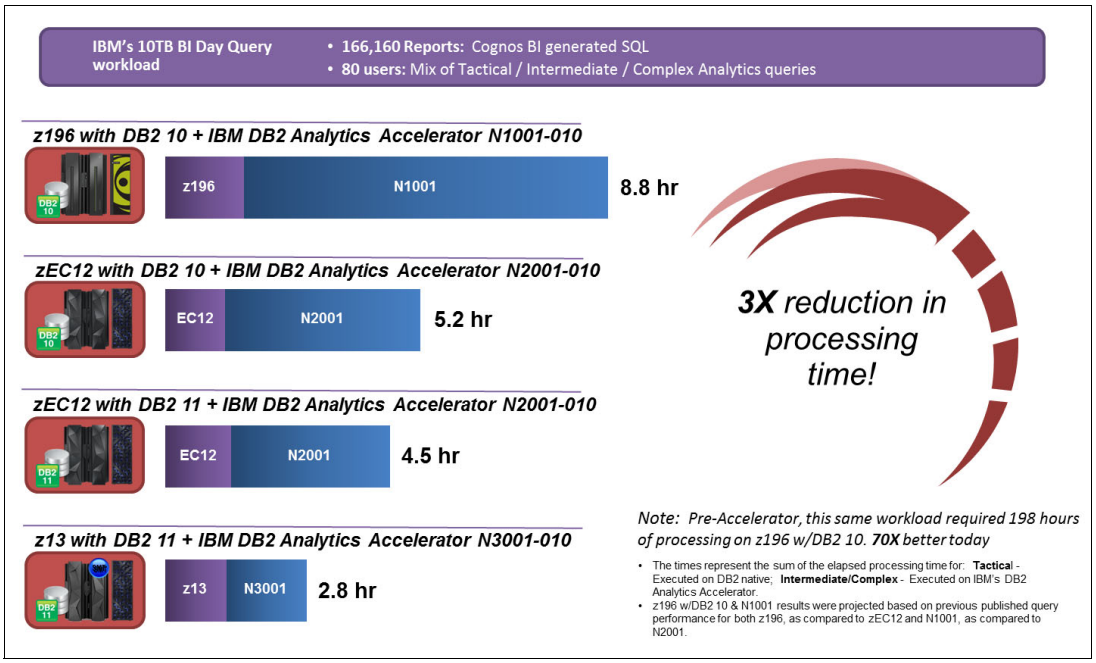


Figure 7-2 Hardware and software workload performance comparison

IBM Cognos BI, powered by IBM DB2 Analytics Accelerator on IBM z13 allows a descriptive analytics solution to be reliable, scalable, secure, and cost-efficient.

Figure 7-3 shows how many more reports per minute can be created on the IBM z13 using data acceleration and IBM Cognos BI.

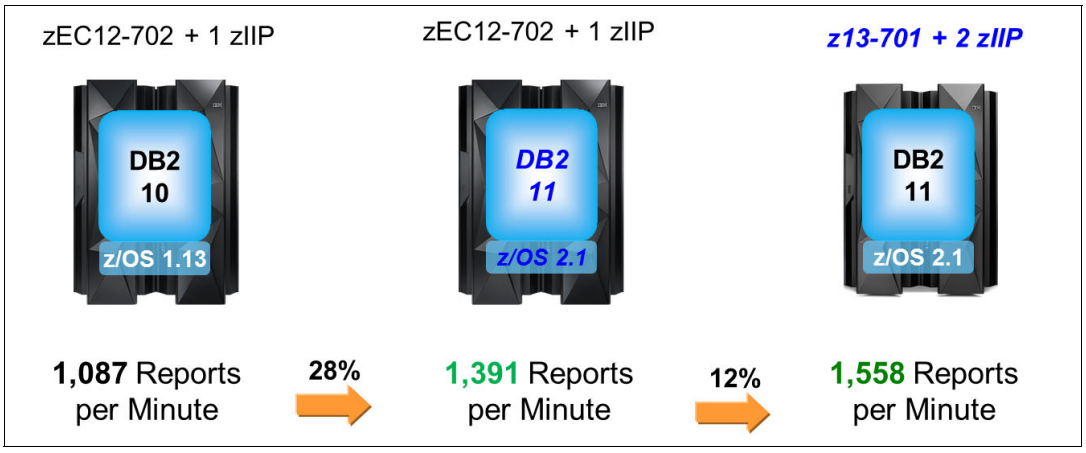


Figure 7-3 Reports per minute

Descriptive analytics does provide significant insight into business performance and enables users to better monitor and manage their business processes.

IBM Cognos BI 10 with IBM DB2 Analytics serves as a first strong foundation for an analytics solution to go to predictive or prescriptive analytics.

For more technical information, see the following publications:

- *Optimizing DB2 Queries with IBM DB2 Analytics Accelerator for z/OS*, SG24-8005
- *IBM Cognos Business Intelligence V10.1 Handbook*, SG24-7912

8

Using R with the IBM DB2 Analytics Accelerator

This chapter introduces the use of the R programming language with the IBM DB2 Analytics Accelerator. It discusses the installation of necessary components and the various techniques for using analytical functionality.

This chapter contains the following topics:

- Introduction to the R programming language
- Prerequisites
- Using RJDBC with the IBM DB2 Analytics Accelerator
- Using RODBC with the IBM DB2 Analytics Accelerator
- Using accelerator-only tables (AOTs) with R
- Using Netezza Analytics stored procedures
- Performance considerations

© Copyright IBM Corp. 2016. All rights reserved.

8.1 Introduction to the R programming language

The R programming language first appeared 1993. It is based on the S programming language, developed at Bell Laboratories, and was created at The University of Auckland in Auckland, New Zealand. The programming language is widely used among statisticians and data scientists because of its wide variety of built-in statistical functions. R can be extended easily by using Comprehensive R Archive Network (CRAN). CRAN contains various functions for standard tasks. R functions operate on *data frames*, the standard data representation within R, similar to a table within a database. In a data frame, data is organized in named columns and rows, where the combination of a specific column and row denotes a cell.

Because most of the operational data resides in databases, such as IBM DB2 for z/OS, combining the powerful language features of R with the business data in the database that has a similar format as R data frames makes sense. Working with the data can be done in various ways:

- Data can be extracted from the operational system, transformed, and loaded into an R compatible format (for example, in an extract, transform, or load (ETL) program).
- R can directly access the database data through various interfaces such as these:
  - RJDBC: This is an R package that implements JDBC.
  - RODBC: This is an R package that implements ODBC.

When R directly accesses the database data through the interfaces, projections, aggregates, joins, and other operations can be directly performed by the database (for example, IBM DB2 for z/OS). If the IBM DB2 Analytics Accelerator is attached to the system, this process is faster and introduces more analytical capabilities.

8.2 Prerequisites

To use R with the IBM DB2 Analytics Accelerator, you must obtain either the source code or a precompiled binary for your operating system. If you use the UNIX or Linux operating system, R is usually available through the integrated package manager. Windows users can download an installer from the CRAN project website:

`https://cran.r-project.org/bin/windows/`

After you install the R base system, continue with the installation of RJDBC or RODBC:

- To install RJDBC, see 8.3, “Using RJDBC with the IBM DB2 Analytics Accelerator” on page 151.
- To install RODBC, see 8.4, “Using RODBC with the IBM DB2 Analytics Accelerator” on page 154.

Note: Using the CRAN modules might require that you compile the contents of the module. The compile process requires a recent compiler on the system and common build utilities. If building a module fails, try to install the required development headers. OS X and Windows users can use prebuilt binary modules provided by CRAN if they exist; Linux and BSD users can find prebuilt R modules within the system repositories.

Connecting to the IBM DB2 Analytics Accelerator requires a valid Time Sharing Option (TSO) user ID and password combination for an IBM DB2 for z/OS system. The DB2 subsystem must be paired with an accelerator before queries can be offloaded to the Accelerator.

For more details about setting up and pairing the Accelerator, see the IBM DB2 Analytics Accelerator for z/OS documentation in the IBM Knowledge Center:

https://www.ibm.com/support/knowledgecenter/SS4LQ8/welcome

8.3 Using RJDBC with the IBM DB2 Analytics Accelerator

RJDBC is an easy-to-use implementation of the R standardized database interface (DBI) using Java Database Connectivity (JDBC) as a back end. Users who are familiar with other DBI-database drivers for R such as RSQLite, RPostgreSQL, or RMySQL profit from this compatibility. As RJDBC suggests, the classes use JDBC drivers and therefore require a Java Runtime Environment[1] to be present.

To install RJDBC open an R shell and type the following line:

```
install.packages("RJDBC", dependencies=TRUE)
```

You might be required to adjust the Java path after installation if the auto-detection mechanism is not working correctly. Use the R built-in **javareconf** command with a correctly set JAVA_HOME environment variable. The following example uses a Linux shell with an OpenJDK:

```
$ export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64/
$ R CMD javareconf
```

If you are using a different operating system or Java version, you must adjust the path accordingly. The output of the **javareconf** utility will indicate the Java setup and update the Java configuration variables.

After that, you can use the RJDBC library within R to connect to your IBM DB2 for z/OS database. Example 8-1 shows how to connect to the IBM DB2 for z/OS database using RJDBC and query a basic table. Note that the path to the `db2jcc4.jar` file must be adjusted to the corresponding path. In addition, the database host, port, and name must be adjusted and so must the user name and password.

Note: The `db2jcc_license_cisuz.jar` might also have to be in the same directory as the `db2jcc4.jar` file or in the default java class path. Alternatively you can force load the JAR file by using the following command before initializing the RJDBC object:

```
library(rJava)
.jaddClassPath("/opt/ibm/jdbc/db2jcc_license_cisuz.jar")
```

Example 8-1 Connecting to the IBM DB2 for z/OS database

```
library(RJDBC)
drv <- JDBC("com.ibm.db2.jcc.DB2Driver","/opt/ibm/jdbc/db2jcc4.jar")
conn <- dbConnect(drv,
         "jdbc:db2://myhost:1011/mydb",
         user="dbadm",
         password="dbadm")
```

[1] The JDK is required in order to build the module from CRAN if not installed by the systems package manager.

The next step is to enforce query acceleration for the currently open connection. Do this by sending the following query to the connection:

```
dbSendQuery(conn, "SET CURRENT QUERY ACCELERATION = ALL")
```

After the query is sent, all further queries sent to DB2 for z/OS are accelerated. If a query cannot run on the accelerator, the query will fail. A preferable approach might be to try acceleration first and then retry on DB2 for z/OS, in case of failures, by using this command:

```
dbSendQuery(conn, "SET CURRENT QUERY ACCELERATION = ENABLE WITH FAILBACK")
```

After setting the query acceleration, the DBI implemented by RJDBC can be used normally. No additional specific database commands must be performed in order to use RJDBC with the IBM DB2 Analytics Accelerator.

Example 8-2 demonstrates a case that uses the processing capabilities of the Accelerator to increase the speed of query processing in combination with RJDBC. The data used here is provided by the United States Environmental Protection Agency and contains metrics about cars such as fuel consumption, the manufacturers, and other parameters. A copy of the data is at the following US Department of Energy web page:

https://www.fueleconomy.gov/feg/download.shtml

Example 8-2 Using the accelerator in combination with RJDBC

```
library(RJDBC)
drv <- JDBC("com.ibm.db2.jcc.DB2Driver","/opt/ibm/jdbc/db2jcc4.jar")
conn <- dbConnect(drv,
         "jdbc:db2://myhost:1011/mydb",
         user="dbadm",
         password="dbadm")
dbSendQuery(conn, "SET CURRENT QUERY ACCELERATION = ENABLE WITH FAILBACK")
rs <- dbSendQuery(conn, "SELECT * FROM fueldata WHERE VClass=\"Two Seaters\" AND
make in (\"BMW\",\"Ferrari\",\"Subaru\",\"Toyota\")")
df.cars <- fetch(rs,-1)
boxplot(barrels08~make,df.cars, main="Fuel consumption by make",ylab="Fuel
consumption in barrels")
```

Figure 8-1 shows the results of that query.

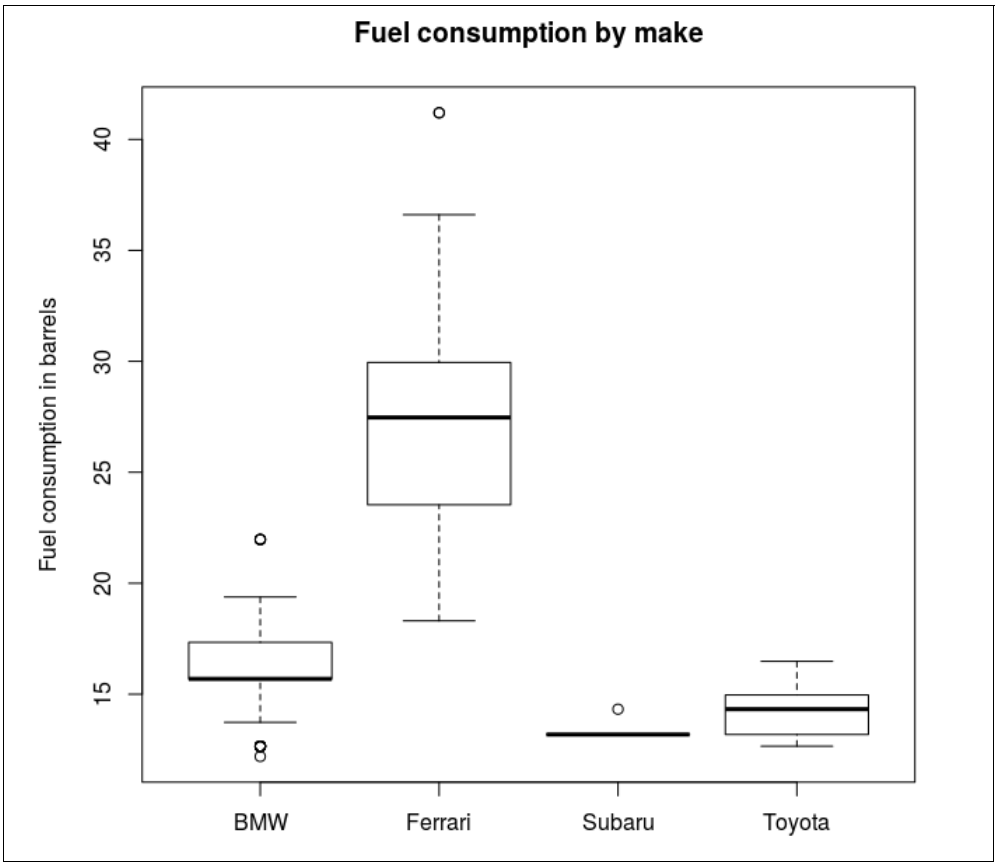


Figure 8-1 Fuel consumption by make

8.3.1 RJDBC functions

In addition to the functions introduced in the examples, other relevant RJDBC functions can automate interaction with RJDBC and the IBM DB2 Analytics Accelerator.

For the JDBCConnection object

The following methods are for the class JDBCConnection in the RJDBC package:

- dbCommit(JDBCConnection)

  Commits the current unit of work performed.

- dbDisconnect(JDBCConnection)

  Disconnects properly from the database with which the driver is connected. If the connection is not closed using this function call, it is terminated when the R application terminates.

