

Complete Analytics with IBM DB2 Query Management Facility

Accelerating Well-Informed Decisions Across the Enterprise

Gain actionable insight with visual reports and interactive dashboards

Deploy cost-effective, self-service analytics with zero coding

Leverage existing System z applications in new ways



Kristi Ramey
Mike Biere
Peter Richardson
Shawn Sullivan
Jeremy Weatherall

Redbooks



International Technical Support Organization

**Complete Analytics with IBM DB2 Query Management
Facility: Accelerating Well-Informed Decisions
Across the Enterprise**

August 2012

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

First Edition (August 2012)

This edition applies to Version 10 Release 1 of IBM DB2 Query Management Facility (QMF) Classic Edition and Enterprise Edition, which are features of both IBM DB2 10 for z/OS (5605-DB2) and IBM DB2 Version 9.1 for z/OS (5635-DB2). This information also applies to Version 10 Release 1 of IBM DB2 QMF Classic Edition Value Unit Edition (VUE). QMF Classic Edition VUE is a feature of DB2 10 for z/OS Value Unit Edition (5697-P31). QMF Enterprise Edition VUE is a feature of both DB2 10 for z/OS Value Unit Edition (5697-P31) and DB2 Version 9.1 for z/OS Value Unit Edition (5697-P12).

© Copyright International Business Machines Corporation 2012. 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	ix
Trademarks	x
Preface	xi
The team who wrote this book	xi
Now you can become a published author, too!	xii
Comments welcome	xiii
Stay connected to IBM Redbooks	xiii
Chapter 1. Assessing business analytics solutions	1
1.1 The value of business analytics	2
1.2 The underlying components of an enterprise business analytics solution	2
1.2.1 Bet-your-business operating platform	3
1.2.2 Centralized architecture	3
1.2.3 Sound data modeling and database management	4
1.2.4 Mapping of features and functions to end-user requirements	5
1.3 An overview of IBM business analytics with QMF	5
1.4 Evaluating total cost of ownership	11
1.5 The Spiffy Insurance Corporation: A fictitious company with real business needs	12
Chapter 2. Business analytics from the ground up: Hardware, data modeling, and data warehousing	15
2.1 The need for speed: IBM System z processors and IBM DB2 Analytics Accelerator ..	16
2.1.1 The z196 processor: Optimal design for enterprise analytics	16
2.1.2 Extensibility with the zEnterprise BladeCenter	17
2.1.3 Maximum query performance with the IBM DB2 Analytics Accelerator	18
2.1.4 The System z Integrated Facility for Linux for support of Linux on System z	19
2.2 Good decisions start with good data: Warehousing with DB2 for z/OS	19
2.2.1 Why a solid data warehousing solution is critical for business analytics	19
2.2.2 Why DB2 for z/OS is ideal for data warehousing	21
2.2.3 IBM industry data models	21
2.2.4 IBM InfoSphere Information Server	23
2.2.5 The IBM InfoSphere Warehouse	24
Chapter 3. DB2 for z/OS as an analytics engine	27
3.1 Advantages for analytics in DB2 10	28
3.1.1 CPU savings for lower total cost of ownership	28
3.1.2 Advancements in scalability	28
3.1.3 Reduced catalog lock contention	28
3.1.4 64-bit virtual storage relief	28
3.1.5 Bitemporal queries	29
3.1.6 Integrated XML support	31
3.1.7 Support for OLAP: Moving sums, averages, and aggregates	31
3.1.8 System Management Facility compression	32
3.1.9 Dynamic compression with INSERT	32
3.1.10 Dynamic SQL EXPLAIN	32
3.1.11 Instance-based statement hints	32
3.1.12 Dynamic SQL information	33
3.1.13 Query parallelism enhancements	33

3.1.14	Index enhancements	33
3.1.15	Buffer pool enhancements	34
3.1.16	Work file enhancements	35
3.1.17	Sort enhancements	35
3.1.18	Inline LOB support	35
3.1.19	Dynamic statement cache enhancements	36
3.2	Effective database management with IBM DB2 tools	36
Chapter 4.	Maximizing your existing System z investment.	39
4.1	Overview of QMF Classic capabilities	41
4.1.1	QMF Classic Version 10 enhancements	41
4.1.2	The QMF Classic user interface	42
4.1.3	Developing a query in QMF for TSO and CICS	43
4.1.4	Formatting the report	43
4.1.5	Charting your data	44
4.1.6	Developing procedures and applications	45
4.1.7	Authentication methods and security	46
4.1.8	Customizing the QMF work environment for users and groups	46
4.1.9	Connecting to remote databases in the DB2 family	47
4.2	Using QMF Classic perspective within QMF for Workstation	48
4.3	The QMF catalog: Accessing QMF Classic objects from QMF for Workstation	49
4.4	Directing work to System z from workstation environments	51
4.4.1	The RUNTSO command in QMF for Workstation	52
4.4.2	How to start QMF for TSO as a DB2 stored procedure	53
4.4.3	Running TSO applications through the stored procedure interface	56
4.4.4	Launching batch jobs through the stored procedure interface	56
4.5	Using QMF HPO to manage and administer the QMF Classic environment	59
4.5.1	Managing QMF objects and tracking and governing session activity	60
4.5.2	Optimizing resource-intensive operations	62
Chapter 5.	Installing QMF for Workstation, QMF for WebSphere, and touring the interfaces.	65
5.1	Deciding on your configuration	66
5.2	System requirements	67
5.3	Installing QMF for Workstation	68
5.4	Installing QMF for WebSphere	73
5.5	Touring the QMF for Workstation interface	76
5.5.1	Eclipse-based architecture	76
5.5.2	Views	77
5.5.3	Perspectives	78
5.5.4	Shared repository storage	84
5.5.5	Repositories	84
5.5.6	Workspaces	85
5.5.7	Repository connections	86
5.6	Touring the QMF for WebSphere interface	86
Chapter 6.	Configuring access to data sources and populating user workspaces	89
6.1	Creating shared repository storage	90
6.1.1	Defining connectivity to the database that will host the shared repository storage	90
6.1.2	Creating and configuring shared repository storage	96
6.2	Creating a repository within shared repository storage	103
6.3	Configuring access to data sources	106
6.3.1	Adding a relational data source	106
6.3.2	Adding a multidimensional data source	114

6.3.3 Adding a hierarchical data source	115
6.3.4 Adding a semi-structured data source	119
6.4 Populating the Repository Explorer	121
6.4.1 Populating user workspaces with content	126
6.4.2 Creating workspace links to content in the Repository Explorer	130
Chapter 7. Defining virtual data sources to reduce complexity for business users.	133
7.1 Benefits of the QMF metadata layer	134
7.2 Reducing data complexity with virtual tables	134
7.2.1 Creating a virtual table from a QMF query	136
7.2.2 Using a virtual table to simplify a table schema	141
7.3 Creating a virtual data source that federates data from multiple sources	143
Chapter 8. Configuring security	145
8.1 Security strategies	146
8.1.1 Database-based security	147
8.1.2 Internal security	158
8.1.3 LDAP security	166
8.1.4 Using no security	168
8.2 Minimizing the number of logins required	168
8.2.1 Automating the repository connection login	168
8.2.2 Using repository storage database credentials to log into the repository	169
8.2.3 Storing authentication information for each data source	170
8.2.4 Using repository storage database credentials to log into each data source	170
8.2.5 Mapping logins to group IDs	173
8.2.6 Using the personal repository	173
8.3 Use of security in dashboards	175
8.3.1 Integration methods for establishing security in dashboards	176
8.3.2 Restricting fourth quarter rows from the table (row-level security)	188
Chapter 9. Getting to the data you need: Query methods.	191
9.1 Queries against relational data	192
9.1.1 Developing queries using the query diagram view	194
9.1.2 Developing queries using the prompted query view	197
9.1.3 Developing queries using the SQL view	200
9.1.4 Calling stored procedures from QMF queries	201
9.2 Queries against hierarchical data	204
9.3 Transforming data using analytical queries	204
9.4 What can I do with my query results?	211
9.4.1 Manipulating data in the query results grid	211
9.4.2 Transferring query results and formatting to Microsoft Excel	215
9.4.3 Adding charts and graphs to visualize query data	216
9.4.4 Developing reports and dashboards from query data	219
9.4.5 Defining drill-down paths through query data	219
9.5 Queries against multidimensional data	222
Chapter 10. Updating data with the QMF table editor	227
10.1 Controlling access to the QMF table editor	228
10.2 Using the QMF table editor	228
Chapter 11. Creating reports	233
11.1 Creating classic reports	234
11.2 Creating visual reports	239
11.2.1 Getting started	241