- dbGetTables(JDBCConnection)

  Returns a data frame with all information about the tables on the database. The output can be used to automate tasks and locate specific tables.

- dbSendUpdate(JDBCConnection, statement)

  This function is similar to dbSendQuery(JDBCConnection, statement) but does not return a result set. It is designed for Data Definition Language (DDL) operations or Data Manipulation Language (DML) operations that do not return a result set.

For the JDBCResult object

The following methods are for the class JDBCResult in the RJDBC package:

- dbClearResult(JDBCResult)

  Clears or releases the result- set. It frees the memory used to store result values but does not release the memory of data frames that inherit from a fetched JDBCResult.

- dbColumnInfo(JDBCResult)

  Returns a data frame with information about the columns in the result including column names and data types.

8.4 Using RODBC with the IBM DB2 Analytics Accelerator

RODBC allows you to connect to databases using R and ODBC database connectivity. To use RODBC with the IBM DB2 Analytics Accelerator you must first install the RODBC package from CRAN. When the package is available through your systems package manager, the module supplied with package manager is the preferred way of installation. To use the version from CRAN, enter the following command in an R shell:

```
install.packages("RODBC", dependencies=TRUE)
```

You might need to install the appropriate build tools and header files to run this command. To do this, you will need at least a compiler and all sources to compile the ODBC drivers.

Two methods are available to connect to IBM DB2 on z/OS using the RODBC driver:

- Example 8-3 shows setting the source connection in the ODBC driver manager.

  Example 8-3 Using the ODBC driver manager predefined profile

```
library(RODBC)
conn <- odbcConnect(dsn="PROFILENAME",uid="USERNAME",pwd="PASSWORD")
```

- Example 8-4 shows setting the source connection directly by specifying the connection string.

  Example 8-4 Using the ODBC DSN

```
library(RODBC)data
# initialize variables
driver.name <- "{IBM DB2 ODBC DRIVER}"
db.name <- "SAMPLEDB"
host.name <- "sampledb2zos.example.ibm.com"
host.port <-  "10011"
user.name <-  "myuser"
user.pwd <-  "supersecret"
# generate connection text
con.text <- paste("DRIVER=",driver.name,
         ";Database=",db.name,
         ";Hostname=",host.name,
         ";Port=",host.port,
         ";PROTOCOL=TCPIP",
         ";UID=", user.name,
         ";PWD=",user.pwd,sep="")
conn <- odbcDriverConnect(con.text)
```

After successfully connecting to the specified database, be sure that query acceleration is enforced. To do this, use the function shown in Example 8-5, which takes the connection function, the query text, and additional fields as parameters. Setting the parameter errors to false ensures that errors are reported if they exist.

Example 8-5 Enforce query acceleration

```
res <- sqlQuery(conn,"SET CURRENT QUERY ACCELERATION = ALL", errors=FALSE)
if (res == -1){
   cat ("An error has occurred.\n")
   msg <- odbcGetErrMsg(con1)
   print (msg)
} else {
   cat ("Query acceleration successfully forced.\n")
}
```

To query the data, use the same function used in Example 8-5 to enforce the query acceleration. Example 8-6 uses the same data as the RJDBC example and aims to determine which car manufacturer produces cars with low fuel consumption.

Example 8-6 Query the data and enforce query acceleration

```
library(RODBC)data
# initialize variables
driver.name <- "{IBM DB2 ODBC DRIVER}"
db.name <- "SAMPLEDB"
host.name <- "sampledb2zos.example.ibm.com"
host.port <-  "10011"
user.name <-  "myuser"
user.pwd <-  "supersecret"
# generate connection text
con.text <- paste("DRIVER=",driver.name,
         ";Database=",db.name,
         ";Hostname=",host.name,
         ";Port=",host.port,
         ";PROTOCOL=TCPIP",
         ";UID=", user.name,
         ";PWD=",user.pwd,sep="")
conn <- odbcDriverConnect(con.text)
res <- sqlQuery(conn,"SET CURRENT QUERY ACCELERATION = ALL", errors=FALSE)
if (res == -1){
   cat ("An error has occurred.\n")
   msg <- odbcGetErrMsg(con1)
   print (msg)
} else {
   cat ("Query acceleration successfully forced.\n")
}
res <- sqlQuery(conn, "SELECT * FROM fueldata WHERE VClass=\"Two Seaters\" AND
make in (\"BMW\",\"Ferrari\",\"Subaru\",\"Toyota\")")
boxplot(barrels08~make,res, main="Fuel consumption by make",ylab="Fuel consumption
in barrels")
```

The output (Figure 8-2) is the same as in the RJDBC example (Figure 8-1 on page 153).

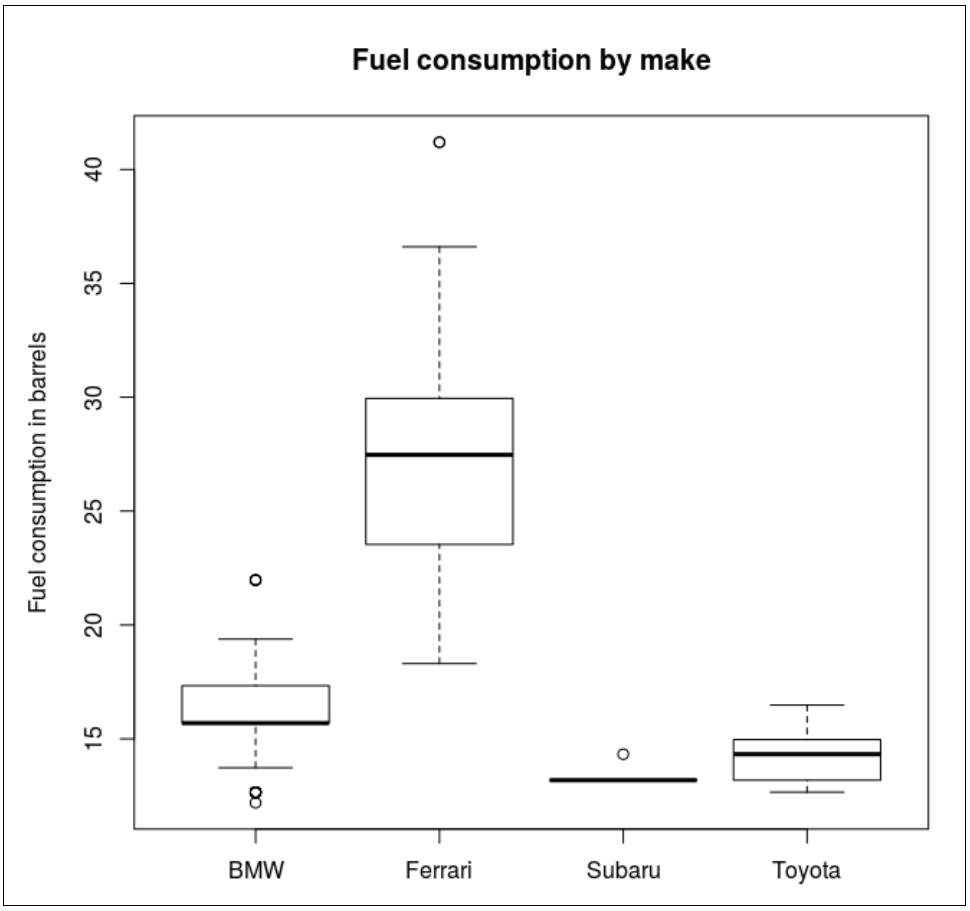


Figure 8-2 Fuel consumption by make

8.4.1 RODBC functions

Two groups of functions are provided in RODBC:

- RODBC low-level functions
- RODBC high-level functions

RODBC low-level functions

These functions are mainly internal **`odbc*`** commands that implement low-level access to ODBC functions with similar names:

- odbcDataSources()

  Lists all known ODBC data sources or drivers that are currently present on the system. Use this function to verify whether your IBM DB2 for z/OS driver is properly installed and accessible by RODBC.

- odbcClose(channel)

  Closes the specified channel or connection and frees up resources. Closing connections should be performed before the program terminates or no more work with the database is performed. Connections are forcefully terminated when the R program terminates.

- odbcCloseAll()

  Closes all currently open ODBC connections. It exposes the same behavior as odbcClose(channel) but iterates through all open connections.

- odbcSetAutoCommit(channel, autoCommit=TRUE)

  You specify whether all SQL statements should be automatically committed as individual transactions. When large amounts of data are ingested, a tremendous performance difference exists when `autoCommit=FALSE`. You must manually send a COMMIT query to the channel using the low-level odbcQuery() method or the high-level sqlQuery() method. It usually works best, if a COMMIT is performed after every 1000 - 10000 rows.

RODBC high-level functions

These **`sql*`** functions operate at a higher level to read, save, copy, and manipulate data between data frames and SQL tables:

- sqlSave(channel, dataframe, tablename, ….)

  Saves the specified data frame to the table specified in tablename on the given channel or connection. If the table does not exist, it gets created. Consider carefully the parameters of the function because they control the error behavior and performance. This function cannot be used to create an AOT, because the AOT creation syntax is not supported by RODBC. Instead create the AOT manually and then use sqlSave to ingest the data.

- sqlUpdate(channel, dataframe, tablename, ….)

  Updates the rows in the specified tablename with the data in the data frame. This function is used best if a data frame is fetched using sqlQuery() and then altered using R; then, it must be saved again.

- sqlFetch(channel, tablename, ….)

  Fetches the contents of the specified table into a data frame. Additional arguments can be used to limit the rows and columns that are fetched.

- sqlTables(channel, ….)

  Returns a data frame with information about schema and table names. This function can be used to automate certain processes, such as scanning for data in all available tables.

8.5 Using accelerator-only tables (AOTs) with R

An *Accelerator-only table* (AOT) is a table that exists only on the IBM DB2 Analytics Accelerator and not on IBM DB2 for z/OS. The feature places a proxy table on the DB2 for z/OS and that points at the corresponding table on the Accelerator. AOTs can be used starting with IBM DB2 Analytics Accelerator Version 4.1 PTF5 and DB2 PUTLEVEL 1504.

Use an AOT for the following reasons:

- An AOT is the required output table type for IBM Netezza Analytics Server stored procedures (see also 8.6, “Using Netezza Analytics stored procedures” on page 159). When you execute such stored procedures, results are written and made available through an AOT. This way prevents the copying of redundant data to the DB2 system and saves time and cost. Results are immediately available in the AOT for other programs without having the need to copy data.
- An AOT is excellent for storing temporary data such as additional external data (for example, relational data needed for your algorithms). The creation is fast and no disk space is taken on the original DB2. After the processing, the AOT can be dropped easily. When transformations are done in an ETL process, AOT can help to speed this process.

- An AOT can be used to quickly store intermediate results, for example when performing long-running jobs. If the job fails at some point, it can safely be resumed when programs abort for whatever reason.
- AOT can be used to store results and thereby make them available to other applications for further processing and usage by other tools and utilities.

To create an AOT, the **CREATE TABLE** statement must be extended by using the following keyword:

```
IN ACCELERATOR <ACCELERATORNAME>
```

Example 8-7 shows an SQL statement that creates an AOT in the Accelerator named ACCEL1 and the database named DBTEMP.

Example 8-7 Create an AOT

```
CREATE TABLE TEMP1 (
   C11 CHAR(2), ...)
   IN ACCELERATOR ACCEL1
   IN DATABASE DBTEMP;
```

After an AOT is created it can be used within RJDBC or RODBC like any other table. If you need to programmatically determine available accelerators to the subsystem you are currently connected to, you can query SYSACCEL.SYSACCELERATORS for details. The ACCELERATORNAME column contains valid names for the IN ACCELERATOR <ACCELERATORNAME> keyword to generate an AOT within that accelerator.

Example 8-8 creates an AOT to store processing results so it can be available to other applications based on the TPC-DS model.

Example 8-8 Creating an AOT to store processing results

```
library(RJDBC)
drv <- JDBC("com.ibm.db2.jcc.DB2Driver","/opt/ibm/jdbc/db2jcc4.jar")
conn <- dbConnect(drv,
          "jdbc:db2://myhost:1011/mydb",
          user="dbadm",
          password="dbadm")

res <- dbSendQuery(conn, "CREATE TABLE TPCDS.RESULTS(
                D_DATE_SK INTEGER,
                CA_STATE CHAR(2),
                AVG_INCOME_BOUND DOUBLE)
             IN ACCELERATOR ACCEL1
             IN DATABASE DBTEMP")

res <- dbSendQuery(conn, "INSERT INTO TPCDS.RESULTS
             SELECT
                a11.D_DATE_SK D_DATE_SK,
                a12.CA_STATE CA_STATE,
                AVG(a13.IB_UPPER_BOUND) AVG_INCOME_BOUND,
             FROM
                DATE_DIM a11,
                CUSTOMER_ADDRESS a12,
                INCOME_BAND a13
             GROUP BY
                a11.D_DATE_SK,
                a12.CA_STATE")
```

AOTs are subject to constraints, as described at the following web page:

http://www.ibm.com/support/knowledgecenter/SS4LQ8_5.1.0/com.ibm.datatools.aqt.doc/gui/references/r_idaa_restrictions_aots.html

The most notable constraints are described in the following list:

- Insert from select statements (such as `INSERT INTO <AOT> SELECT …. FROM ….`) are supported only if the FROM clause references only tables that are currently accelerated on the accelerator. In an environment with multiple accelerators, all tables that are referenced including the AOT must be present on the accelerator that is running the query. For instance, if you use the following statement, the AOT named MYAOT, and also TABLE1 and TABLE2 must all be on the accelerator.

  ```
  INSERT INTO MYAOT (SELECT * FROM TABLE1, TABLE2 WHERE TABLE1.ID = TABLE2.ID)
  ```

- You cannot run UPDATE statements for columns that define the distribution key. If you created an AOT with a non-random distribution key you cannot update the column that serves as a distribution key. The technical reason for this limitation is that an update of a distribution key requires the relocation of the updated row, which is not supported. As a workaround, you can split the UPDATE operation into a transaction with a DELETE and INSERT statement. Example 8-9 shows an SQL used to transform an UPDATE operation.

  Example 8-9 Transform UPDATE

```
# TABLE DEFINITION:
   CREATE TABLE UDEMO (
      ID INT,
      NAME CHAR(20),
      COUNTRY CHAR(20))
   IN ACCELERATOR ACCEL1
   IN DATABASE DBTEMP;

   # INSERT BASE DATA
   INSERT INTO UDEMO (ID,NAME,COUNTRY) VALUES (1,”SEBASTIAN”,”GERMANY”);

   # UPDATE STATEMENT THAT IS UNSUPPORTED
   # UPDATE UDEMO SET ID=2 WHERE NAME=”SEBASTIAN”

   # TRANSFORMED STATEMENTS
   DELETE FROM UDEMO WHERE ID=1;
   INSERT INTO UDEMO (ID,NAME,COUNTRY) VALUES (2,”SEBASTIAN”,GERMANY);
   COMMIT;
```

8.6 Using Netezza Analytics stored procedures

The IBM Netezza Analytics stored procedures are a set of analytical algorithms to perform research on data. They enable data mining capabilities for various business applications and predictive analytics based on different analytical models. The stored procedures are directly executed on the Accelerator and thereby reduce data movement. In addition, the algorithms are optimized for speed because they directly work on the Accelerator where the data resides.