11.2.2	The visual designer	243
11.2.3	Adding a heading with a graphic	247
11.2.4	Adding group summary highlighting and labels	254
11.2.5	Adding conditional formatting	261
11.2.6	Including data from additional queries	262
11.2.7	Including fixed pages in the report	263
Chapter 12.	Working with procedures	265
12.1	Creating a procedure	266
12.1.1	A sample linear procedure	267
12.1.2	A sample procedure with logic	271
12.2	How QMF finds objects referenced by the procedure	272
12.3	Uniquely referencing objects in the procedure	273
12.3.1	Using the Key property to point to the object	273
12.3.2	Referencing the object by its relative path	274
12.4	Scheduling a procedure	275
Chapter 13.	Analysis and forecasting functions	277
13.1	Using the QMF analytical functions	278
13.2	Forecasting outcomes	281
13.3	Embedding SPSS functions in QMF dashboards	286
Chapter 14.	Putting it all together: Developing dashboards	289
14.1	Basic elements of a QMF dashboard	290
14.2	How to create a dashboard	291
14.3	Scenario 1: Field agent dashboard (relational and hierarchical data) with three scenes (three data sources)	299
14.3.1	Creating a virtual data source	300
14.3.2	Creating a new dashboard	302
14.3.3	Creating data source connections	303
14.3.4	Specifying the first query that will supply data to the dashboard	304
14.3.5	Designing scene 1: Insurance business overview	305
14.3.6	Designing scene 2: Historic customer activity	314
14.3.7	Designing scene 3: Customer activity in the last 30 days	318
14.4	Scenario 2: Market analysis dashboard (OLAP and relational data) with one scene	320
14.4.1	Creating data source connections	320
14.4.2	Specifying the first query that will supply data to the dashboard	321
14.4.3	Designing the scene: Adding grid for sales by region with product type slicer	325
14.4.4	Designing the scene: Adding column chart for sales by time with a time slicer	330
14.4.5	Designing the scene: Adding a geospatial map driven by selections in the column chart	338
14.4.6	Designing the scene: Adding table driven by state selected on the map	351
Chapter 15.	Deploying created content to QMF users	373
15.1	Including QMF content in enterprise mashups	374
15.2	Deploying content to users by a web link	378
Appendix A.	Analytical functions available in the Expression Designer	383
A.1	Arithmetic functions	384
A.2	Color functions	384
A.3	Conversion functions	385
A.4	Data formatting functions	386
A.5	Date and time functions	387
A.6	Hierarchy functions	388

A.7 Information functions	388
A.8 Logical functions	389
A.9 Math and trigonometry functions	389
A.10 Measured functions	390
A.11 Security function	390
A.12 Spatial functions	391
A.13 Statistical functions	391
A.14 Text functions	392
A.15 Visual report functions	393
Related publications	395
IBM Redbooks publications	395
Other publications	395
Online resources	395
Help from IBM	396
Index	397

Notices

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

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

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

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

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

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

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

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

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

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


COPYRIGHT LICENSE:

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

Trademarks

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

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

CICS®	Informix®	RACF®
Cognos®	InfoSphere®	Redbooks®
DataStage®	Lotus Notes®	Redbooks (logo)  ®
DB2 Connect™	Lotus®	SPSS®
DB2®	Notes®	System z®
DRDA®	OMEGAMON®	Tivoli®
FICON®	Optim™	WebSphere®
GDDM®	POWER®	z/OS®
Guardium®	pureXML®	z/VM®
IBM®	QMF™	z10™
IMS™	Query Management Facility™	zEnterprise™

The following terms are trademarks of other companies:

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

Netezza, and N logo are trademarks or registered trademarks of Netezza Corporation, an IBM Company.

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.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

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

Preface

There is enormous pressure today for businesses across all industries to cut costs, enhance business performance, and deliver greater value with fewer resources. To take business analytics to the next level and drive tangible improvements to the bottom line, it is important to manage not only the volume of data, but the speed with which actionable findings can be drawn from a wide variety of disparate sources. The findings must be easily communicated to those responsible for making both strategic and tactical decisions. Historical, current, and predictive views of business data and operations are needed in real time at all levels of the organization. Yet, strained IT budgets require that the solution be self-service for everyone from DBAs to business users and be easily deployed to thin, browser-based clients.

Business analytics hosted in the IBM Query Management Facility™ (QMF™) on IBM DB2 and IBM System z allow you to tackle these challenges in a practical way using new features and functions that are easily deployed across the enterprise and easily consumed by business users who do not have prior IT experience. This IBM® Redbooks® publication provides step-by-step instructions on using these new features:

- ▶ Access to data that resides in any JDBC-compliant data source (DB2 family, Oracle, Informix®, SQL Server, IBM IMS™, and more)
- ▶ OLAP access through XMLA
- ▶ 150+ new analytical functions
- ▶ New metadata capabilities, including a feature that allows you to define virtual data sources, making it possible to simplify underlying database schemas for business users as well as present federated data as a single data source
- ▶ Graphical reports
- ▶ Graphical query interfaces
- ▶ Graphical, interactive dashboards
- ▶ Ability to integrate QMF functions with third-party applications
- ▶ Support for the IBM DB2 Analytics Accelerator, which provides rapid analysis of complex queries
- ▶ A new QMF Classic perspective, available within QMF for Workstation, which provides the same look and feel as that found in QMF for TSO and IBM CICS
- ▶ Capabilities that allow any client that can start a DB2 for z/OS stored procedure to start QMF for TSO, run a predefined query or procedure, and return up to 20 result sets to the calling program

The team who wrote this book

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

Kristi Ramey has spent 17 years in IT working as a technical writer, editor, and publications manager, as well as in marketing support roles, for various companies. Kristi joined IBM and QMF in 1991 as an information developer and is currently the Offering Team Lead for QMF.

Mike Biere has spent 34 years in IT, the majority spent in the areas of database, business intelligence/analytics, and data warehousing. He has been a technical consultant, product development manager, and worldwide support individual fulfilling a variety of roles within IBM, Ferguson Information Systems, and IBM Cognos®. Mike is a published author of two books on Business Intelligence as well as an author and contributor for numerous journal articles and white papers. His current role is worldwide technical marketing support for QMF.

Peter Richardson leads the Business Intelligence and Analytics business unit at Rocket Software, an IBM Business Partner. Prior to joining Rocket, he served as the Director of Engineering at SystemSoft, where he co-invented a self-healing technology that was widely shipped by PC OEMs before its acquisition by Microsoft in early 2000. He also has worked as a software consultant, playing a lead role in the development and deployment of enterprise software in the banking and retailing industries. Peter has over 20 years of experience in software development and over 12 years of experience working with the DB2 QMF family of products.

Shawn Sullivan has spent 14 years in IT, with the majority spent in the areas of QMF, DB2, and business intelligence/analytics. Shawn has been a technical consultant, trainer, and product manager, fulfilling a variety of roles for Rocket Software. His current role is Product Manager, Business Intelligence and Analytics.

Jeremy Weatherall has spent 15 years in IT, the majority spent in the areas of QMF, DB2, and business intelligence/analytics. He has been a lead developer and product specialist, fulfilling a variety of roles for Rocket Software. His current role is Product Specialist, Business Intelligence and Analytics.

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

Paolo Bruni
Whei-Jen Chen
Emma Jacobs
IBM ITSO, San Jose Center

Claudia Franco
Ronald Williams
IBM Silicon Valley Lab

Geof Reilly
Rocket Software

Willie Favero
IBM Software Group, Houston

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. 1WLB 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>



Assessing business analytics solutions

Business analytics has emerged as one of the key areas that many organizations look to as a means of unlocking information from myriad data sources, platforms, and formats. Information can be stored in traditional structured formats, semi-structured formats, or within pure text or other unstructured formats. Today, the key to data delivery is access and control: the ability to leverage information regardless of where it resides or the format in which it is stored.

The IT industry and global markets have been dramatically altered by forces both economic and technical in scale and scope. The ability to reach anyone, anywhere, and at any time has required many enterprises to rethink their infrastructure and ways of doing business. No other area of the business has been affected as much as IT. There are massive pressures to deliver greater value with lower cost, yet do so within the scope of using modern, flexible, and scalable products and services.

In this chapter, we cover how to evaluate business analytics solutions. We consider the following topics:

- ▶ The value of business analytics
- ▶ The underlying components of an enterprise business analytics solution
- ▶ An overview of IBM business analytics with QMF
- ▶ Evaluating total cost of ownership
- ▶ The Spiffy Insurance Corporation: A fictitious company with real business needs

1.1 The value of business analytics

Business analytics today can provide a wider and deeper set of capabilities than ever before. Traditional business intelligence has primarily been limited to a subset of individuals within an organization. It dealt primarily with what has already happened, which is akin to looking in a car's rear view mirror to see where you've already been. Today, the trend in analytics embraces predictive functions, allowing you to see where you should be headed in the context of where you have been.

There are many articles and publications that discuss the value of business analytics and the impact upon an individual or an enterprise. The greatest impact and success from intelligent application of business analytics occurs when there is a corporate goal with executive commitment. At times, far too much emphasis is placed on the cost of acquisition and not enough emphasis is placed on the business value that will result from effective use of the solution. Regardless of product, platform, or technology, the use of analytics has associated costs. However, when applied as the means to deliver a corporate vision and series of goals, it can completely alter the profitability and health of an entire organization.

Companies that successfully employ business analytics typically have the following attributes driving the implementation of the solution:

- ▶ Executive commitment and a clear mandate for expectations and return on investment
- ▶ A business-oriented implementation plan (not one based on platform bias or “favored” technologies) that includes a thorough cost analysis of the total solution
- ▶ Selection of a tool suite that delivers results as efficiently and as economically as possible
- ▶ A top-to-bottom data delivery strategy for all employees that could potentially be enabled and assisted by business analytics technology
- ▶ A deployment plan that rapidly enables the user community at all required levels
- ▶ A means to measure success against the plan, not merely evaluate the cost of the infrastructure

Today, self-service query, reporting, and analysis are much more important than ever before. Any business analytics solution must be personalized and capable of delivering data that is in context and relevant to the end user.

QMF now provides a full suite of business analytics that delivers on these requirements, addressing the corporate goal of putting analytical function into the hands of end users everywhere. It provides massive scalability at no incremental cost, thus providing a means to constrain costs without denying access to users within an enterprise.