To use the IBM Netezza Analytics with the IBM DB2 Analytics Accelerator, at least version 5.1 GA of the Accelerator software is required. In addition, the Analytics extension is required for this feature to work.

See the following resources:

- For Analytics Server documentation, see the following web page:

  https://www.ibm.com/support/knowledgecenter/SS4LQ8_5.1.0/com.ibm.datatools.aqt.doc/sp_msg/SPs/sp_idaa_wrapper.html

- For detailed information about the stored procedures, see Appendix B, “Job to install wrapper stored procedures” on page 179.
- *IBM Netezza In-Database Analytics Reference Guide* (release 3.0.1):

  https://ibm.biz/BdrFpq

IBM Netezza Analytics stored procedures are wrapped inside IBM DB2 for z/OS stored procedures. Calling IBM Netezza Analytics stored procedures using the wrapper-stored procedures slightly differs from the way it is described in the reference guide.

8.6.1 Basic concept

The analytical algorithms introduced in this chapter share a similar concept. The start is usually a training data set to create or train a model. After the model is generated or trained, the model can be used to classify data, for instance to put a specific set of data into a specific group or cluster based on certain characteristics. Models can be further enhanced by training them with additional training data.

Choosing the correct algorithm for the desired use-case is important. Although different algorithms can produce similar models, be sure you understand how the algorithms are supposed to work. Also, the quality of the resulting model heavily depends on the quality of the training data set. For example, if you want a classification model that can distinguish between ten categories, the training data set should contain all of the categories with significant data in it. If only two categories are present, an accurate model cannot be computed. In addition, the quality of the model might vary depending on the parameters supplied to the modeling algorithm. Be sure to understand the documentation about the algorithm to fully understand the influence of each parameter.

Calling IBM Netezza Analytics stored procedures using R programming language is simple. You can use RJDBC or RODBC to call them using simple SQL queries. Note that the location of the stored procedure might be different on your DB2 subsystem. For the JCL installation job for the stored procedures, see Appendix B, “Job to install wrapper stored procedures” on page 179.

The following generic syntax is for calling stored procedures:

```
CALL <schema>.<procedure>(<accelerator>,<parameterString>,<returnValue>,<message>)
```

In the stored procedure call, `<accelerator>` specifies the accelerator on which the procedure is executed. The Analytics Server packages must be installed on the accelerator, otherwise the stored procedure will fail. In addition, all tables referenced in `<parameterString>`[2] must be present on the specified accelerator. The `<returnValue>` depends on the stored procedure executed, alternatively a question mark (?) can be used as a placeholder.

[2] For syntax and semantics of the parameter string, see the *IBM Netezza In-Database Analytics Reference Guide*: https://ibm.biz/BdrFpq

Example 8-10 shows how to call a stored procedure using RODBC.

Example 8-10 Using RODBC to call a stored procedure

```
res <- sqlQuery(conn,"CALL INZA.KMEANS ('ACCEL01', 'model=customer_model,
   intable=TPCH.CUSTOMER, outtable=TPCH.CUSTOUT,
      id=C_CUSTKEY, target=C_NATIONKEY, transform=S,
      distance=euclidean, k=3, maxiter=5,
      randseed=12345, idbased=false',
      ?,'')", errors=FALSE)
```

Example 8-11 shows how to call a stored procedure using RJDBC.

Example 8-11 Using RJDBC to call a stored procedure

```
res <- dbSendQuery(conn, "CALL INZA.KMEANS ('ACCEL01', 'model=customer_model,
   intable=TPCH.CUSTOMER, outtable=TPCH.CUSTOUT,
      id=C_CUSTKEY, target=C_NATIONKEY, transform=S,
      distance=euclidean, k=3, maxiter=5,
      randseed=12345, idbased=false',
      ?,'')")
```

8.6.2 Helper stored procedures

In contrast to the analytical modeling stored procedures, the following IBM Netezza Analytics stored procedures help to prepare data and manage existing models on the accelerator:

- SPLIT_DATA

  Can be used to split data into two sets, for example, into a training data set and a test data set. The R equivalent for the stored procedure is the `sample(x, size, …)` function.

- MODEL_EXISTS

  Can be used to determine whether a model with the name given in the parameter string exists on the accelerator. Using this function is advised in multiple accelerator environments.

- LIST_MODELS

  Lists all models available on the accelerator specified.

- DROP_MODEL

  Drops a model from an accelerator. This process cannot be undone, so be careful.

- PMML_MODEL

  Returns the PMML representation of the model to the output table specified. The output type can be either standard PMML format or an SPSS specific format. The contents of the output table is compatible with the **`pmml`** package present in CRAN.

Note: Importing PMML models is not supported. Therefore, models modified with R utilities cannot be used with the IBM Netezza Analytics Server.

8.6.3 Decision trees

IBM Netezza Analytics stored procedures for decision trees (classification trees) are used for modeling decisions and their possible consequences. A typical decision tree contains the probability of each outcome or class in each leaf, and branches according to specific criteria:

- DECTREE

  Is used to build a decision tree based on the parameter string supplied. The most important parameters in the parameter string are the input columns, the ID column, and the stop criteria. In contrast to GROW_DECTREE, this stored procedure first grows a decision tree and then prunes it after.

  The “party” library from CRAN with the `ctree(...)` function or the `rpart` library with the `rpart(class,...)` function provides similar functionality.

- GROW_DECTREE

  Grows a decision tree according to the parameter string supplied. If you want to immediately prune the decision tree after, use DECTREE() instead.

- PRUNE_DECTREE

  Prunes a decision tree with the name specified in the parameter string. Note that pruning a decision tree will replace the model.

- PREDICT_DECTREE

  Applies the decision tree model specified in the parameter string to the input table specified. The resulting prediction will be saved in the specified `outtable`. When `outtableprob` is set in the parameter string, the probability predictions for each possible output class will be stored in the output table specified.

Hint: Data that is written to the table specified for `outtableprob` is useful when a simple decision is not sufficient. For instance in a scenario where customer demand is analyzed knowing the two or three most likely needs can be helpful.

8.6.4 Regression trees

The following IBM Netezza Analytics stored procedures are for regression trees, which are similar to decision trees but the predicted outcome is a real number instead of a class to which the data belongs:

- REGTREE

  Builds a regression tree. The use is similar to the decision tree stored procedure, therefore it creates the regression tree and prunes it directly after creation. The procedure provide similar functionality to the `rpart(anova,..)` function from the `rpart` CRAN module or the standard functionality of the “tree” package.

- GROW_REGTREE

  Grows a regression tree based on the parameters in the parameter string.

- PRUNE_REGTREE

  Prunes a regression tree based on the parameters in the parameter string. When building a new model, be sure to perform REGTREE directly because it builds and prunes the resulting tree in one step.

- PREDICT_REGTREE

  Applies the regression tree model specified in the parameter string to the input table specified. The resulting prediction will be saved in the specified `outtable`.

8.6.5 K-means clustering

The following IBM Netezza Analytics stored procedures are for K-means clustering. The algorithm uses a predefined number of clusters:

- KMEANS

  Generates a K-means clustering model that clusters the data of the input table into a number (n) of clusters. The parameter string determines the number of clusters, the input table, and the input-columns. The stored procedure can replace the built-in k-means functionality of R.

  Note: If K-means must be performed on large data frames that are not part of tables present on the accelerator, a faster way is to load the data into the accelerator through accelerator-only tables and use this stored procedure for K-Means clustering.

- PREDICT_KMEANS

  Applies a previously created K-means clustering model to an input table and returns the results through an output table specified in the parameter string.

8.6.6 TwoStep clustering

The following TwoStep clustering stored procedures automatically estimate the number of clusters when no number of clusters is specified. Data is processed in two passes, in contrast to K-Means:

- TWOSTEP

  Generates a TwoStep clustering model that distributes and compresses the input data into a set of subclusters using the data distance. Then it reduces the tree to the specified number of clusters using a hierarchical clustering method to merge the subclusters into larger clusters.

  The behavior of the algorithm is controlled using the parameter string. The `mclust` package available in CRAN provides similar clustering capabilities.

- PREDICT_TWOSTEP

  Applies the previously created TwoStep clustering model to the data to predict to which cluster the data belongs. Input and output tables are specified using the parameter string.

8.6.7 Naive Bayes

The following list describes the Naive Bayes modeling stored procedures. The Naive Bayes algorithm is a classification algorithm based on a vector of feature values:

- NAIVEBAYES

  Generates a Naive Bayes model based on the input parameters in the parameter string. Because of the nature of the algorithm, the input string can become long because many parameters must be supplied. The `klaR` and `e1071` modules from CRAN provide similar capabilities in R.

- PREDICT_NAIVEBAYES

  applies the Naive Bayes model, given in the parameter string, to the specified input table to generate classification predictions. Similar to the DECTREE prediction, the `outtableprob` parameter can be used to get probabilities for all classes.

8.7 Performance considerations

To make the R program as fast as possible, consider this information when programming:

- Make sure that `SET CURRENT QUERY ACCELERATION = ALL` is the first SQL statement executed in the RJDBC or RODBC session. This way forces all queries in the session to execute in the Accelerator rather than in the IBM DB2 for z/OS database.
- If you need to perform projections, joins, aggregates or other operations that can be expressed with SQL, they should be performed by the database rather than the R program. Transferring large data sets from the database to the computer that runs the R program can be time-intensive and computation is usually much slower than in the database. Especially when forcing acceleration when one can make use of the Accelerator infrastructure that is highly optimized to perform those operations.
- Whenever an analytical or statistical function is available in the IBM Netezza Analytics Server, use the corresponding stored procedure rather than the R function to execute it. The analytical and statistical functions of IBM Netezza Analytics are executed directly on the accelerator in parallel. This is usually faster than fetching the data from the database and executing the R functions on the data.
- Using RODBC instead of RJDBC is usually faster. Fetching result sets with RJDBC can be enhanced by setting the number of rows to fetch with the `fetch(JDBCResult, rows)` function set to a `-1` value. When set to the value of `-1`, the driver fetches chunks of up to 512K rows at one time.

A

ODBC connectivity between SPSS Modeler and DB2 for z/OS

SPSS Modeler must communicate with DB2 for z/OS. This appendix provides some basic terminology and context and then gives two alternatives of how to configure the Open Database Connectivity (ODBC) layer. ODBC is a standard application programming interface (API) for client applications to access a database management system (DBMS).

© Copyright IBM Corp. 2016. All rights reserved.

ODBC related terms and concepts

An ODBC driver is a driver used for database connectivity. It allows the client application to communicate with the database management system (DBMS) using an interoperable protocol that is independent from the used programming language of the client application and also from the used DBMS. Basically, an ODBC driver is similar to a device driver for a piece of hardware, with the ODBC Driver Manager orchestrating the registered ODBC drivers for different data sources in a way that is similar to the operating system orchestrating the installed device drivers for the used hardware components.

An ODBC Data Source Name (DSN) represents a connection to a database. The related data structure contains information such as DB2 location name, IP address, port number, and so on.

ODBC Driver Manager

In Microsoft Windows, an ODBC Driver Manager is integrated as part of the operating system. Driver manager configuration can be started by selecting the Windows **Start** menu and then typing `ODBC` in the Search field, as shown in Figure A-1.

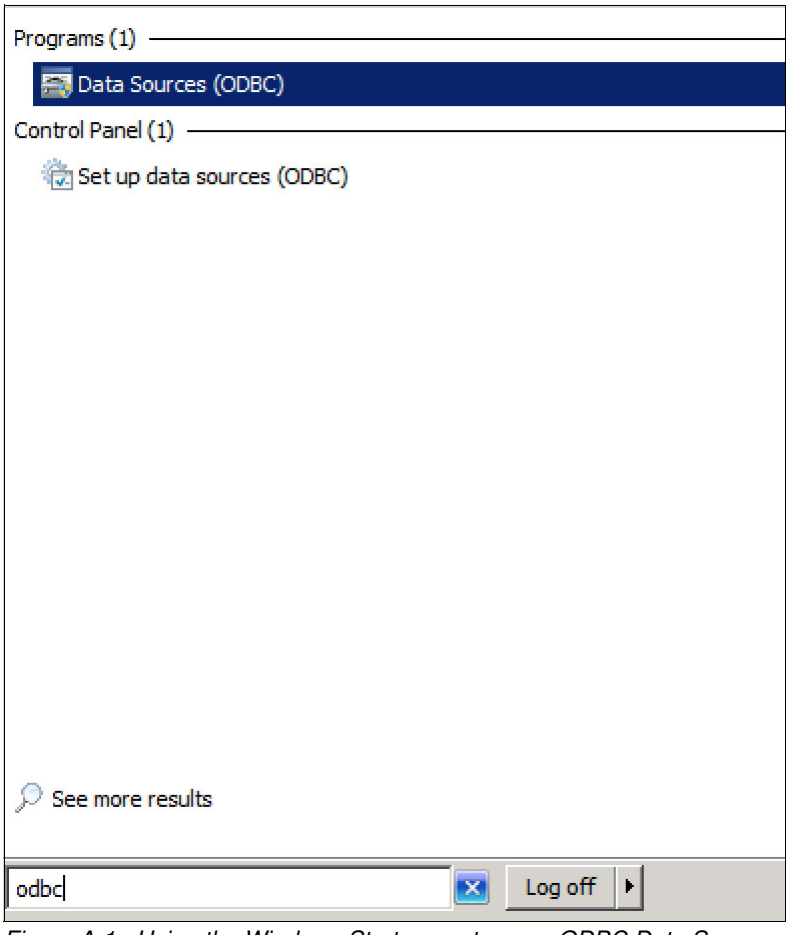


Figure A-1 Using the Windows Start menu to open ODBC Data Sources program

The Windows configuration program opens (Figure A-2).

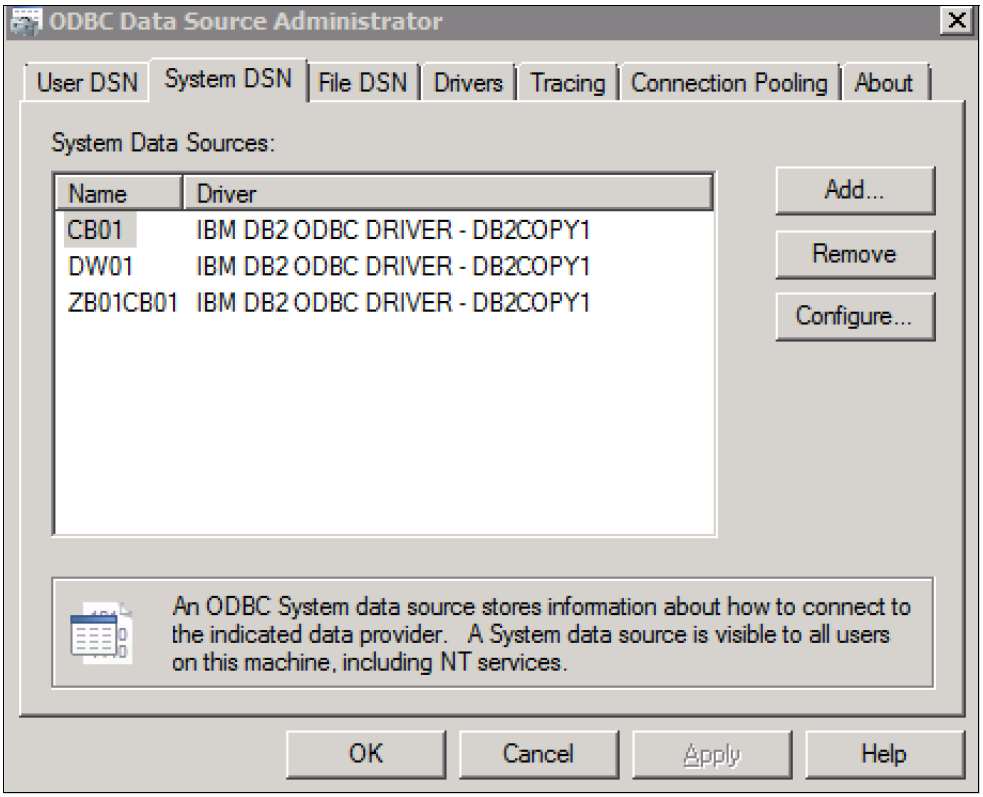


Figure A-2 Windows ODBC Data Source Administrator allowing management of ODBC DSNs

If an ODBC client application (such as SPSS Modeler client or MicroStrategy) looks for ODBC data sources, it asks the ODBC Driver Manager, which manages all ODBC drivers found on the platform.