1.2 The underlying components of an enterprise business analytics solution

The traditional description of a business analytics solution typically outlines the many features and functions of the tool itself with little mention of the many co-dependencies within the solution. Many business analytics solutions take the position that data is, well, just data, and the source is irrelevant to the tool. We have a vastly different view of business analytics. A successful business analytics implementation must have these four underlying components:

- ▶ A platform that is reliable, scalable, high performing, and highly secure
- ▶ A centralized architecture
- ▶ Sound data modeling and a reliable, high-performing database management system
- ▶ A road map for how each feature of the solution will meet end user requirements

1.2.1 Bet-your-business operating platform

In recent years, we have seen an evolution in analytics from rich clients with a heavy desktop presence to browser-based or mobile interfaces. As analytics tools deliver more and more function through thin-client interfaces, the greater the need becomes to ensure that the underlying server technology can truly serve well as the backbone of the solution. System z serves these needs with industry-leading capabilities in the areas of reliability, availability, performance, security, and scalability.

1.2.2 Centralized architecture

The most effective architecture for analytics is the one with the fewest number of moving parts. Testing has shown that the ideal scenario for any analytics tool is one where the tool is placed as close to the data as possible, preferably on the same platform.

Figure 1-1 depicts a high-level view of an enterprise business analytics solution that is centralized on System z. The greater the distance between the data and the analytics tools, the greater the overhead and subsequent cost of infrastructure. We envision the structure portrayed in Figure 1-1 as the “ideal” enterprise business analytics environment, where the data and analytics are centralized on System z as much as possible, resulting in lower overhead costs.

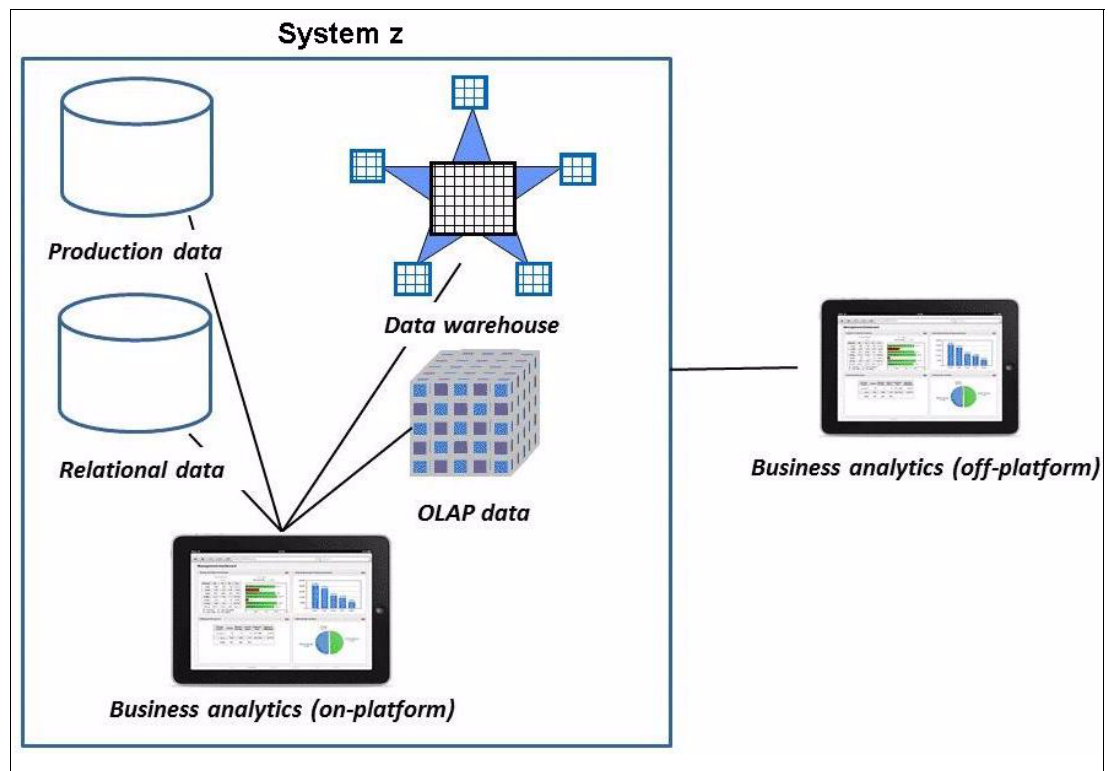


Figure 1-1 An ideal business analytics environment

An overwhelming number of installations today have set up multiple data sources, data marts, and data warehouses, with many also installing multiple analytics tools from different vendors. Compare Figure 1-1 here with Figure 1-2 next, where a decentralized architecture results not only in increased complexity of administration, but increased cost as well.

To draw all components, the picture would be too complex to interpret (see Figure 1-2). We can see it in many organizations, where pockets of analytics tools have evolved through individual departments acquiring point solutions, or where different generations of analytics tools have accumulated over time. Our recommendation is to examine your current analytics infrastructure to be sure that it aligns more closely with the architecture shown in Figure 1-1 on page 3.

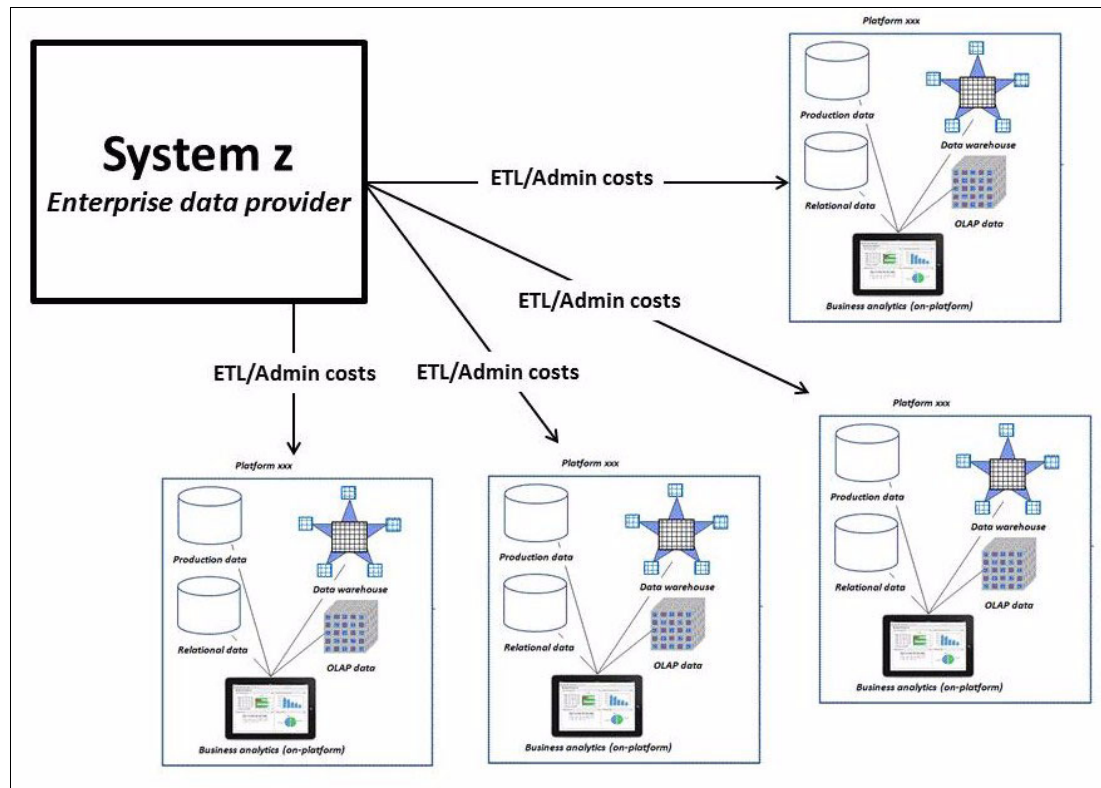


Figure 1-2 A decentralized business analytics architecture, which results in higher overhead costs

1.2.3 Sound data modeling and database management

The database does matter. There are fundamental elements that must be taken into consideration, such as scalability, reliability, accuracy, security, built-in functions, and performance. The “shape” of the data must match the end user requirements. For example, a star-schema structure with fact tables and dimensions is a common data warehouse format.

However, it is not a good design for end users requiring substantial text fields as part of their analysis and reporting output. More users want real-time information as well as highly detailed data that might not be held in a warehouse. Proper end user support today most often requires an amalgam of data from multiple sources. Later in this book, we discuss DB2 for IBM z/OS®, with its unique new features, as a key differentiator for enterprise business analytics.

Modeling of the data should be rapid as well as malleable as business requirements change. If the current shape of the data is not conducive to the analysis required, then tooling to make dramatic and immediate changes is essential to success. The majority of business analytics usage involves multidimensional elements within most queries. If additional dimensions are required or new elements need to be created, they must be supported by an infrastructure that lends itself to such change in an expedient manner.

1.2.4 Mapping of features and functions to end-user requirements

There are extensive checklists available for evaluating analytics tools, and you probably have at least one in house already. These lists typically have a set of necessary elements, such as the databases supported, access methods, and connectivity options, as well as a huge set of functional specifications (such as bar charts, pie charts, graphical reports, and so on). What we recommend is that you correlate the main areas of focus in any such evaluation with the actual requirements of the users in your organization so that “nice to have” elements do not take on unwarranted weight.

In cases where there is a strong drive for consolidation of servers, data, and analytics tools, the challenge is to select the tool that delivers maximum value at the lowest cost without sacrificing function. If an analytics tool is difficult to deploy and manage, or difficult for business users to use, the costs of administration and support far outweigh the savings in acquisition.