UNIX based operating systems do not include an integrated ODBC Driver Manager. SPSS SDAP contains an ODBC driver manager on UNIX platforms (based on Progress DataDirect). Other ODBC driver manager products are also available on UNIX.

ODBC gateway mode

If a DBMS is not performing the SQL query execution alone, but forwards the request to and the reply from a different DBMS, the used ODBC driver must support ODBC gateway mode. This is the case with all IBM DB2 drivers in service.

If connecting a client application to z/OS DB2 without using accelerator, gateway mode is not needed. If connecting to z/OS DB2 with accelerator, it is needed.

SPSS SDAP prior Version 7.1 shows an error similar to the following message:

```
[ODBC DB2 Wire Protocol driver] Unhandled CodePoint: 1C03
```

In SPSS Modeler, the `Unhandled Codepoint` message is displayed if DataDirect or SDAP ODBC driver not supporting DB2 Analytics Accelerator (with ODBC gateway mode) is used, as shown in Figure A-3.

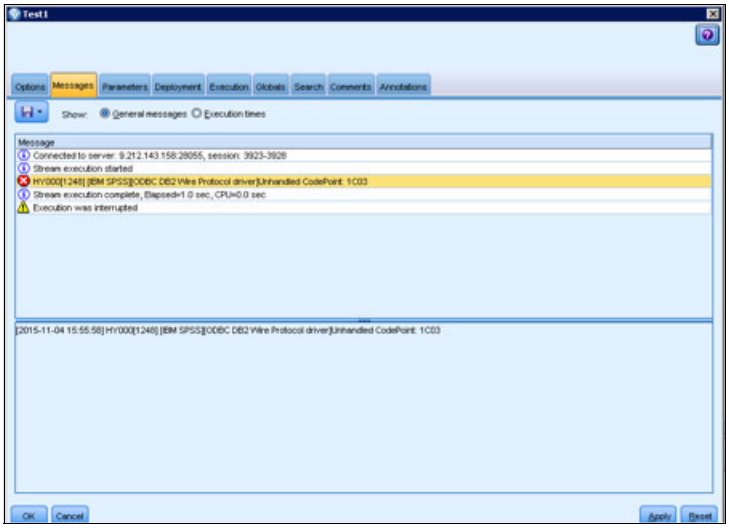


Figure A-3 SPSS Modeler message

Although Progress DataDirect does not support gateway mode, for this book, SDAP 7.11 worked to some degree. However, if being able to request service or being able to open a defect report if something is not working, use DB2 ODBC drivers for now.

The major disadvantage of DB2 ODBC drivers compared to SDAP is that they are slightly more difficult to install.

The generation of an efficient `INSERT FROM SELECT` SQL statement using SPSS SDAP 7.11, and exporting data from accelerated tables to a DB2 base table, with an AOT for intermediate results in between proved unsuccessful for our tests. The ODBC driver returned an error message to the SPSS Modeler server and the SPSS Modeler failed back to using `INSERT via VALUES` SQL, which is significantly less efficient.

Using the IBM DB2 10.5 for a logical unit of work (LUW) ODBC driver and the `INSERT FROM SELECT` SQL statements, this worked as expected and used the optimized code within DB2 to copy the data from the Accelerator to a base table.

We found that when using the SPSS SDAP driver, it removes error messages returned by IBM Netezza Analytics stored procedures. So, if doing in-database modeling and using DB2 ODBC driver, the system will report issues such as the record ID attribute does not contain unique values, or some other attribute contains NULL values, although that is not supported by the used IBM Netezza Analytics algorithm. If using SPSS SDAP driver, only the information that something went wrong with the model build is reported to the user.[1]

To visualize this, we created a stream with invalid input data and a database modeling DB2 z/OS Decision Tree node. Figure A-4 on page 169 indicates what is displayed if using DB2 ODBC driver (in this case, the user learns from the message that the used column `"AMOUNT"` in the input stream either does not contain unique values or it contains NULL values). Actually that is the attribute used as the Record ID, but we deliberately chose an attribute that is not unique.

[1] IBM Netezza Analytics error reporting works only with SPSS Modeler 18; it does not perform in SPSS Modeler 17.1.

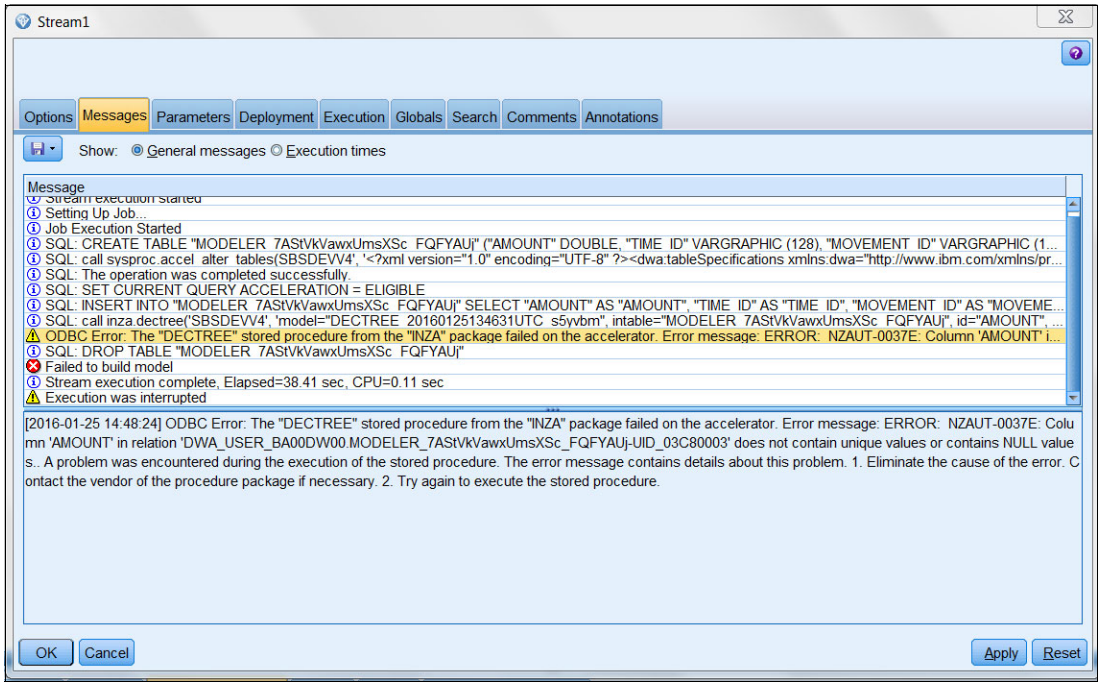


Figure A-4 Error reporting if using IBM DB2 ODBC driver

Running the same stream by using SDAP ODBC driver produced the result shown in Figure A-5.

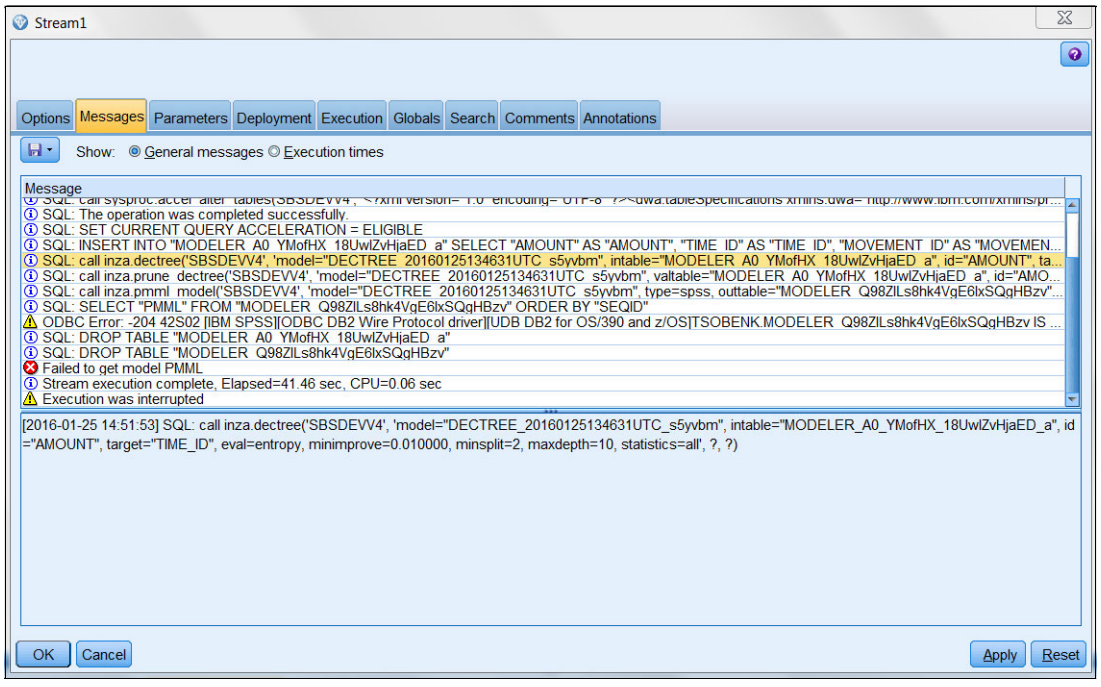


Figure A-5 Error reporting if using IBM SDAP ODBC driver (not supported for DB2 Analytics Accelerator)

Here, the SPSS Modeler does not understand from the communication with the database that the DECTREE stored procedure failed; it reports that the model build stored procedure completed successfully.

The other two IBM Netezza Analytics stored procedures (`PRUNE_DECTREE`, `PMML_MODEL`) fail also because they cannot find a model to work on, but that is also not reported. Only at the point SPSS Modeler tries to read the generated PMML model using an SQL SELECT query, it fails with a message that the specified table does not exist.

In summary, to avoid such issues, invest the time to install DB2 ODBC. SDAP works most of the time, but (in our tests) not always, and it is not supported with IDAA.

ODBC driver used by SPSS Modeler server

The ODBC driver used is always the one available within the operating system that hosts the modeler server. Although this might be evident, it is easy to forget.

Figure A-6 shows a flow chart of database connectivity from the perspective of the SPSS Modeler.

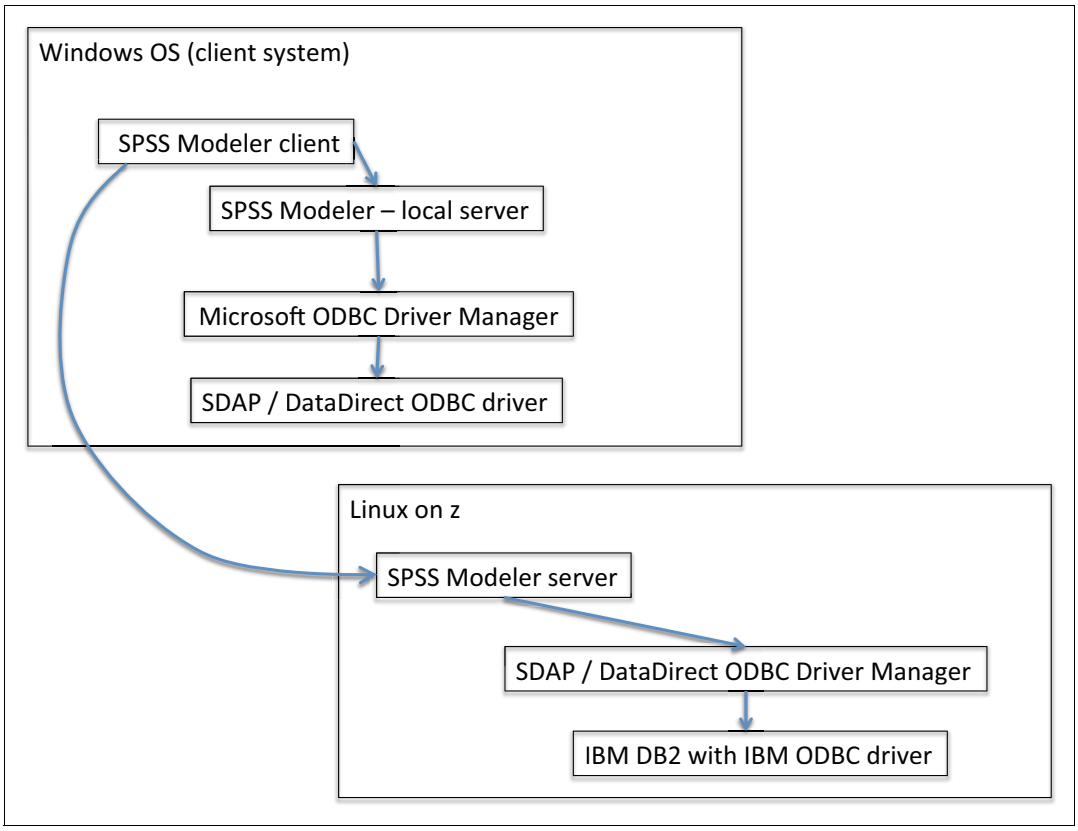


Figure A-6 Database connectivity by SPSS Modeler

In this sample configuration, the local server (which is used by default) uses the DataDirect ODBC driver through Microsoft ODBC Driver Manager from the Windows OS; the Linux on z based SPSS Modeler server used the DB2 ODBC driver.

Installation and configuration of IBM DB2 ODBC driver

Various DB2 LUW offerings contain the IBM DB2 ODBC driver, plus the stand-alone command-line interface (CLI) driver that has a smaller footprint (IBM Data Server Driver for ODBC/CLI) but also limited functionality. For SPSS Modeler connectivity, all DB2 ODBC or CLI drivers in service should work.

The setup of the environment for this publication uses the IBM DB2 for Linux, UNIX, and Windows version 10.5 client.

Base installation and DB2 instance create

We completed the base installation of the DBMS, including the db2inst1 instance, with the commands in the following procedure, using Linux on z Systems and SUSE Linux Enterprise Server 11 Linux distribution. As user root, the base installation of DB2 LUW used for the contained ODBC driver and the DB2 Communications Database (CDB) was as follows:

1. Extracted (**unpack**) the DB2 package.
2. Changed to (**cd**) the directory containing db2_install.
3. Used defaults to run **./db2install**.
4. Continued with the following commands:
   a. **groupadd -p 999 db2iadm1**
   b. **useradd -u 1004 -g db2iadm1 -m -d /home/db2inst1 db2inst1**
   c. **cd /opt/ibm/db2/V10.5/instance**
   d. **./db2icrt db2inst1 -s client -u db2inst1 -p db2inst1**

Note that **db2setup** can do some of these steps in the background, however it requires X11 for its graphical user interface (GUI), and not everyone has that available for server management.

The DB2 instance is just used for connectivity to SPSS Modeler server. Although having enough reachable free space in the file system for the modeler server process is important, this DB2 instance usually does not consume significant file system storage resources (except for potential error reporting), because it does not process significant amounts of persistent data on its own.

Starting DB2

To first start DB2, change the userid to `db2inst1` and type in `db2start`.

DB2 communication database and BIND

For every DB2 subsystem that should be reachable by SPSS Modeler server, an entry in the DB2 Communications Database is required, which corresponds to an ODBC Data Source Name (DSN), which is a data structure describing the data path and connectivity information related to that database connection.

The DB2 directory can be created as shown in Example A-1.

Example A-1 Connect from DB2 LUW to DB2 z/OS

```
db2inst1@zbapi:~> db2 catalog tcpip node MYNODE remote MYNODE.ibm.com server 446
ostype mvs
DB20000I  The CATALOG TCPIP NODE command completed successfully.
DB21056W  Directory changes may not be effective until the directory cache is
refreshed.
db2inst1@zbapi:~> db2 catalog database MYDB as MYALIAS at node MYNODE
authentication server
DB20000I  The CATALOG DATABASE command completed successfully.
DB21056W  Directory changes may not be effective until the directory cache is
refreshed.
db2inst1@zbapi:~> db2 catalog database MYDB as MYALIAS at node MYNODE
authentication server
DB20000I  The CATALOG DATABASE command completed successfully.
DB21056W  Directory changes may not be effective until the directory cache is
refreshed.
db2inst1@zbapi:~> db2 connect to MYDB user tsobenk
Enter current password for tsobenk:

   Database Connection Information

 Database server        = DB2 z/OS 11.1.5
 SQL authorization ID   = TSOBENK
 Local database alias   = MYDB
```

Next, do a BIND by going to the DB2 subdirectory that contains the bind scripts and running the command shown in Example A-2.

Example A-2 MVS BIND from DB2 LUW

```
db2inst1@zbapi:~> cd /opt/ibm/db2/V10.5/bnd
db2inst1@zbapi:~> db2 bind '@ddcsmvs.lst' blocking all sqlerror
 continue grant public action add

LINE    MESSAGES FOR ddcsmvs.lst
------  ------------------------------------------------------------------------
        SQL0061W  The binder is in progress.

LINE    MESSAGES FOR db2clist.bnd
------  ------------------------------------------------------------------------
        SQL0038W  The bind option SQLERROR CONTINUE has been
                  activated since it is required when binding this DB2-supplied

                  list file to DB2/MVS, SQL/DS, or OS/400.
        SQL0038W  The bind option SQLERROR CONTINUE has been
                  activated since it is required when binding this DB2-supplied

                  list file to DB2/MVS, SQL/DS, or OS/400.

LINE    MESSAGES FOR ddcsmvs.lst
------  ------------------------------------------------------------------------
        SQL0091N  Binding was ended with "0" errors and "2" warnings.
```

DB2 Connect license

If you see error message SQL1598N (`An attempt to connect to the database server failed because of a licensing problem`), you must purchase and activate an IBM DB2 Connect™ license. If you installed an traditional DB2 version, you can use the **`db2licm`** utility to install the license, as shown in Example A-3.

Example A-3 Installing the DB2 Connect license

```
db2licm -a db2consv_zs.lic
```

SPSS Modeler configuration for use of DB2 ODBC driver

For SPSS Modeler, a requirement is that some DB2 libraries must be found by the dynamic loader. On current Linux distributions, this can be achieved by adding a `db2.conf` file to the `/etc/ld.so.conf.d/` subdirectory (as user root), with the contents pointing to the 64-bit library folder in DB2, as shown in Example A-4.

Example A-4 Contents of /etc/ld.so.conf.d/db2.conf file

```
/opt/ibm/db2/V10.5/lib64/
```

Next, complete the following steps:

1. As root, use the **`ldconfig`** command.
2. In the `modelersrv.sh` file, add lines similar to those shown in Example A-5.

   Example A-5 Part of modelersrv.sh file in SPSS Modeler server directory

   ```
   DB2_INSTALL_PATH=/opt/ibm/db2/V10.5
   export DB2_INSTALL_PATH
   DB2_INSTANCE_HOME=/home/db2inst1
   export DB2_INSTANCE_HOME
   # ...
   ODBC_DRIVER_MANAGER_PATH=/opt/ibm/db2/V10.5/lib64/
   ```

3. Go to the SPSS Modeler server installation directory
   (the default is `/usr/IBM/SPSS/ModelerServer/17.1` for version 17.1).
4. Go to the `bin/` subdirectory.
5. Change the used ODBC library shown in Example A-6.

   Example A-6 Changing symbolic links to the used SPSS ODBC library (for DB2 ODBC driver)

   ```
   rm libspssodbc.so
   ln -s libspssodbc_db2cli.so libspssodbc.so
   ```

6. Verify that all shared libraries are correctly resolved, similar to Example A-7.

Example A-7 Use "ldd" to check whether all libraries used by "libspssodbc.so" are found

```
zbapi:/usr/IBM/SPSS/ModelerServer/17.1/bin # ldd libspssodbc.so
   libdb2o.so.1 => /opt/ibm/db2/V10.5/lib64/libdb2o.so.1 (0x000003fffb59d000)
   libdl.so.2 => /lib64/libdl.so.2 (0x000003fffb598000)
   libunicodelib.so => /data/IBM/SPSS/ModelerServer/17.1/bin/./libunicodelib.so
(0x000003fffb58f000)
   libstdc++.so.6 => /usr/lib64/libstdc++.so.6 (0x000003fffb43c000)
   libm.so.6 => /lib64/libm.so.6 (0x000003fffb3a2000)
   libgcc_s.so.1 => /lib64/libgcc_s.so.1 (0x000003fffb38f000)
   libc.so.6 => /lib64/libc.so.6 (0x000003fffb204000)
   libcrypt.so.1 => /lib64/libcrypt.so.1 (0x000003fffb1c5000)
   libpthread.so.0 => /lib64/libpthread.so.0 (0x000003fffb1a6000)
   libpam.so.0 => /lib64/libpam.so.0 (0x000003fffb195000)
   libdb2dascmn.so.1 => /opt/ibm/db2/V10.5/lib64/libdb2dascmn.so.1
(0x000003fffb178000)
   libdb2g11n.so.1 => /opt/ibm/db2/V10.5/lib64/libdb2g11n.so.1 (0x000003fffab03000)
   libdb2genreg.so.1 => /opt/ibm/db2/V10.5/lib64/libdb2genreg.so.1
(0x000003fffaac8000)
   libdb2install.so.1 => /opt/ibm/db2/V10.5/lib64/libdb2install.so.1
(0x000003fffaac1000)
   libdb2locale.so.1 => /opt/ibm/db2/V10.5/lib64/libdb2locale.so.1
(0x000003fffaa9c000)
   libdb2osse.so.1 => /opt/ibm/db2/V10.5/lib64/libdb2osse.so.1 (0x000003fffa56d000)
   libdb2osse_db2.so.1 => /opt/ibm/db2/V10.5/lib64/libdb2osse_db2.so.1
(0x000003fffa516000)
   libdb2sdbin.so.1 => /opt/ibm/db2/V10.5/lib64/libdb2sdbin.so.1
(0x000003fffa48d000)
   libdb2trcapi.so.1 => /opt/ibm/db2/V10.5/lib64/libdb2trcapi.so.1
(0x000003fffa480000)
   /lib/ld64.so.1 (0x000002aace099000)
   libicuuc.so.51 => /data/IBM/SPSS/ModelerServer/17.1/bin/./libicuuc.so.51
(0x000003fffa2d6000)
   libaudit.so.0 => /lib64/libaudit.so.0 (0x000003fffa2ba000)
   librt.so.1 => /lib64/librt.so.1 (0x000003fffa2ae000)
   libicudata.so.51 => /data/IBM/SPSS/ModelerServer/17.1/bin/./libicudata.so.51
(0x000003fff8d63000)
```

7. If no errors are produced, start (or restart) the SPSS Modeler server.

After connecting to this server using SPSS Modeler client, you should see the DB2 ODBC based data sources.

Installation and configuration of SPSS SDAP driver

Use of SDAP driver is not yet officially supported. With SDAP 7.1, we know DB2 Analytics Accelerator does not work because of the missing support for ODBC gateway mode. Although SDAP 7.11 partially works, it is still unsupported. Setup is slightly less complicated than what is required for the IBM DB2 ODBC driver though, which is fully serviced. Issues we found with SPSS SDAP are that it is significantly slower in some special cases and that an important part of the error reporting is removed; as a result, developing SPSS streams doing in-database modeling is more difficult.

To install and configure the SPSS SDAP driver, complete the following steps:

1. Go to the SPSS Modeler server installation directory (for version 17.1, the default is `/usr/IBM/SPSS/ModelerServer/17.1`).
2. Go to the `bin/` subdirectory.
3. Change the used ODBC library, as shown in Example A-8.

Example A-8 Changing symbolic links to the used SPSS ODBC library (for SPSS SDAP or Progress DataDirect ODBC driver)

```
rm libspssodbc.so
ln -s libspssodbc_datadirect_utf16.so libspssodbc.so
```

4. In the `modelersrv.sh` file, make sure you source the `odbc.sh` file from the SDAP package, as shown in Example A-9 (assuming SDAP is installed to /root/SDAP711/).

Example A-9 File "modelersrv.sh" needs to source "SDAP711/odbc.sh" file

```
. /root/SDAP711/odbc.sh
```

5. In the SDAP directory, run the **`setodbcpath.sh`** script, then edit the `odbc.ini` file. In our environment, we used the file shown in Example A-10.

Example A-10 Sample odbc.ini file used for SDAP ODBC

```
ODBC Data Sources]
CBO1=Montpellier CBO1 - EBCDIC - z/OS DB2

[ODBC]
IANAAppCodePage=4
InstallDir=/root/SDAP711
Trace=0
TraceFile=odbctrace.out
TraceDll=/root/SDAP711/lib/XEtrc27.so

[CBO1]
Driver=/root/SDAP711/lib/XEdb227.so
Description=IBM Corp. 7.1 DB2 Wire Protocol
AccountingInfo=
AddStringToCreateTable=
AlternateID=
AlternateServers=
ApplicationName=
ApplicationUsingThreads=1
AuthenticationMethod=0
BulkBinaryThreshold=32
BulkCharacterThreshold=-1
BulkLoadBatchSize=1024
BulkLoadFieldDelimiter=
BulkLoadRecordDelimiter=
CatalogSchema=
CharsetFor65535=0
ClientHostName=
ClientUser=
#Collection applies to z/OS and iSeries only
Collection=
ConcurrentAccessResolution=0
```

```
ConnectionReset=0
ConnectionRetryCount=0
ConnectionRetryDelay=3
CurrentFuncPath=
#Database applies to DB2 UDB only
Database=<database_name>
DefaultIsolationLevel=1
DynamicSections=1000
EnableBulkLoad=0
EncryptionMethod=0
FailoverGranularity=0
FailoverMode=0
FailoverPreconnect=0
GrantAuthid=PUBLIC
GrantExecute=1
GSSClient=native
HostNameInCertificate=
IpAddress=200.1.1.20
KeyPassword=
KeyStore=
KeyStorePassword=
LoadBalanceTimeout=0
LoadBalancing=0
#Location applies to z/OS and iSeries only
Location=BA00CB00
LogonID=
MaxPoolSize=100
MinPoolSize=0
Password=
PackageCollection=NULLID
PackageNamePrefix=DD
PackageOwner=
Pooling=0
ProgramID=
QueryTimeout=0
ReportCodePageConversionErrors=0
TcpPort=446
TrustStore=
TrustStorePassword=
UseCurrentSchema=0
ValidateServerCertificate=1
WithHold=1
XMLDescribeType=-10
```

EBCDIC and z/OS DB2 are just comments, but viewable from SPSS Modeler client. In our installation (with complete `odbc.ini` file, where we show only the first entry), it looks similar to Figure A-7.

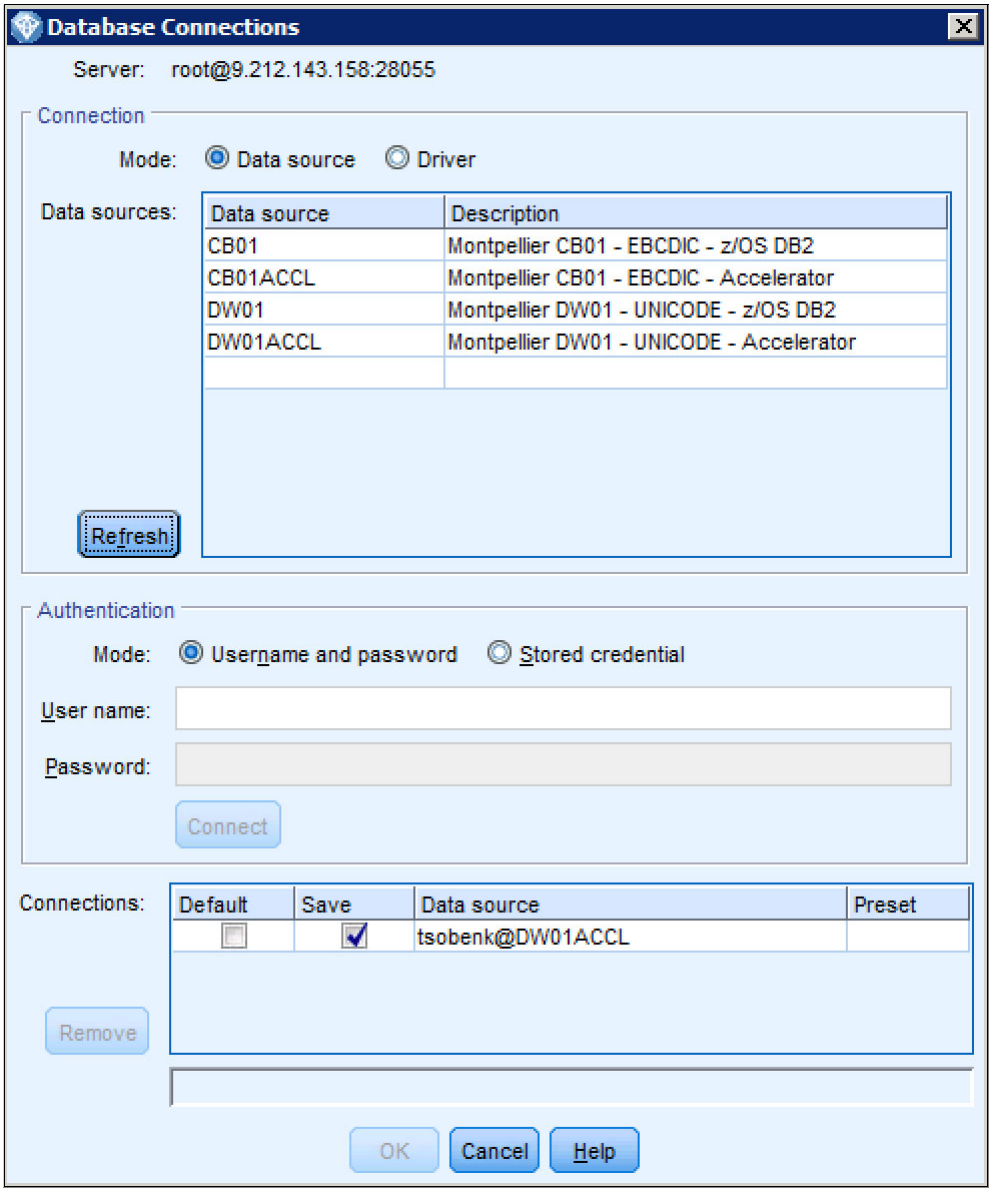


Figure A-7 SPSS Modeler client GUI shows available data sources if odbc.ini shown in Example A-10 on page 175 is used

B

Job to install wrapper stored procedures

The sample we used is listed in “JCL job sample” on page 180.