The new QMF provides a unique suite of functions, delivering query, reporting, business graphics, dashboards, customized user interfaces, and analytics across the enterprise. Next, we take a quick tour of these functions, then we cover each area in more detail in later topics.

1.3 An overview of IBM business analytics with QMF

Historically, QMF was created to provide an easy, powerful query and reporting tool for DB2 on z/OS. Today, QMF has been enormously expanded and enhanced since its first release many years ago, not only continuing strong support for DB2 on z/OS, but evolving into a family of products that offers industry-leading advantages in heterogeneous data access and presentation across a wide variety of platforms, databases, and browsers.

The QMF product family consists of four components:

- ▶ QMF for TSO and CICS®, which provides query, reporting, charting, procedure, and application development functions for users who work entirely from 3270 terminal emulators.
- ▶ QMF High Performance Option (HPO), a multifaceted tool that helps database administrators manage QMF objects, resources, and performance in a z/OS environment.
- ▶ QMF for Workstation, a rich desktop application that provides an environment within which objects such as queries, reports, procedures, and dashboards can be created, managed, and used. QMF for Workstation extends QMF functionality to the Windows, Linux, and Macintosh operating systems.
- ▶ QMF for IBM WebSphere®, the QMF family's browser-based portal to business information on demand. This component uses a pure HTML, thin-client deployment model, making it easy to provide the most frequently used QMF capabilities to large numbers of users quickly and easily.

Chapter 4, “Maximizing your existing System z investment” on page 39 covers the QMF for TSO, CICS, and HPO components in detail. Chapters 5 through 15 of this book are devoted to installing, configuring, and using QMF for Workstation and WebSphere, as well as deploying created content to broad audiences within your organization.

There are two main editions of QMF available, as listed here. Each edition is sold as a feature of DB2 for z/OS. Value-unit editions are also available:

- ▶ QMF Classic Edition: Offers QMF for the TSO and CICS environments.
- ▶ QMF Enterprise Edition: Consists of all four components in the QMF product family.

The role of QMF has historically been that of a query and reporting mainstay for a large number of customers. With the addition of new features and functions in Version 10, QMF now extends its reach even deeper into the end user community.

With QMF for Workstation and QMF for WebSphere, customers can now use their inventory of QMF Classic objects (queries, procedures, and forms) in dynamic, creative new ways. It thus protects significant existing System z investment by repurposing these objects for an entirely new class of users.

QMF 10 offers dramatically greater capabilities than in previous versions. Just one example of these enhancements is the capability to rapidly create and deploy interactive dashboards such as the one shown in Figure 1-3. Dashboards integrate data from a variety of sources and provide a unified display of relevant contextual information. Unlike reports, which tend to contain a fixed amount of information, dashboards are truly interactive, with the ability to deliver real-time information on demand, as needed by the dashboard user.

For example, an executive might need to see an operational summary across sales teams. Real-time color coding of data can be used to draw the executive's attention to areas of concern. Clicking problematic areas immediately produces dynamic reports that reveal the information underlying each area of concern.

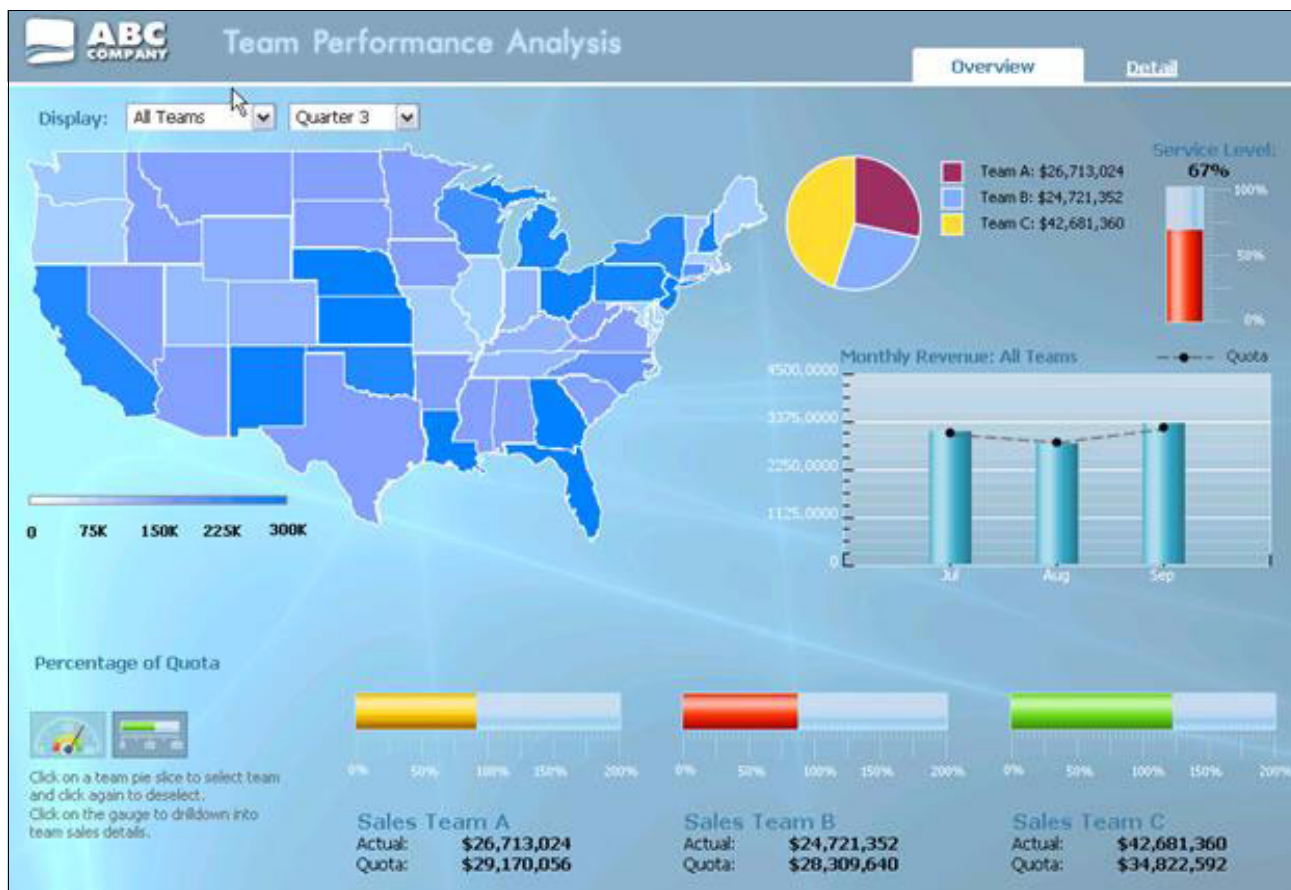


Figure 1-3 A QMF dashboard

Figure 1-4 shows another feature of QMF for Workstation and WebSphere, the query diagram interface. Several different query development methods are available within QMF to accommodate a wide variety of user knowledge levels. The query in Figure 1-4 shows data in the Q.STAFF and Q.ORG sample tables, which are part of a package of sample data that is shipped with both QMF for TSO and CICS as well as QMF for Workstation and WebSphere.

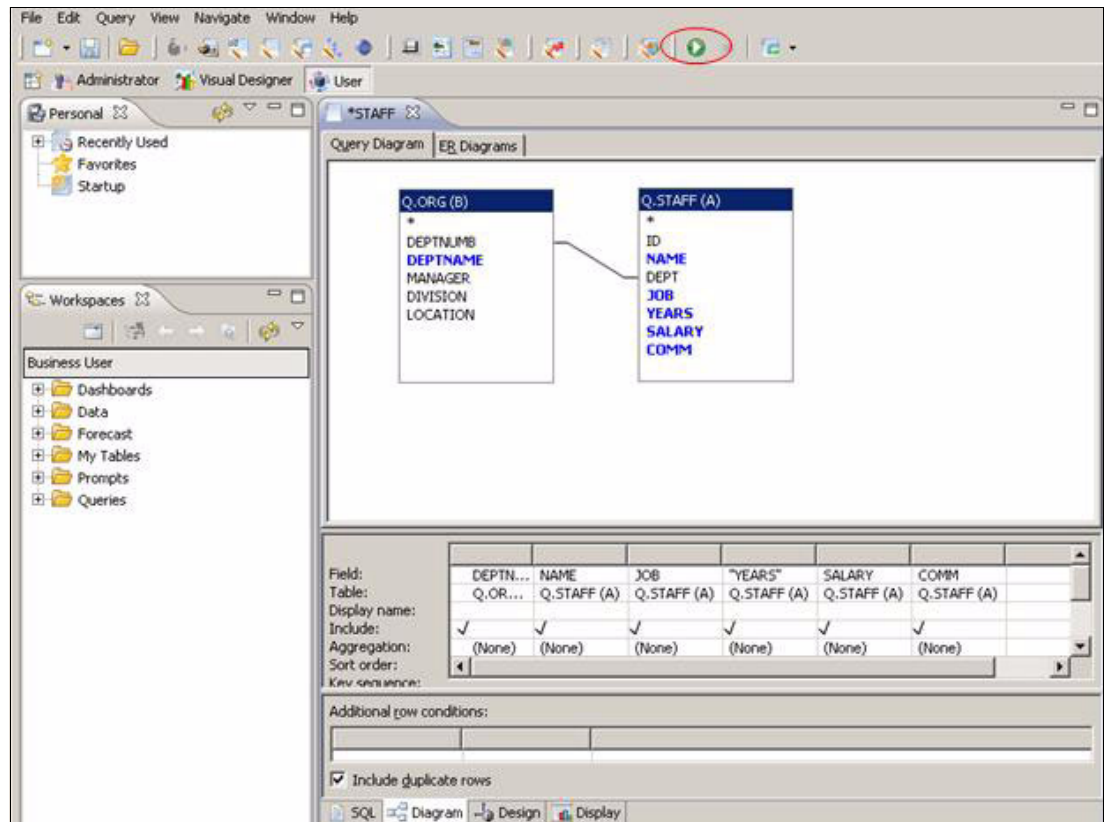


Figure 1-4 Query diagram view showing a join; the query is run using the Run Query icon in the toolbar

To build this query, we first select the Q.STAFF and Q.ORG tables and drag them into the query designer. It fits the paradigm of most modern analytics tools. Notice that the line between the tables connects the DEPT column with the DEPTNUMB column, creating a join between these two tables. After a join has been defined, QMF “remembers” it and will automatically join these tables with subsequent use. It is easy to override or delete the join conditions if it is not what you want. To run the query, simply click the Run Query icon, which is highlighted in the toolbar in Figure 1-4.