The in-database models are described in “IBM Netezza Analytics stored procedures” on page 190.

© Copyright IBM Corp. 2016. All rights reserved.

JCL job sample

The base we used to register the IBM Netezza Analytics Wrapper Stored Procedures is listed in Example B-1.

Example B-1 JCL job sample to install wrapper stored procedures

```
//INZADDL  JOB <job parameters>
//*********************************************************************
//* JOB NAME = INZADDL
//*
//* Licensed Materials - Property of IBM
//* 5697-DA5
//* (C) COPYRIGHT IBM Corp. 2015.
//*
//* US Government Users Restricted Rights
//* Use, duplication or disclosure restricted by GSA ADP Schedule
//* Contract with IBM Corporation
//*
//* DISCLAIMER OF WARRANTIES :
//* Permission is granted to copy and modify this  Sample code provided
//* that both the copyright  notice,- and this permission notice and
//* warranty disclaimer  appear in all copies and modified versions.
//*
//* THIS SAMPLE CODE IS LICENSED TO YOU AS-IS.
//* IBM  AND ITS SUPPLIERS AND LICENSORS  DISCLAIM ALL WARRANTIES,
//* EITHER EXPRESS OR IMPLIED, IN SUCH SAMPLE CODE, INCLUDING THE
//* WARRANTY OF NON-INFRINGEMENT AND THE IMPLIED WARRANTIES OF
//* MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT
//* WILL IBM OR ITS LICENSORS OR SUPPLIERS BE LIABLE FOR ANY DAMAGES
//* ARISING OUT OF THE USE OF OR INABILITY TO USE THE SAMPLE CODE OR
//* COMBINATION OF THE SAMPLE CODE WITH ANY OTHER CODE. IN NO EVENT
//* SHALL IBM OR ITS LICENSORS AND SUPPLIERS BE LIABLE FOR ANY LOST
//* REVENUE, LOST PROFITS OR DATA, OR FOR DIRECT, INDIRECT, SPECIAL,
//* CONSEQUENTIAL,INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND
//* REGARDLESS OF THE THEORY OF LIABILITY,-, EVEN IF IBM OR ITS
//* LICENSORS OR SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH
//* DAMAGES.
//*
//*    STATUS = Version 5
//*
//* DESCRIPTION: This JCL creates DB2 for z/OS objects
//*              required by
//*              IBM DB2 Analytics Accelerator
//*              for z/OS, Version 05.01.00.
//*
//*********************************************************************
//*
//*  Pseudocode =
//*    INZACOO  Step Create INZA Wrapper Procedures and
//*                  grant execute to PUBLIC.
//*
//*  Notes =
//*    PRIOR TO RUNNING THIS JOB, customize it for your system:
//*    (1) Review GRANT statements issued by this job and
//*        change GRANT TO PUBLIC if needed.
//*    (2) Add a valid job card
//*    (3) Locate and change all occurrences of the following strings
//*        as indicated:
```

```
//*        (A) The subsystem name '!DSN!' to the name of your DB2
//*        (B) 'DSN!!0' to the prefix of the target library for DB2
//*        (C) 'DSNTIA!!' to the plan name for DSNTIAD on your DB2
//*        (D) '!WLMENV!' to the name of the WLM environment for
//*            the stored procedures
//*
//*********************************************************************
//JOBLIB   DD  DISP=SHR,
//             DSN=DSN!!0.SDSNLOAD
//*
//INZAC00  EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//SYSTSPRT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSTSIN  DD  *
  DSN SYSTEM(!DSN!)
  RUN PROGRAM(DSNTIAD) PLAN(DSNTIA!!) -
  LIB('DSN!!0.RUNLIB.LOAD')
  END
//SYSIN    DD  *
CREATE PROCEDURE "INZA"."DECTREE" (
      IN ACCELERATOR_NAME VARCHAR(128) CCSID UNICODE FOR MIXED DATA,
      IN PARAMETER CLOB(65536) CCSID UNICODE FOR MIXED DATA,
      OUT RETVAL INTEGER,
      INOUT MESSAGE CLOB(65536) CCSID UNICODE FOR MIXED DATA)
   DYNAMIC RESULT SETS 2
   EXTERNAL NAME "AQT02ISP"
   LANGUAGE C
   MODIFIES SQL DATA
   PARAMETER STYLE DB2SQL
   FENCED
   WLM ENVIRONMENT !WLMENV!
   ASUTIME NO LIMIT
   COLLID SYSACCEL
    RUN OPTIONS
    'TRAP(ON),HEAP(32K,32K,ANY,KEEP,8K,4K),POSIX(ON),
    STACK(128K,128K,ANY,KEEP,512K,128K),BELOW(4K,4K,FREE),
    XPLINK(ON),ALL31(ON),ENVAR("_CEE_ENVFILE_S=DD:AQTENV")'
   STAY RESIDENT NO
   COMMIT ON RETURN N O
   PROGRAM TYPE MAIN
   SECURITY USER
   PARAMETER CCSID UNICODE;
GRANT EXECUTE ON PROCEDURE "INZA"."DECTREE" TO PUBLIC;

CREATE PROCEDURE "INZA"."DROP_ALL_MODELS" (
      IN ACCELERATOR_NAME VARCHAR(128) CCSID UNICODE FOR MIXED DATA,
      IN PARAMETER CLOB(65536) CCSID UNICODE FOR MIXED DATA,
      OUT RETVAL SMALLINT,
      INOUT MESSAGE CLOB(65536) CCSID UNICODE FOR MIXED DATA)
   DYNAMIC RESULT SETS 2
   EXTERNAL NAME "AQT02ISP"
   LANGUAGE C
   MODIFIES SQL DATA
   PARAMETER STYLE DB2SQL
   FENCED
   WLM ENVIRONMENT !WLMENV!
   ASUTIME NO LIMIT
   COLLID SYSACCEL
```

```
    RUN OPTIONS
    'TRAP(ON),HEAP(32K,32K,ANY,KEEP,8K,4K),POSIX(ON),
    STACK(128K,128K,ANY,KEEP,512K,128K),BELOW(4K,4K,FREE),
    XPLINK(ON),ALL31(ON),ENVAR("_CEE_ENVFILE_S=DD:AQTENV")'
   STAY RESIDENT NO
   COMMIT ON RETURN NO
   PROGRAM TYPE MAIN
   SECURITY USER
   PARAMETER CCSID UNICODE;
GRANT EXECUTE ON PROCEDURE "INZA"."DROP_ALL_MODELS" TO PUBLIC;

CREATE PROCEDURE "INZA"."DROP_MODEL" (
      IN ACCELERATOR_NAME VARCHAR(128) CCSID UNICODE FOR MIXED DATA,
      IN PARAMETER CLOB(65536) CCSID UNICODE FOR MIXED DATA,
      OUT RETVAL INT,
      INOUT MESSAGE CLOB(65536) CCSID UNICODE FOR MIXED DATA)
   DYNAMIC RESULT SETS 2
   EXTERNAL NAME "AQT02ISP"
   LANGUAGE C
   MODIFIES SQL DATA
   PARAMETER STYLE DB2SQL
   FENCED
   WLM ENVIRONMENT !WLMENV!
   ASUTIME NO LIMIT
   COLLID SYSACCEL
    RUN OPTIONS
    'TRAP(ON),HEAP(32K,32K,ANY,KEEP,8K,4K),POSIX(ON),
    STACK(128K,128K,ANY,KEEP,512K,128K),BELOW(4K,4K,FREE),
    XPLINK(ON),ALL31(ON),ENVAR("_CEE_ENVFILE_S=DD:AQTENV")'
   STAY RESIDENT NO
   COMMIT ON RETURN NO
   PROGRAM TYPE MAIN
   SECURITY USER
   PARAMETER CCSID UNICODE;
GRANT EXECUTE ON PROCEDURE "INZA"."DROP_MODEL" TO PUBLIC;

CREATE PROCEDURE "INZA"."GROW_DECTREE" (
      IN ACCELERATOR_NAME VARCHAR(128) CCSID UNICODE FOR MIXED DATA,
      IN PARAMETER CLOB(65536) CCSID UNICODE FOR MIXED DATA,
      OUT RETVAL INTEGER,
      INOUT MESSAGE CLOB(65536) CCSID UNICODE FOR MIXED DATA)
   DYNAMIC RESULT SETS 2
   EXTERNAL NAME "AQT02ISP"
   LANGUAGE C
   MODIFIES SQL DATA
   PARAMETER STYLE DB2SQL
   FENCED
   WLM ENVIRONMENT !WLMENV!
   ASUTIME NO LIMIT
   COLLID SYSACCEL
    RUN OPTIONS
    'TRAP(ON),HEAP(32K,32K,ANY,KEEP,8K,4K),POSIX(ON),
    STACK(128K,128K,ANY,KEEP,512K,128K),BELOW(4K,4K,FREE),
    XPLINK(ON),ALL31(ON),ENVAR("_CEE_ENVFILE_S=DD:AQTENV")'
   STAY RESIDENT NO
   COMMIT ON RETURN NO
   PROGRAM TYPE MAIN
   SECURITY USER
   PARAMETER CCSID UNICODE;
```

```
GRANT EXECUTE ON PROCEDURE "INZA"."GROW_DECTREE" TO PUBLIC;

CREATE PROCEDURE "INZA"."GROW_REGTREE" (
      IN ACCELERATOR_NAME VARCHAR(128) CCSID UNICODE FOR MIXED DATA,
      IN PARAMETER CLOB(65536) CCSID UNICODE FOR MIXED DATA,
      OUT RETVAL INTEGER,
      INOUT MESSAGE CLOB(65536) CCSID UNICODE FOR MIXED DATA)
   DYNAMIC RESULT SETS 2
   EXTERNAL NAME "AQTO2ISP"
   LANGUAGE C
   MODIFIES SQL DATA
   PARAMETER STYLE DB2SQL
   FENCED
   WLM ENVIRONMENT !WLMENV!
   ASUTIME NO LIMIT
   COLLID SYSACCEL
    RUN OPTIONS
    'TRAP(ON),HEAP(32K,32K,ANY,KEEP,8K,4K),POSIX(ON),
    STACK(128K,128K,ANY,KEEP,512K,128K),BELOW(4K,4K,FREE),
    XPLINK(ON),ALL31(ON),ENVAR("_CEE_ENVFILE_S=DD:AQTENV")'
   STAY RESIDENT NO
   COMMIT ON RETURN NO
   PROGRAM TYPE MAIN
   SECURITY USER
   PARAMETER CCSID UNICODE;
GRANT EXECUTE ON PROCEDURE "INZA"."GROW_REGTREE" TO PUBLIC;

CREATE PROCEDURE "INZA"."KMEANS" (
      IN ACCELERATOR_NAME VARCHAR(128) CCSID UNICODE FOR MIXED DATA,
      IN PARAMETER CLOB(65536) CCSID UNICODE FOR MIXED DATA,
      OUT RETVAL BIGINT,
      INOUT MESSAGE CLOB(65536) CCSID UNICODE FOR MIXED DATA)
   DYNAMIC RESULT SETS 2
   EXTERNAL NAME "AQTO2ISP"
   LANGUAGE C
   MODIFIES SQL DATA
   PARAMETER STYLE DB2SQL
   FENCED
   WLM ENVIRONMENT !WLMENV!
   ASUTIME NO LIMIT
   COLLID SYSACCEL
    RUN OPTIONS
    'TRAP(ON),HEAP(32K,32K,ANY,KEEP,8K,4K),POSIX(ON),
    STACK(128K,128K,ANY,KEEP,512K,128K),BELOW(4K,4K,FREE),
    XPLINK(ON),ALL31(ON),ENVAR("_CEE_ENVFILE_S=DD:AQTENV")'
   STAY RESIDENT NO
   COMMIT ON RETURN NO
   PROGRAM TYPE MAIN
   SECURITY USER
   PARAMETER CCSID UNICODE;
GRANT EXECUTE ON PROCEDURE "INZA"."KMEANS" TO PUBLIC;

CREATE PROCEDURE "INZA"."LIST_MODELS" (
      IN ACCELERATOR_NAME VARCHAR(128) CCSID UNICODE FOR MIXED DATA,
      IN PARAMETER CLOB(65536) CCSID UNICODE FOR MIXED DATA,
      OUT RETVAL INTEGER,
      INOUT MESSAGE CLOB(65536) CCSID UNICODE FOR MIXED DATA)
   DYNAMIC RESULT SETS 2
   EXTERNAL NAME "AQTO2ISP"
```

```
   LANGUAGE C
   MODIFIES SQL DATA
   PARAMETER STYLE DB2SQL
   FENCED
   WLM ENVIRONMENT !WLMENV!
   ASUTIME NO LIMIT
   COLLID SYSACCEL
    RUN OPTIONS
    'TRAP(ON),HEAP(32K,32K,ANY,KEEP,8K,4K),POSIX(ON),
    STACK(128K,128K,ANY,KEEP,512K,128K),BELOW(4K,4K,FREE),
    XPLINK(ON),ALL31(ON),ENVAR("_CEE_ENVFILE_S=DD:AQTENV")'
   STAY RESIDENT NO
   COMMIT ON RETURN NO
   PROGRAM TYPE MAIN
   SECURITY USER
   PARAMETER CCSID UNICODE;
GRANT EXECUTE ON PROCEDURE "INZA"."LIST_MODELS" TO PUBLIC;

CREATE PROCEDURE "INZA"."MODEL_EXISTS" (
      IN ACCELERATOR_NAME VARCHAR(128) CCSID UNICODE FOR MIXED DATA,
      IN PARAMETER CLOB(65536) CCSID UNICODE FOR MIXED DATA,
      OUT RETVAL INT,
      INOUT MESSAGE CLOB(65536) CCSID UNICODE FOR MIXED DATA)
   DYNAMIC RESULT SETS 2
   EXTERNAL NAME "AQT02ISP"
   LANGUAGE C
   MODIFIES SQL DATA
   PARAMETER STYLE DB2SQL
   FENCED
   WLM ENVIRONMENT !WLMENV!
   ASUTIME NO LIMIT
   COLLID SYSACCEL
    RUN OPTIONS
    'TRAP(ON),HEAP(32K,32K,ANY,KEEP,8K,4K),POSIX(ON),
    STACK(128K,128K,ANY,KEEP,512K,128K),BELOW(4K,4K,FREE),
    XPLINK(ON),ALL31(ON),ENVAR("_CEE_ENVFILE_S=DD:AQTENV")'
   STAY RESIDENT NO
   COMMIT ON RETURN NO
   PROGRAM TYPE MAIN
   SECURITY USER
   PARAMETER CCSID UNICODE;
GRANT EXECUTE ON PROCEDURE "INZA"."MODEL_EXISTS" TO PUBLIC;

CREATE PROCEDURE "INZA"."NAIVEBAYES" (
      IN ACCELERATOR_NAME VARCHAR(128) CCSID UNICODE FOR MIXED DATA,
      IN PARAMETER CLOB(65536) CCSID UNICODE FOR MIXED DATA,
      OUT RETVAL INTEGER,
      INOUT MESSAGE CLOB(65536) CCSID UNICODE FOR MIXED DATA)
   DYNAMIC RESULT SETS 2
   EXTERNAL NAME "AQT02ISP"
   LANGUAGE C
   MODIFIES SQL DATA
   PARAMETER STYLE DB2SQL
   FENCED
   WLM ENVIRONMENT !WLMENV!
   ASUTIME NO LIMIT
   COLLID SYSACCEL
    RUN OPTIONS
    'TRAP(ON),HEAP(32K,32K,ANY,KEEP,8K,4K),POSIX(ON),
```

```
    STACK(128K,128K,ANY,KEEP,512K,128K),BELOW(4K,4K,FREE),
    XPLINK(ON),ALL31(ON),ENVAR("_CEE_ENVFILE_S=DD:AQTENV")'
   STAY RESIDENT NO
   COMMIT ON RETURN NO
   PROGRAM TYPE MAIN
   SECURITY USER
   PARAMETER CCSID UNICODE;
GRANT EXECUTE ON PROCEDURE "INZA"."NAIVEBAYES" TO PUBLIC;

CREATE PROCEDURE "INZA"."PMML_MODEL" (
      IN ACCELERATOR_NAME VARCHAR(128) CCSID UNICODE FOR MIXED DATA,
      IN PARAMETER CLOB(65536) CCSID UNICODE FOR MIXED DATA,
      OUT RETVAL INTEGER,
      INOUT MESSAGE CLOB(65536) CCSID UNICODE FOR MIXED DATA)
   DYNAMIC RESULT SETS 2
   EXTERNAL NAME "AQTO2ISP"
   LANGUAGE C
   MODIFIES SQL DATA
   PARAMETER STYLE DB2SQL
   FENCED
   WLM ENVIRONMENT !WLMENV!
   ASUTIME NO LIMIT
   COLLID SYSACCEL
    RUN OPTIONS
    'TRAP(ON),HEAP(32K,32K,ANY,KEEP,8K,4K),POSIX(ON),
    STACK(128K,128K,ANY,KEEP,512K,128K),BELOW(4K,4K,FREE),
    XPLINK(ON),ALL31(ON),ENVAR("_CEE_ENVFILE_S=DD:AQTENV")'
   STAY RESIDENT NO
   COMMIT ON RETURN NO
   PROGRAM TYPE MAIN
   SECURITY USER
   PARAMETER CCSID UNICODE;
GRANT EXECUTE ON PROCEDURE "INZA"."PMML_MODEL" TO PUBLIC;

CREATE PROCEDURE "INZA"."PREDICT_DECTREE" (
      IN ACCELERATOR_NAME VARCHAR(128) CCSID UNICODE FOR MIXED DATA,
      IN PARAMETER CLOB(65536) CCSID UNICODE FOR MIXED DATA,
      OUT RETVAL BIGINT,
      INOUT MESSAGE CLOB(65536) CCSID UNICODE FOR MIXED DATA)
   DYNAMIC RESULT SETS 2
   EXTERNAL NAME "AQTO2ISP"
   LANGUAGE C
   MODIFIES SQL DATA
   PARAMETER STYLE DB2SQL
   FENCED
   WLM ENVIRONMENT !WLMENV!
   ASUTIME NO LIMIT
   COLLID SYSACCEL
    RUN OPTIONS
    'TRAP(ON),HEAP(32K,32K,ANY,KEEP,8K,4K),POSIX(ON),
    STACK(128K,128K,ANY,KEEP,512K,128K),BELOW(4K,4K,FREE),
    XPLINK(ON),ALL31(ON),ENVAR("_CEE_ENVFILE_S=DD:AQTENV")'
   STAY RESIDENT NO
   COMMIT ON RETURN NO
   PROGRAM TYPE MAIN
   SECURITY USER
   PARAMETER CCSID UNICODE;
GRANT EXECUTE ON PROCEDURE "INZA"."PREDICT_DECTREE" TO PUBLIC;
```

```
CREATE PROCEDURE "INZA"."PREDICT_KMEANS" (
      IN ACCELERATOR_NAME VARCHAR(128) CCSID UNICODE FOR MIXED DATA,
      IN PARAMETER CLOB(65536) CCSID UNICODE FOR MIXED DATA,
      OUT RETVAL BIGINT,
      INOUT MESSAGE CLOB(65536) CCSID UNICODE FOR MIXED DATA)
   DYNAMIC RESULT SETS 2
   EXTERNAL NAME "AQTO2ISP"
   LANGUAGE C
   MODIFIES SQL DATA
   PARAMETER STYLE DB2SQL
   FENCED
   WLM ENVIRONMENT !WLMENV!
   ASUTIME NO LIMIT
   COLLID SYSACCEL
    RUN OPTIONS
    'TRAP(ON),HEAP(32K,32K,ANY,KEEP,8K,4K),POSIX(ON),
    STACK(128K,128K,ANY,KEEP,512K,128K),BELOW(4K,4K,FREE),
    XPLINK(ON),ALL31(ON),ENVAR("_CEE_ENVFILE_S=DD:AQTENV")'
   STAY RESIDENT NO
   COMMIT ON RETURN NO
   PROGRAM TYPE MAIN
   SECURITY USER
   PARAMETER CCSID UNICODE;
GRANT EXECUTE ON PROCEDURE "INZA"."PREDICT_KMEANS" TO PUBLIC;

CREATE PROCEDURE "INZA"."PREDICT_NAIVEBAYES" (
      IN ACCELERATOR_NAME VARCHAR(128) CCSID UNICODE FOR MIXED DATA,
      IN PARAMETER CLOB(65536) CCSID UNICODE FOR MIXED DATA,
      OUT RETVAL INTEGER,
      INOUT MESSAGE CLOB(65536) CCSID UNICODE FOR MIXED DATA)
   DYNAMIC RESULT SETS 2
   EXTERNAL NAME "AQTO2ISP"
   LANGUAGE C
   MODIFIES SQL DATA
   PARAMETER STYLE DB2SQL
   FENCED
   WLM ENVIRONMENT !WLMENV!
   ASUTIME NO LIMIT
   COLLID SYSACCEL
    RUN OPTIONS
    'TRAP(ON),HEAP(32K,32K,ANY,KEEP,8K,4K),POSIX(ON),
    STACK(128K,128K,ANY,KEEP,512K,128K),BELOW(4K,4K,FREE),
    XPLINK(ON),ALL31(ON),ENVAR("_CEE_ENVFILE_S=DD:AQTENV")'
   STAY RESIDENT NO
   COMMIT ON RETURN NO
   PROGRAM TYPE MAIN
   SECURITY USER
   PARAMETER CCSID UNICODE;
GRANT EXECUTE ON PROCEDURE "INZA"."PREDICT_NAIVEBAYES" TO PUBLIC;

CREATE PROCEDURE "INZA"."PREDICT_REGTREE" (
      IN ACCELERATOR_NAME VARCHAR(128) CCSID UNICODE FOR MIXED DATA,
      IN PARAMETER CLOB(65536) CCSID UNICODE FOR MIXED DATA,
      OUT RETVAL INTEGER,
      INOUT MESSAGE CLOB(65536) CCSID UNICODE FOR MIXED DATA)
   DYNAMIC RESULT SETS 2
   EXTERNAL NAME "AQTO2ISP"
   LANGUAGE C
   MODIFIES SQL DATA
```

```
   PARAMETER STYLE DB2SQL
   FENCED
   WLM ENVIRONMENT !WLMENV!
   ASUTIME NO LIMIT
   COLLID SYSACCEL
    RUN OPTIONS
    'TRAP(ON),HEAP(32K,32K,ANY,KEEP,8K,4K),POSIX(ON),
    STACK(128K,128K,ANY,KEEP,512K,128K),BELOW(4K,4K,FREE),
    XPLINK(ON),ALL31(ON),ENVAR("_CEE_ENVFILE_S=DD:AQTENV")'
   STAY RESIDENT NO
   COMMIT ON RETURN NO
   PROGRAM TYPE MAIN
   SECURITY USER
   PARAMETER CCSID UNICODE;
GRANT EXECUTE ON PROCEDURE "INZA"."PREDICT_REGTREE" TO PUBLIC;

CREATE PROCEDURE "INZA"."PREDICT_TWOSTEP" (
      IN ACCELERATOR_NAME VARCHAR(128) CCSID UNICODE FOR MIXED DATA,
      IN PARAMETER CLOB(65536) CCSID UNICODE FOR MIXED DATA,
      OUT RETVAL BIGINT,
      INOUT MESSAGE CLOB(65536) CCSID UNICODE FOR MIXED DATA)
   DYNAMIC RESULT SETS 2
   EXTERNAL NAME "AQTO2ISP"
   LANGUAGE C
   MODIFIES SQL DATA
   PARAMETER STYLE DB2SQL
   FENCED
   WLM ENVIRONMENT !WLMENV!
   ASUTIME NO LIMIT
   COLLID SYSACCEL
    RUN OPTIONS
    'TRAP(ON),HEAP(32K,32K,ANY,KEEP,8K,4K),POSIX(ON),
    STACK(128K,128K,ANY,KEEP,512K,128K),BELOW(4K,4K,FREE),
    XPLINK(ON),ALL31(ON),ENVAR("_CEE_ENVFILE_S=DD:AQTENV")'
   STAY RESIDENT NO
   COMMIT ON RETURN NO
   PROGRAM TYPE MAIN
   SECURITY USER
   PARAMETER CCSID UNICODE;
GRANT EXECUTE ON PROCEDURE "INZA"."PREDICT_TWOSTEP" TO PUBLIC;

CREATE PROCEDURE "INZA"."PRUNE_DECTREE" (
      IN ACCELERATOR_NAME VARCHAR(128) CCSID UNICODE FOR MIXED DATA,
      IN PARAMETER CLOB(65536) CCSID UNICODE FOR MIXED DATA,
      OUT RETVAL INTEGER,
      INOUT MESSAGE CLOB(65536) CCSID UNICODE FOR MIXED DATA)
   DYNAMIC RESULT SETS 2
   EXTERNAL NAME "AQTO2ISP"
   LANGUAGE C
   MODIFIES SQL DATA
   PARAMETER STYLE DB2SQL
   FENCED
   WLM ENVIRONMENT !WLMENV!
   ASUTIME NO LIMIT
   COLLID SYSACCEL
    RUN OPTIONS
    'TRAP(ON),HEAP(32K,32K,ANY,KEEP,8K,4K),POSIX(ON),
    STACK(128K,128K,ANY,KEEP,512K,128K),BELOW(4K,4K,FREE),
    XPLINK(ON),ALL31(ON),ENVAR("_CEE_ENVFILE_S=DD:AQTENV")'
```

```
   STAY RESIDENT NO
   COMMIT ON RETURN NO
   PROGRAM TYPE MAIN
   SECURITY USER
   PARAMETER CCSID UNICODE;
GRANT EXECUTE ON PROCEDURE "INZA"."PRUNE_DECTREE" TO PUBLIC;

CREATE PROCEDURE "INZA"."PRUNE_REGTREE" (
      IN ACCELERATOR_NAME VARCHAR(128) CCSID UNICODE FOR MIXED DATA,
      IN PARAMETER CLOB(65536) CCSID UNICODE FOR MIXED DATA,
      OUT RETVAL INTEGER,
      INOUT MESSAGE CLOB(65536) CCSID UNICODE FOR MIXED DATA)
   DYNAMIC RESULT SETS 2
   EXTERNAL NAME "AQT02ISP"
   LANGUAGE C
   MODIFIES SQL DATA
   PARAMETER STYLE DB2SQL
   FENCED
   WLM ENVIRONMENT !WLMENV!
   ASUTIME NO LIMIT
   COLLID SYSACCEL
    RUN OPTIONS
    'TRAP(ON),HEAP(32K,32K,ANY,KEEP,8K,4K),POSIX(ON),
    STACK(128K,128K,ANY,KEEP,512K,128K),BELOW(4K,4K,FREE),
    XPLINK(ON),ALL31(ON),ENVAR("_CEE_ENVFILE_S=DD:AQTENV")'
   STAY RESIDENT NO
   COMMIT ON RETURN NO
   PROGRAM TYPE MAIN
   SECURITY USER
   PARAMETER CCSID UNICODE;
GRANT EXECUTE ON PROCEDURE "INZA"."PRUNE_REGTREE" TO PUBLIC;

CREATE PROCEDURE "INZA"."REGTREE" (
      IN ACCELERATOR_NAME VARCHAR(128) CCSID UNICODE FOR MIXED DATA,
      IN PARAMETER CLOB(65536) CCSID UNICODE FOR MIXED DATA,
      OUT RETVAL INTEGER,
      INOUT MESSAGE CLOB(65536) CCSID UNICODE FOR MIXED DATA)
   DYNAMIC RESULT SETS 2
   EXTERNAL NAME "AQT02ISP"
   LANGUAGE C
   MODIFIES SQL DATA
   PARAMETER STYLE DB2SQL
   FENCED
   WLM ENVIRONMENT !WLMENV!
   ASUTIME NO LIMIT
   COLLID SYSACCEL
    RUN OPTIONS
    'TRAP(ON),HEAP(32K,32K,ANY,KEEP,8K,4K),POSIX(ON),
    STACK(128K,128K,ANY,KEEP,512K,128K),BELOW(4K,4K,FREE),
    XPLINK(ON),ALL31(ON),ENVAR("_CEE_ENVFILE_S=DD:AQTENV")'
   STAY RESIDENT NO
   COMMIT ON RETURN NO
   PROGRAM TYPE MAIN
   SECURITY USER
   PARAMETER CCSID UNICODE;
GRANT EXECUTE ON PROCEDURE "INZA"."REGTREE" TO PUBLIC;

CREATE PROCEDURE "INZA"."SPLIT_DATA" (
      IN ACCELERATOR_NAME VARCHAR(128) CCSID UNICODE FOR MIXED DATA,
```

```
      IN PARAMETER CLOB(65536) CCSID UNICODE FOR MIXED DATA,
      OUT RETVAL DOUBLE,
      INOUT MESSAGE CLOB(65536) CCSID UNICODE FOR MIXED DATA)
   DYNAMIC RESULT SETS 2
   EXTERNAL NAME "AQT02ISP"
   LANGUAGE C
   MODIFIES SQL DATA
   PARAMETER STYLE DB2SQL
   FENCED
   WLM ENVIRONMENT !WLMENV!
   ASUTIME NO LIMIT
   COLLID SYSACCEL
    RUN OPTIONS
    'TRAP(ON),HEAP(32K,32K,ANY,KEEP,8K,4K),POSIX(ON),
    STACK(128K,128K,ANY,KEEP,512K,128K),BELOW(4K,4K,FREE),
    XPLINK(ON),ALL31(ON),ENVAR("_CEE_ENVFILE_S=DD:AQTENV")'
   STAY RESIDENT NO
   COMMIT ON RETURN NO
   PROGRAM TYPE MAIN
   SECURITY USER
   PARAMETER CCSID UNICODE;
GRANT EXECUTE ON PROCEDURE "INZA"."SPLIT_DATA" TO PUBLIC;

CREATE PROCEDURE "INZA"."TWOSTEP" (
      IN ACCELERATOR_NAME VARCHAR(128) CCSID UNICODE FOR MIXED DATA,
      IN PARAMETER CLOB(65536) CCSID UNICODE FOR MIXED DATA,
      OUT RETVAL BIGINT,
      INOUT MESSAGE CLOB(65536) CCSID UNICODE FOR MIXED DATA)
   DYNAMIC RESULT SETS 2
   EXTERNAL NAME "AQT02ISP"
   LANGUAGE C
   MODIFIES SQL DATA
   PARAMETER STYLE DB2SQL
   FENCED
   WLM ENVIRONMENT !WLMENV!
   ASUTIME NO LIMIT
   COLLID SYSACCEL
    RUN OPTIONS
    'TRAP(ON),HEAP(32K,32K,ANY,KEEP,8K,4K),POSIX(ON),
    STACK(128K,128K,ANY,KEEP,512K,128K),BELOW(4K,4K,FREE),
    XPLINK(ON),ALL31(ON),ENVAR("_CEE_ENVFILE_S=DD:AQTENV")'
   STAY RESIDENT NO
   COMMIT ON RETURN NO
   PROGRAM TYPE MAIN
   SECURITY USER
   PARAMETER CCSID UNICODE;
GRANT EXECUTE ON PROCEDURE "INZA"."TWOSTEP" TO PUBLIC;

//*
```