When the query completes, we see the results shown in Figure 1-5.

	1	2	3	4	5	6
	DEPTNAME	NAME	JOB	YEARS	SALARY	COMM
1	MID ATLANTIC	SANDERS	MGR	7	18357.50	0.00
2	MID ATLANTIC	PERNAL	SALES	8	18171.25	612.45
3	SOUTH ATLANTIC	MARENGHI	MGR	5	17506.75	0.00
4	SOUTH ATLANTIC	O'BRIEN	SALES	6	18006.00	846.55
5	NEW ENGLAND	HANES	MGR	10	20659.80	0.00
6	SOUTH ATLANTIC	QUIGLEY	SALES	5	16808.30	650.25
7	NEW ENGLAND	ROTHMAN	SALES	7	16502.83	1152.00
8	MID ATLANTIC	JAMES	CLERK	5	13504.60	128.20
9	GREAT LAKES	KOONITZ	SALES	6	18001.75	1386.70
10	GREAT LAKES	PLOTZ	MGR	7	18352.80	0.00
11	NEW ENGLAND	NGAN	CLERK	5	12508.20	206.60
12	SOUTH ATLANTIC	NAUGHTON	CLERK	5	12954.75	180.00
13	GREAT LAKES	YAMAGUCHI	CLERK	6	10505.90	75.60
14	PLAINS	FRAYE	MGR	6	21150.00	0.00
15	PLAINS	WILLIAMS	SALES	6	19456.50	637.65
16	HEAD OFFICE	MOLINARE	MGR	7	22959.20	0.00
17	NEW ENGLAND	KERMISCH	CLERK	4	12258.50	110.10
18	SOUTH ATLANTIC	ABRAHAMS	CLERK	3	12009.75	236.50
19	MID ATLANTIC	SNEIDER	CLERK	8	14252.75	126.50
20	GREAT LAKES	SCOUTTEN	CLERK	5	11508.60	84.20
21	HEAD OFFICE	LU	MGR	10	20010.00	0.00
22	PLAINS	SMITH	SALES	7	17654.50	992.80
23	PLAINS	LUNDQUIST	CLERK	3	13369.80	189.65

Figure 1-5 Query results, which are initially displayed in a grid

The data is initially displayed in a grid, which can be manipulated further using a number of options. Typical options include sorting, applying conditions (such as a color if some value is greater than a specified threshold), grouping, and aggregating data.

In Figure 1-6, we have created groupings on the JOB and DEPT columns and have also applied some additional formatting, such as adding currency signs and applying conditional formatting to highlight values that fall within a specified range.

To create a group, simply drag the column header to the far left until a vertical bar appears, at which time you can drop the column where desired to create the grouping. You can see many examples of how to perform more advanced functions in subsequent chapters.

File Edit Query Results View Navigate Window Help

Run command:

QMF User Visual Designer Administrator

Personal

Recently Used

Favorites

Startup

Workspaces

BusinessAnalystView

Relational Data Sources

Analyst Queries

APPLICANT_interview

Query1

SalesTotals

Staff Data - Formatted

Staff Data - Raw

Regional Data - Formatted

Regional Data - Raw

Dashboards

Reports

Staff Data - Formatted

	1	2	3	4	5	6	7	8		
	DEPT	JOB	NAME	YEARS	S	SALARY	COMM	TOTAL		
1	10	MGR	MOLINARE	7	SAMPLE	22959.20	0.00	\$22,959.20		
2			LU	10	SAMPLE	20010.00	0.00	\$20,010.00		
3			DANIELS	5	SAMPLE	19260.25	0.00	\$19,260.25		
4			JONES	12	SAMPLE	21234.00	0.00	\$21,234.00		
5			All values for 10							83463.45
6								83463.45		
7	15	CLERK	NGAN	5	SAMPLE	12508.20	206.60	\$12,714.80		
8			KERMISCH	4	SAMPLE	12258.50	110.10	\$12,368.60		
9										25083.40
10			MGR	HANES	10	SAMPLE	20659.80	0.00	\$20,659.80	
11										
12	SALES	ROTHMAN	7	SAMPLE	16502.83	1152.00	\$17,654.83			
13									17654.83	
14								63398.03		
15	20	CLERK	JAMES	0	SAMPLE	13504.60	128.20	\$13,632.80		
16			SNEIDER	8	SAMPLE	14252.75	126.50	\$14,379.25		
17										28012.05
18			MGR	SANDERS	7	SAMPLE	18357.50	0.00	\$18,357.50	
19										
20	SALES	PERIAL	8	SAMPLE	18171.25	612.45	\$18,783.70			
21									18783.70	
22								65153.25		
23	38	CLERK	NAUGHTON	0	SAMPLE	12954.75	180.00	\$13,134.75		
24			ABRAHAMS	3	SAMPLE	12009.75	236.50	\$12,246.25		
25										25381.00

SQL Prompted Diagram Layout Results Preview

Figure 1-6 Query results after grouping, conditions, and additional formatting have been applied

Moving beyond the grid, QMF users can now instantly generate one or more charts from a query result set by selecting the respective columns and clicking the Chart icon in the toolbar. Charts can display either individual rows or aggregations, such as a sum or average, of row values against a given dimension. If aggregations are charted, users can drill down into the data by merely clicking the chart values.

In Figure 1-7, we have created three companion charts to present differing aspects of the query data, each of which is accessible by the results navigator in the sidebar. Salary by Department is the chart currently selected in the navigator in the example.

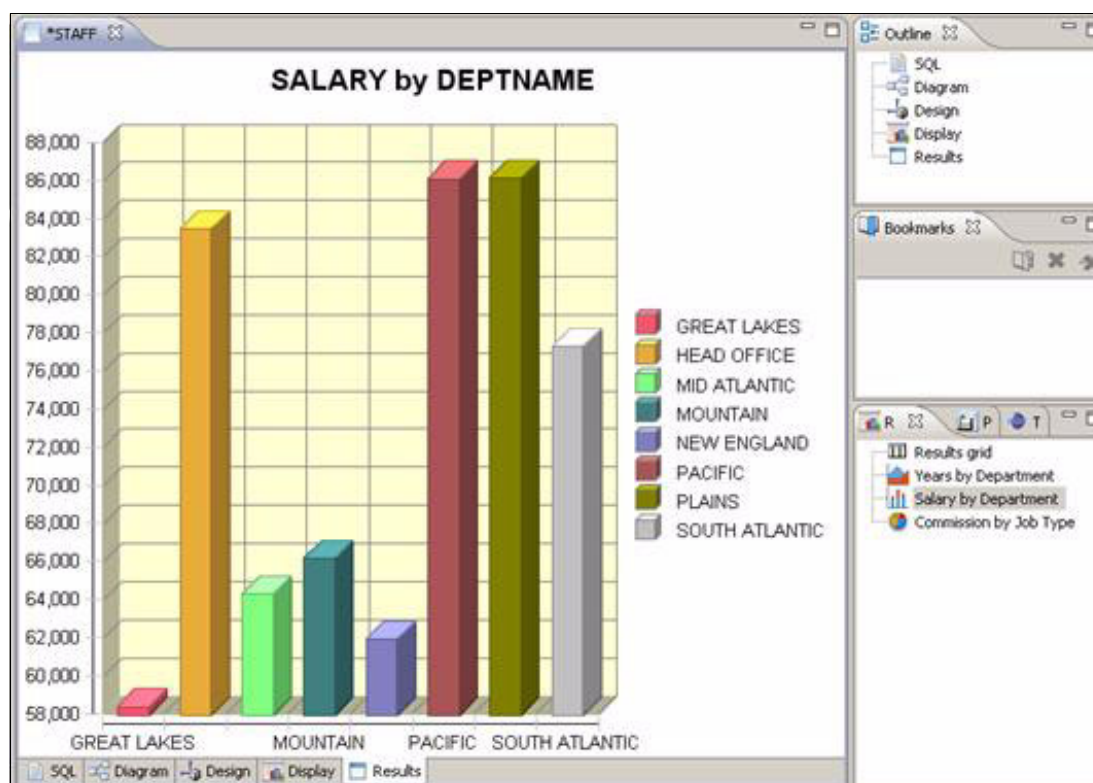


Figure 1-7 Business charting within QMF for Workstation and WebSphere

The following list summarizes just some of the features that have been added to QMF 10:

- ▶ Support for IBM DB2 Analytics Accelerator, which provides rapid analysis of complex queries
- ▶ Capabilities that allow any client that can start a DB2 for z/OS stored procedure to start QMF for TSO, run a predefined query or procedure, and return up to 20 result sets to the calling program
- ▶ Access to data that resides in any JDBC-compliant data source (DB2 family, Oracle, Informix®, SQL Server, IMS™, and more)
- ▶ OLAP access through XMLA
- ▶ 150+ new analytical functions
- ▶ New metadata capabilities, including entity-relationship diagrams (ERDs)
- ▶ Virtual data sources that reduce the complexity of the underlying data schemas, allowing end users to navigate the data more easily
- ▶ Graphical reports
- ▶ Graphical query interfaces
- ▶ Graphical dashboards and KPIs
- ▶ Ability to integrate QMF functions with third-party applications
- ▶ A new QMF Classic perspective, available within QMF for Workstation, which provides the same look and feel as that found in QMF for TSO and CICS

1.4 Evaluating total cost of ownership

Those who implement business analytics without carefully studying not only the costs associated with the acquisition of the tools, but also the hidden costs of deploying, maintaining, using, and supporting the solution, might be unpleasantly surprised as they measure their success as the project moves forward. All of the following areas need to be taken into consideration when evaluating the total cost of ownership (TCO) of a given solution, because some solutions on the market today can have significant hidden costs in these areas:

- Charges by user or by role (or both):

QMF is licensed as an enterprise solution, with charges to the customer tied only to the size of DB2 utilized on z/OS. It means that, after QMF is acquired, it can be deployed anywhere, to any number of users (regardless of user role) with no further incremental charge. The wider the deployment of QMF, the greater the value realized across the enterprise.

- Costs associated with a decentralized architecture:

Earlier in Figure 1-2 on page 4, we looked briefly at the costs associated with a decentralized business analytics architecture, where ETL and administration costs are incurred many times over as the data moves from one system to another. When evaluating TCO, it is important to remember that, in a decentralized architecture such as this one, there are many more potential points of failure as well as disaster recovery and security requirements for each individual system in the architecture. A more centralized approach with QMF on System z, depicted earlier in Figure 1-1 on page 3, has fewer overhead costs as a result.

- Resource usage:

QMF fully supports the IBM DB2 Analytics Accelerator, which provides extremely fast processing for queries flagged as costly by the DB2 optimizer.

Multiple other options also exist for controlling resource usage in other ways, such as those found in the QMF HPO component.

- Ease of use:

Unforeseen costs can be incurred when a solution is difficult to install and configure, and even greater costs are seen in lost productivity if users cannot get up and running with the product quickly or the user base cannot be centrally managed and controlled. Still further costs are incurred when users need help performing ongoing tasks, from building a query to developing dashboards and deploying them to groups of end users across the enterprise.

All of these areas have the potential for hidden costs that can be difficult, if not impossible, to accurately track. QMF 10 has made improvements in all of these areas, as we demonstrate in later chapters when we highlight key functions in more detail.

- Development costs:

If you currently use QMF Classic Edition only (QMF for TSO or CICS), you likely have already invested in developing QMF objects such as queries, procedures, forms, and applications. You might have a large number of these objects that are quite complex and have been tuned to perform well, delivering value throughout your enterprise for years. Rather than facing the cost of redeveloping, revalidating, and retesting these objects with other analytics solutions, with QMF Enterprise Edition you can use your current inventory of QMF Classic objects in QMF for Workstation and QMF for WebSphere because these components share an object catalog with QMF Classic Edition. Work that is created in QMF on System z can be shared with users on the workstation or within a browser and vice-versa.

Even if you are not an existing QMF Classic user, we show you in Chapter 4, “Maximizing your existing System z investment” on page 39 how you can access and run existing System z applications in TSO directly from within QMF for Workstation, helping you capitalize on your investment.

- Capability for global deployment:

Global organizations require tools that offer multilingual support for all staff to be as productive as possible. QMF 10 is available in 19 languages other than English, with product help and documentation translated into a subset of these languages, depending on components.

- Annual support and services:

With QMF, there is a single charge for annual support and services, which does not increase with higher usage.

1.5 The Spiffy Insurance Corporation: A fictitious company with real business needs

To highlight the new QMF as effectively as possible throughout this book, we have developed a scenario around the new features and functions. This scenario is based on a fictitious corporation, Spiffy Insurance. The Spiffy Insurance Corporation is a US-based insurer for personal and corporate accounts. The company is looking to expand its offerings within existing business areas as well as beyond its geographical borders. Like every corporation today, there are pressures to lower costs, provide better service, increase productivity, and access key information in a timelier manner.

Spiffy's mission-critical applications primarily reside on System z and have been in place since the 70s and 80s. Like many other large corporations, over time and after numerous acquisitions, the company began to implement a number of point solutions and, with those solutions, myriad analytics tools that users pressured IT into continuing to support. Data is also provided from outside sources that Spiffy must absorb into its existing infrastructure.

Spiffy's existing System z infrastructure provides the majority of data to these disparate systems, as shown in Figure 1-8.

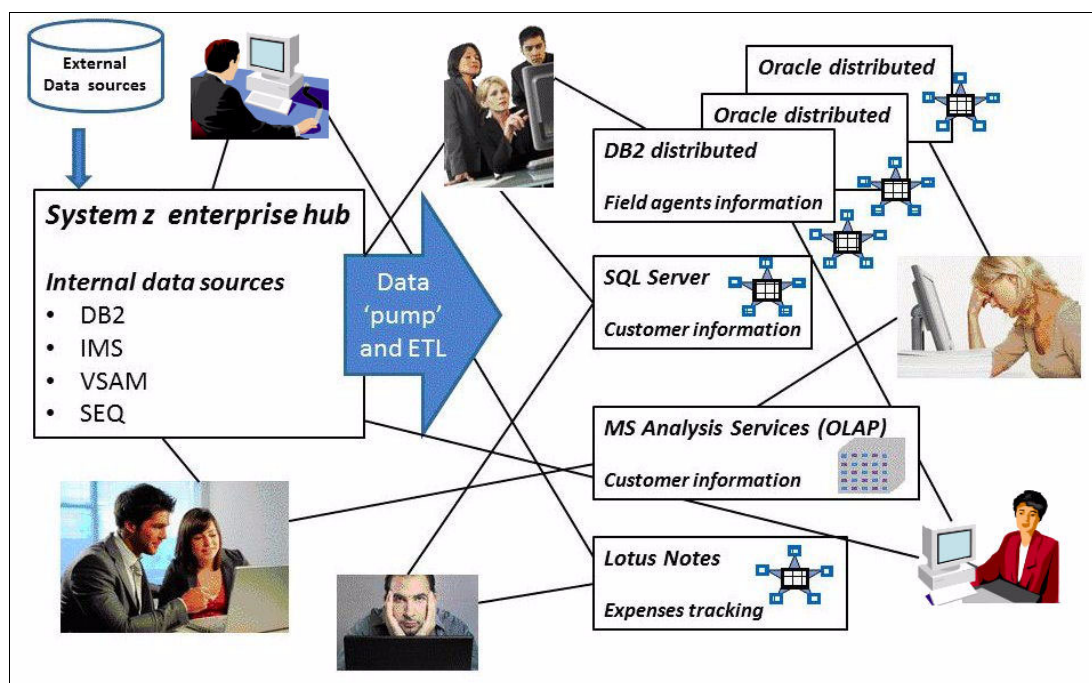


Figure 1-8 Spiffy Insurance Corporation's previous analytics architecture

As with any insurance corporation, at Spiffy there are many types of users who all need different things from the system:

- ▶ Spiffy's executives want interactive dashboards with historical as well as current (real-time) information.
- ▶ Field agents and other personnel want much of the same, but at a more granular level.
- ▶ Claims adjusters need access to many different types of information about each claim, from vehicle identification numbers to the claimant's medical history, in order to bring each claim to valid closure and protect against fraud. The complete picture for each claimant is developed from a wide variety of sources, many of them outside Spiffy's systems.
- ▶ Actuaries in all sectors of the corporation need to spend more time analyzing data (and less time reconciling and explaining inconsistencies in the data) in order to recommend changes in business proposals that result in more beneficial outcomes for Spiffy.
- ▶ Customer support representatives want data from IMS delivered within an easy-to-use dashboard; they have no desire to learn even the basic functions of QMF or any other analytics tools.
- ▶ Sales personnel and their managers have a mandate to tighten their expense budgets and to track expenses more closely. This information is held in IBM Lotus Notes, which is not considered a business analytics database, yet it is extremely expensive to rewrite the application when IT is already burdened with so many other projects.

Satisfying the needs of all users is complicated by the fact that Spiffy has allowed the creation of disparate data marts on multiple platforms. As business requirements changed, Spiffy found that these silos of information were inhibiting growth and planning. End users were constantly complaining that the information they received was either out of date or at a level that was of little use to them.

Spiffy has a strong mainframe environment despite the emergence of numerous distributed solutions. Senior management within IT and the business community mandated a thorough cost analysis, which revealed System z to be far more economical and better suited to users' needs than the company's burgeoning distributed environments. Though Spiffy had been copying System z data to distributed targets for quite some time, the analysis proved that the cost of these massive copies, as well as the delay in information delivery, was far greater than they initially believed.

As a result of the cost study, Spiffy decided to centralize its data warehouse, numerous data marts, and business analytics on System z. However, one dilemma that the company still faced was that its existing infrastructure contained distributed data on a variety of platforms that had to remain in operation despite the move. Spiffy is gradually eliminating many of these sources but, for now, any business analytics tool selected must be able to support all data on existing platforms as well as that hosted on System z.