IBM Netezza Analytics stored procedures

The following sections provide information about the five in-database models as referenced in 4.3, “Nodes supporting SQL generation for DB2 Accelerator” on page 88 and in the following publications:

- *IBM Netezza Analytics Release 3.2.0.0, In-Database Analytics Developer’s Guide* (November 20, 2014)
- *IBM Netezza Analytics Release 3.0.1, IBM Netezza In-Database Analytics Reference Guide*

K-means

The K-means algorithm is the most widely used clustering algorithm that uses an explicit distance measure to partition the data set into clusters.

The main concept behind the K-means algorithm is to represent each cluster by the vector of mean attribute values of all training instances assigned to that cluster, called the cluster’s center. There are direct consequences of such a cluster representation:

- The algorithm handles continuous attributes only, although workarounds for discrete attributes are possible.
- Both the cluster formation and cluster modeling processes can be performed in a computationally efficient way by applying the specified distance function to match instances against cluster centers.

The algorithm operates by performing several iterations of the same basic process. Each training instance is assigned to the closest cluster with respect to the specified distance function, applied to the instance and cluster center. All cluster centers are then recalculated as the mean attribute value vectors of the instances assigned to particular clusters.

The cluster centers are initialized by randomly picking *k* training instances, where *k* is the desired number of clusters. The iterative process should terminate when there are either no or sufficiently few changes in cluster assignments. In practice, however, specifying the number of iterations, typically a number in the range of 3 - 36, is sufficient.

The range of reasonable distance (or dissimilarity) measures is the same for the kNN algorithm, including all the standard difference-based measures: Euclidean, Manhattan, Canberra, and Maximum. For k-means, two additional distance measures are available: Norm_Euclidean and Mahalanobis. The Norm_Euclidean measure is the default used for k-means clustering.

This stored procedure builds a K-means clustering model. The model is saved to the database in a set of tables and registered in the database model metadata. Use the Model Management functions to further manipulate the model or access the model tables.

The clusters are built iteratively by partitioning data into *k* (or less) separate clusters according to their distance to the cluster center, then recalculating the cluster centers. The K-means clustering algorithm stops after <maxiter> iterations.

The output table <outtable> is created with the following columns: id, cluster_id, distance. The id column matches the <id> column of the input table. Each input table record is associated with a cluster, where the distance from the record to the cluster center is the smallest. The cluster ID and the distance to the cluster center are given in the columns cluster_id and distance.

Examples

Examples are as follows:

```
CALL nza..KMEANS('model=adult_mdl, intable=nza..adult,
outtable=adult_out, id=id, target=income, transform=S,
distance=euclidean, k=3, maxiter=5, randseed=12345,
idbased=false');
CALL nza..DROP_MODEL('model=adult_mdl');
CALL nza..DROP_TABLE('adult_out');
KMEANS
--------
3
(1 row)
DROP_MODEL
------------
t
(1 row)
DROP_TABLE
------------
t
(1 row)
```

Naive Bayes

The naive Bayes classifier is a simpler classification algorithm than most, which makes it quick and easy to apply. Although it does not compete with more sophisticated algorithms with respect to classification accuracy, in some cases it is able to deliver similar results in a fraction of the computation time.[1]

This stored procedure builds a naive Bayes model. The model is saved to the database in a set of tables and registered in the database model metadata. Use the Model Management functions to further manipulate the model or access the model tables.

Building a naive Bayes model works only on nominal <target> column. Only VARCHAR and NVARCHAR columns are treated as nominal attributes. Discretization is applied on numeric attributes according to the <disc> and <bins> parameters.

The output of the building process is a contingency table that gathers statistics about the class labels per attribute and attribute value. The contingency table has following columns: attributename, val, class, classvalcount, classcount, totalcount.

This algorithm also supports missing data.

[1] This information is from the *IBM Netezza Analytics Release 3.2.0.0, In-Database Analytics Developer's Guide*.

Examples

Examples are as follows:

```
CALL nza..NAIVEBAYES('model=NB_soybean,
intable=nza..soybean_train, id=instance, target=class');
CALL nza..DROP_MODEL('model=NB_soybean');
NAIVEBAYES
------------
1881
(1 row)
DROP_MODEL
------------
t
(1 row)
CALL nza..NAIVEBAYES('model=NB_iris, intable=nza..iris,
coldeftype=cont,
incolumn=PETALWIDTH:nom;PETALLENGTH:ignore;class:nom:targ
et;id:id');
CALL nza..DROP_MODEL('model=NB_iris');
NAIVEBAYES
------------
126
(1 row)
DROP_MODEL
------------
t
(1 row)
```

Decision Tree

A decision tree is a hierarchical structure that represents a classification model using a “divide and conquer” approach. Internal tree nodes represent splits applied to decompose the data set into subsets, and terminal nodes, also referred to as leaves, assign class labels to sufficiently small or uniform subsets. Splits are specified by logical conditions based on selected single attributes, with a separate outgoing branch corresponding to each possible outcome.

The concept of decision tree construction is to select splits that decrease the impurity of class distribution in the resulting subsets of instances, and increase the domination of one or more classes over the others. The goal is to find a subset containing only or mostly instances of one class after a small number of splits, so that a leaf with that class label is created. This approach promotes simple trees, which typically generalize better.

Creating and using decision tree models involves three major algorithmic sub-tasks:

- Decision tree growing
- Decision tree pruning
- Decision tree prediction

This Decision Tree stored procedure builds a decision tree model by growing and (optionally) pruning the tree. The model is saved to the database in a set of tables and registered in the database model metadata. Use the Model Management functions to further manipulate the model or access the model tables.

A top-down tree-growing algorithm is used with the following features:

- Binary splits (equality-based for nominal attributes, inequality-based for continuous attributes).
- No missing value handling.
- Entropy or Gini index for split evaluation.
- Stop criteria is satisfied at a node with a uniform class.
- Stop criteria is satisfied when further splits do not improve the class impurity by at least <minimprove>.
- Stop criteria is satisfied when the number of instances is less than <minsplit>.
- Stop criteria is satisfied when the tree depth reaches <maxdepth>.

A bottom-up reduced error pruning algorithm is used. It bases on the prediction accuracy of the model against the validation data set. The pruning is activated when parameter <valtable> is specified.

Examples

Examples are as follows:

```
CALL nza..DECTREE('model=adult_tree, intable=nza..adult_train,
id=id, target=income, minsplit=1000, eval=entropy,
valtable=nza..adult_prune, qmeasure=wAcc');
CALL nza..DROP_MODEL('model=adult_tree');
DECTREE
---------
13
(1 row)
DROP_MODEL
------------
t
(1 row)
```

Regression Tree

Regression trees are decision trees adapted to the regression task; regression trees store numeric target attribute values instead of class labels in leaves, and use appropriately modified split selection and stop criteria.

As with decision trees, regression tree nodes decompose the data into subsets, and regression tree leaves correspond to sufficiently small or sufficiently uniform subsets. Splits are selected to decrease the dispersion of target attribute values, so that they can be reasonably well predicted by their mean values at leaves. The resulting model is piecewise-constant, with fixed predicted values assigned to regions to which the domain is decomposed by the tree structure.[2]

Creating and using regression tree models involves three major algorithmic subtasks:

- Regression tree growing
- Regression tree pruning
- Regression tree prediction

[2] This information is from *IBM Netezza Analytics Release 3.2.0.0, In-Database Analytics Developer's Guide*.

This Regression Tree stored procedure builds a regression tree model. The model is saved to the database in a set of tables and registered in the database model metadata. Use the Model Management functions to further manipulate the model or access the model tables.

A top-down tree-growing algorithm is used with the following features:

- Binary splits (equality-based for nominal attributes, inequality-based for continuous attributes).
- No missing value handling.
- Variance for split evaluation.
- Stop criteria is satisfied when further splits do not improve the variance by at least <minimprove>.
- Stop criteria is satisfied when the number of instances is less than <minsplit>.
- Stop criteria is satisfied when the tree depth reaches <maxdepth>.

Examples

Examples are as follows:

```
CALL nza..GROW_REGTREE('model=wrt, intable=nza..weatherr,
id=instance, target=grade, minsplit=2, maxdepth=3');
CALL nza..DROP_MODEL('model=wrt');
GROW_REGTREE
--------------
7
(1 row)
DROP_MODEL
------------
t
(1 row)
```

TwoStep

TwoStep clustering is a data mining algorithm for large data sets. It is faster than traditional methods because it typically scans a data set only once before it saves the data to a clustering feature (CF) tree. TwoStep clustering can make clustering decisions without repeated data scans, whereas other clustering methods scan all data points, which requires multiple iterations. Non-uniform points are not gathered, so each iteration requires a reinspection of each data point, regardless of the significance of the data point. Because TwoStep clustering treats dense areas as a single unit and ignores pattern outliers, it provides high-quality clustering results without exceeding memory constraints.

The TwoStep algorithm has the following advantages:

- It automatically determines the optimal number of clusters. You do not have to manually create a different clustering model for each number of clusters.
- It detects input columns that are not useful for the clustering process. These columns are automatically set to supplementary. Statistics are gathered for these columns but they do not influence the clustering algorithm.
- The configuration of the CF tree can be granular, so that you can balance between memory usage and model quality, according to the environment and needs.

To cluster data, the TwoStep clustering algorithm does the following actions:

1. The algorithm scans all data and builds a CF tree. This tree is built by arranging the input records in a way so that similar records become part of the same tree node. If a memory issue occurs, the tree is rebuilt with an increased threshold and outliers are removed.
2. The leaves of the CF tree are clustered hierarchically in memory. The clustering is done by calculating the n * (n-1) / 2 distances between each pair of leaves and merging the two clusters with the smallest distance. The process of calculating distances between clusters and merging the closest two is repeated until the root of the tree is reached. All data is contained in one cluster, thus forming a binary tree.

   Starting with the root node of the binary tree, the child node with the worst quality is added. The process of adding child nodes continues until the number of clusters that is determined automatically or that is specified by the user is reached. The number of clusters that is determined automatically is also the optimal number of clusters.
3. The clustering result is refined by a final pass over the data where each record is assigned to the closest cluster. This behavior is similar to the behavior of the K-means algorithm.

This TwoStep stored procedure builds a TwoStep clustering model. The model is saved to the database in a set of tables and registered in the database model metadata. Use the Model Management functions to further manipulate the model or access the model tables.

The clustering model is built in two steps:

1. The input data records are distributed into a balanced tree according to their distance to the data records already in the tree node. The tree size is limited: outliers are removed and similar tree nodes are merged. Then, k clusters are determined out of the tree.
2. The input data records are assigned again to the nearest of the k clusters. The output table <outtable> is created with following columns: id, cluster_id, distance. The id column matches the <id> column of the input table. Each input table record is associated with a cluster, where the distance from the record to the cluster is the smallest. The cluster ID and the distance to the cluster are given in the columns cluster_id and distance.

Examples

Examples are as follows:

```
CALL nza..TWOSTEP('model=adult_mdl, intable=nza..adult,
id=id, target=income');
CALL nza..DROP_MODEL('model=adult_mdl');
TWOSTEP
---------
6
(1 row)
DROP_MODEL
------------
t
(1 row)
```

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. Some publications referenced in this list might be available in softcopy only.

- *Accelerating Data Transformation with IBM DB2 Analytics Accelerator for z/OS,* SG24-8314
- *Fraud Detection Blueprint on IBM System z*, TIPS-1035
- *Real-time Fraud Detection Analytics on IBM System z, SG24-8066*
- *Risk Scoring for a Loan Application on IBM System z*: Running IBM SPSS Real-Time Analytics, SG24-8153

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Other publications

These publications are also relevant as further information sources:

- *IBM Netezza Analytics Release 3.2.0.0, In-Database Analytics Developer's Guide* (November 20, 2014)
- *IBM Netezza Analytics Release 3.0.1, IBM Netezza In-Database Analytics Reference Guide*

Online resources

You can also find online information at these web pages:

- Predictive analytics using IBM SPSS Modeler in DB2 for z/OS:

  http://www.ibm.com/developerworks/data/library/techarticle/dm-1505predictive-spss-db2zos/index.html

- Architect Real-Time Analytics Into Operational Systems for Competitive Advantage:

  http://www.ibmsystemsmag.com/mainframe/Business-Strategy/BI-and-Analytics/milliseconds-competitive/

- Retailer Optimizes Analytics to Drive a More Customized Client Experience:

  http://www.ibmsystemsmag.com/mainframe/Business-Strategy/BI-and-Analytics/client-experience/

© Copyright IBM Corp. 2016. All rights reserved.

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Redbooks

Enabling Real-time Analytics on IBM z Systems Platform

(0.2”spine)
0.17”<->0.473”
90<->249 pages

SG24-8272-00

ISBN 0738441864

Printed in U.S.A.

Get connected