The changes Spiffy made encompass several key business areas. The company needed to extend new services to its customers as well as to internal users. New solutions had to access information from a variety of sources and deliver it to a set of dashboards customized for specific business users and segments. Here are some of the data sources in use at Spiffy with QMF:

- ▶ DB2 for z/OS
- ▶ DB2 for LUW (Linux, UNIX, Windows)
- ▶ IMS
- ▶ Oracle on distributed platforms
- ▶ SQL Server
- ▶ MS Analysis Services
- ▶ Lotus Notes application data

Among Spiffy's objectives for the move were the following goals:

- ▶ Develop a unified data strategy to deliver data to end user dashboards from any source.
- ▶ Host an enterprise data warehouse on DB2 for z/OS.
- ▶ Reduce the number of platforms and data sources.
- ▶ Provide direct access to operational data, such as that stored in IMS.

Spiffy's business requirements parallel those of many enterprise customers today. We have broken down the overall solution elements in the remaining chapters of this book, so that you can explore them in detail for yourself.



Business analytics from the ground up: Hardware, data modeling, and data warehousing

Business analytics tools, such as QMF, provide the front end for users to easily track, analyze, and act upon data to support business processes and goals. However, these components are only part of the picture in a successful business analytics system.

Drawing actionable conclusions vitally depends on getting the right data to the right people at the right time. Therefore, it is important to take a comprehensive approach when evaluating business analytics solutions, exploring not only the front-end tools, but also the underlying systems that support them. When both operational and analytical data come from myriad sources, a single version of the truth can be difficult to come by quickly and effectively without a well architected foundation in place. Speed and optimization become differentiators for the hardware in such a system, while a warehouse design that offers minimum latency and maximum throughput becomes key to ensuring the availability of real-time information to business processes that require it.

This chapter looks at technological advancements in IBM hardware and warehousing that serve as the driving forces behind smarter analytics systems. We look at the following topics:

- ▶ The need for speed: IBM System z processors and IBM DB2 Analytics Accelerator
- ▶ Good decisions start with good data: Warehousing with DB2 for z/OS

2.1 The need for speed: IBM System z processors and IBM DB2 Analytics Accelerator

As explained in 1.5, “The Spiffy Insurance Corporation: A fictitious company with real business needs” on page 12, Spiffy Insurance has set a course for centralization of the company’s data, warehousing, and analytics on System z. At the same time, however, the company must be able to continue to access and utilize data that is scattered throughout the enterprise. Spiffy’s ideal solution to this problem involves infrastructure changes in both System z hardware and software, as shown in Figure 2-1.

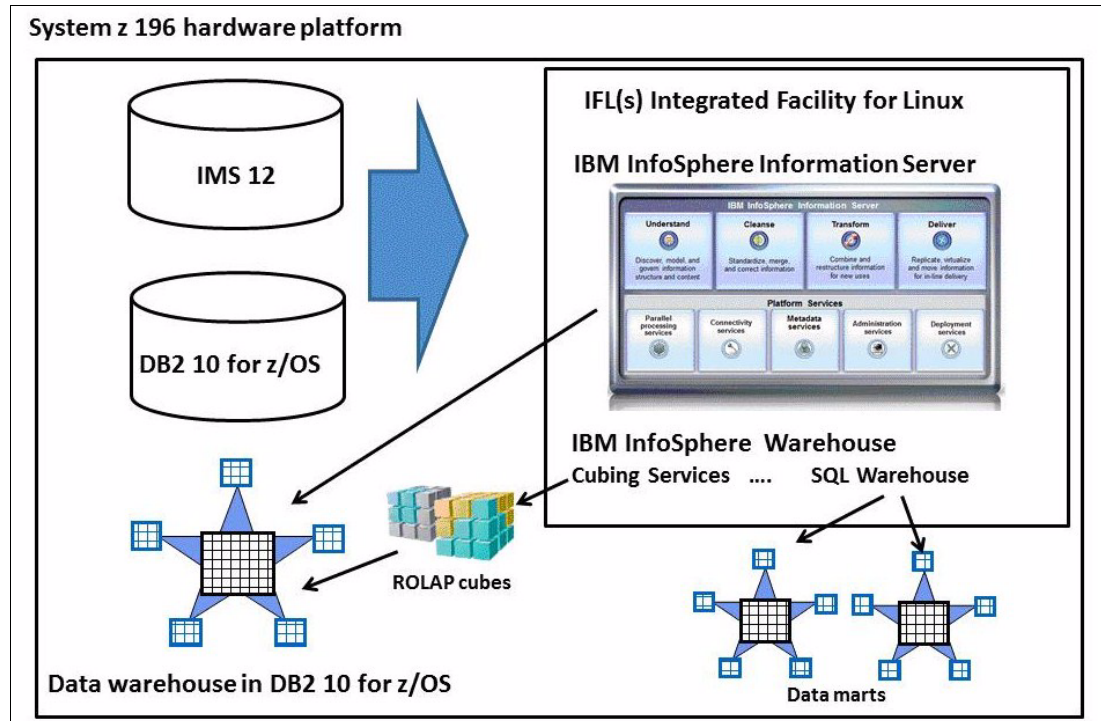


Figure 2-1 Spiffy's ideal enterprise analytics and data warehouse solution on System z

This topic looks at some of the components of the System z infrastructure that support Spiffy's business analytics transformation:

- ▶ New System z processors
- ▶ The IBM zEnterprise™ Blade Center Extension (zBX)
- ▶ The IBM DB2 Analytics Accelerator
- ▶ The Integrated Facility for Linux

2.1.1 The z196 processor: Optimal design for enterprise analytics

In the zEnterprise 196 (z196), IBM has delivered the premier system for enterprise-level processing with a number of unique features and capabilities. The z196 offers the following capabilities:

- ▶ The industry's fastest and most scalable and flexible enterprise server
- ▶ Improved performance for traditional and modern workloads
- ▶ Virtualization to enable virtually limitless capacity for massive consolidation and infrastructure simplification

- Enhanced security with support for Elliptic Curve Cryptography (ECC) technology
- 60 percent more capacity with the same energy footprint as an IBM z10™ EC, plus options for water cooling and high-voltage DC power for greater energy efficiency

To support a workload-optimized system, the z196 can scale up (over 52,000 MIPS in a single footprint), scale out (80 configurable cores) and scale within (specialty engines, cryptographic processors, hypervisors), all while executing in an environmentally friendly footprint. The z196 is designed to work together with system software, middleware, and storage to be the most robust and cost effective data server available. The z196 offers an industry-standard PCIe I/O drawer for IBM FICON® and OSA-Express multimode and single-mode fiber optic environments for increased capacity, infrastructure bandwidth, and reliability.

The z196 offers a total of 96 cores running at an astonishing 5.2 GHz, delivering up to 40 percent improvement in performance per core and up to 60 percent increase in total capacity for z/OS, IBM z/VM®, and Linux on System z workloads when compared to its predecessor, the z10 EC. The z196's 80 configurable cores can be configured as general purpose processors (CPs), Integrated Facilities for Linux (IFLs), System z Application Assist Processors (zAAPs), System z Integrated Information Processors (zIIPs), additional System Assist Processors (SAPs), Internal Coupling Facilities (ICFs), or used as additional spares.

This design, with increased capacity and number of available processor cores per server, as well as reduced energy usage and floor space, makes the z196 a perfect fit for large scale consolidation. The virtualization capabilities can support up to 47 distributed servers on a single core, and up to thousands on a single system.

System z security is one of the many reasons that the world's top banks and retailers rely on IBM mainframes to help secure sensitive business transactions. Support for the next generation of public key technologies is available with ECC. ECC is ideal for constrained environments such as mobile devices. The z196 also offers support for key ANSI and ISO standards for the banking and finance industry. The z196 has received the Common Criteria Evaluation Assurance Level 5 (EAL5) certification for security of logical partitions. It certifies System z as having the ability to secure data at the highest level in the industry. As work and applications can span numerous logical partitions, System z ensures a consistent, high level of security throughout the system.

As environmental concerns raise the focus on energy consumption, the z196 offers new efficiencies that enable a dramatic reduction of energy usage and floor space when consolidating workloads from distributed servers. For organizations looking to build green data centers, optional water cooling and high-voltage DC power allow a bold step into the future of cooler computing without changing the footprint.

2.1.2 Extensibility with the zEnterprise BladeCenter

One of the most compelling factors about the z196 and beyond is the extensibility of the architecture. The IBM commitment to enhancing not only the core System z complex, but its direction in supporting new options, such as Microsoft Windows on the new zEnterprise Blade Center Extension (zBX), is one of the key benefits of the platform.

The zBX is configured with either the z196 or z114 Central Processing Complex (CPC) through a secure high-performance private network. The zBX houses specialty processors for specific workloads, such as the IBM Smart Analytics Optimizer for DB2 for z/OS, technology which is leveraged by the IBM DB2 Analytics Accelerator, as explained next.

2.1.3 Maximum query performance with the IBM DB2 Analytics Accelerator

The ability to pose complex questions without extensive wait times is vital to business transformation within an enterprise. To address this problem, IBM offers the IBM DB2 Analytics Accelerator, a high-performance analytics accelerator appliance for System z. The solution is uniquely designed with new breakthrough technologies to reroute to a workload-optimized platform queries typically found in business analytics and data warehousing applications. The DB2 Analytics Accelerator plugs into your DB2 for z/OS environment, complementing its traditional query processing. Interfacing directly with DB2, it can speed up a substantial percentage of queries and make dramatic improvements in cost and performance, all transparently to the end user or application. The IBM DB2 Analytics Accelerator runs on z196 or z114 or higher processors.

Though the IBM DB2 Analytics Accelerator is based on IBM Netezza® technology, it is controlled and managed by DB2 for z/OS. The product ships with an administration tool that provides a means to target specific DB2 tables to load into the appliance. The DB2 optimizer then examines incoming queries. Those that are deemed costly are rerouted, if appropriate, to the IBM DB2 Analytics Accelerator, which provides extremely fast processing without the need to build indexes or further tune the database. Improved query speeds orders of magnitude faster are not uncommon.

Figure 2-2 shows the architecture of a system employing the IBM DB2 Analytics Accelerator.

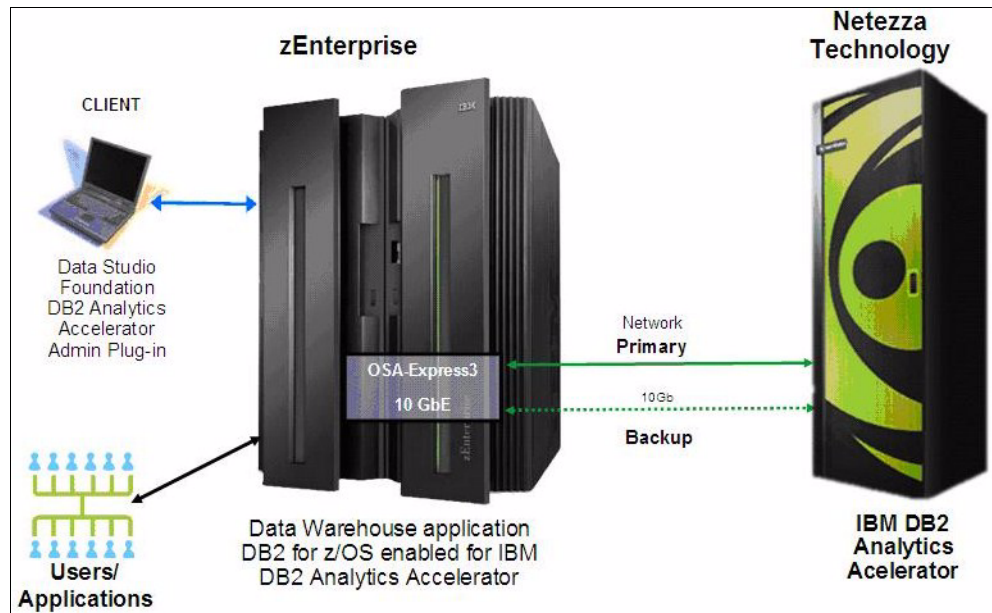


Figure 2-2 IBM DB2 Analytics Accelerator configuration

Customer acceptance and usage of the IBM DB2 Analytics Accelerator spans far more than its improvement in business analytics queries. Some, for example, have used the IBM DB2 Analytics Accelerator to accelerate their batch cycles, which have historically proven a challenge due to the sheer volume of processing required.

The IBM DB2 Analytics Accelerator is fast and easy to deploy. Two new DB2 subsystem parameters, **ACCEL_LEVEL** and **QUERY_ACCELERATION**, are the means by which support for IBM DB2 Analytics Accelerator is enabled. The **QUERY_ACCELERATION** parameter specifies the initial setting for a new special register called **CURRENT QUERY ACCELERATION**. This register controls whether queries will be accelerated if the DB2

optimizer determines that it is advantageous to do so. QMF Version 10 supports the SET CURRENT QUERY ACCELERATION statement in SQL queries, allowing you to turn evaluation for query acceleration on or off as desired.

For more information about how to accelerate queries with IBM DB2 Analytics Accelerator, see *Optimizing DB2 Queries with IBM DB2 Analytics Accelerator for z/OS*, SG24-8005.

2.1.4 The System z Integrated Facility for Linux for support of Linux on System z

As shown in Figure 2-1 on page 16, Linux on System z is a strategic direction for Spiffy Insurance because IBM InfoSphere® Information Server and InfoSphere Warehouse, both of which require Linux, are used to host the company's data warehouse. To support this effort, Spiffy has invested in the IBM Integrated Facility for Linux (IFL). The IFL is a processor dedicated to Linux workloads on IBM System z servers. It runs on z/VM or Linux operating systems.

Spiffy finds that the z196 and its IFLs offer IT optimization and cloud computing on zEnterprise with better economies of scale. The faster speed of the IFL on the new zEnterprise 196, in conjunction with the larger cache memory structure, enables support for more virtual servers per processor core than other server platforms. Linux on System z further offers these benefits:

- ▶ Lower acquisition costs of hardware and software versus distributed servers
- ▶ Reduction in floor space by up to 90% when compared to distributed servers
- ▶ Reduction in labor costs by up to 70% when compared to distributed servers

2.2 Good decisions start with good data: Warehousing with DB2 for z/OS

Both quality and timeliness of information are centrally important to making sound business decisions. Not too many years ago, it was standard for analytics tools to attempt to perform tasks beyond their design scope, such as data extraction, transformation, and cleansing. Now there are solution elements that provide these functions, offering the means of creating, maintaining, and administering an enterprise data warehouse that is separate from the analytics tool.

2.2.1 Why a solid data warehousing solution is critical for business analytics

Without good data, it does not really matter how good your business analytics tools are. If users do not have the data they need in a timely enough manner to act on it, or if that data is not accurate or up to date, the solution will fail. But what are the essential components of a solid data warehousing solution and what function does each perform?

A complete enterprise data warehouse includes the following components:

- ▶ Data definition and metadata capture:

There must be a single version of the truth for the definitions being used within an enterprise. A typical warehouse will capture data from multiple sources where there might not be common terminology for all elements that are to be collected and amalgamated.

For example, if there are multiple sources for regional sales information, regardless of the input terminology used, the output terminology must be consistent. An effective enterprise data warehouse system will retain the old definitions and sources as well as the new, target information. Metadata must be shared with all components within the warehouse.

- ▶ Data capture, extraction, and change capture:

Data is in constant flux, with new information and updates to existing information arriving continually. The timeliness of capture and rapid processing of changes and inserts are absolutely critical to the success and usefulness of the warehouse.

- ▶ Data replication:

Data must be uploaded to the warehouse from operational systems. Any replication tool used must offer low latency with very high throughput in order to be effective. Mission-critical data events must be replicated in real time without impacting system performance. Together, IBM InfoSphere Change Data Capture and Change Data Capture for z/OS replicate heterogeneous data to and from DB2 for z/OS, offering solutions to these issues.

- ▶ Data validation and cleansing:

Because data in a warehouse comes from many different sources, the same data is often presented in inconsistent ways. The causes of these inconsistencies can be differences in metadata definition, entry errors, or corruption. These problems are prevented or removed by data validation and cleansing functions.

- ▶ Data transformation:

Data must often be transformed with new values or changed definitions based upon the process that needs to be applied to it. For example, an input record might use a terse, numeric key to indicate a particular state, while the output to a warehouse must transform this into an actual state value, such as “Michigan.” Conditional processes are also often applied during data transformation.

- ▶ Speed of data deployment:

Real-time data processing is becoming more critical to organizations everywhere, requiring increasingly shorter turnaround times for data to be captured, cleansed, transformed, and loaded into the data warehouse.

In addition to the previous attributes, the ideal warehousing solution also has these important characteristics:

- ▶ Proximity to the business analytics tools and processes that will access and use the data in the warehouse:

Close proximity of a process to its data results in higher efficiency and less overhead. As shown in Figure 2-1 on page 16, Spiffy stores the bulk of its data in databases on z/OS: IMS and DB2. For Spiffy, placing the warehouse on System z yields significant returns in the time taken to access, update, and deliver the necessary information. The impact on business analytics operations is significant, as business users and end users alike express greater confidence in the quality and timeliness of the information.

- ▶ A “shape” suited to the purpose and usage of the data:

Clean, accurate data is extremely important, but there is an additional element to consider. If you have experience with analytics tools, you have probably encountered situations where the request for a particular report or output seemed simple enough. Then you were provided access to data that was extremely challenging to work with to produce the desired results. After many hours of trial and error, you might have been able to create what was requested. However, your parting thought might have been, “If only the data had been structured this way in the first place, I could have done it in a fraction of the time!”

The structure of the information to be used (star schema, snowflake, and so on) can be just as critical to the effectiveness and efficiency of the business analytics implementation. The majority of queries by end users contain some degree of dimensionality. For example, “I want to see the sum of sales by month, by line of business, from the year 2007 forward.” In a typical data warehouse scenario, the elements of time and the line of business would be developed as separate tables (dimensions) and linked to a central fact table where the values are stored.

2.2.2 Why DB2 for z/OS is ideal for data warehousing

An earlier Redbooks publication on data warehousing *Enterprise Data Warehousing with DB2 9 for z/OS*, SG24-7637, provides a compelling study in the advantages of DB2 for z/OS as a data warehouse server. DB2 10 for z/OS takes these advantages several steps further, making it a continued superior choice for data warehousing.

The goal with this release of DB2 has been to deliver greater processing power with less resource consumption. DB2 for z/OS now provides these capabilities:

- ▶ Increased uptime, supporting mission-critical analytics
- ▶ Increased security; the best in the industry
- ▶ Greater throughput
- ▶ Reduction in processing time to complete required tasks, with most customers able to achieve out-of-the-box CPU savings of 5 -10% for traditional workloads and up to 40% for specific workloads
- ▶ Support of the IBM DB2 Analytics Accelerator, which not only dramatically changes the speed of many queries, as previously explained, but also dramatically simplifies the data warehousing process
- ▶ Capabilities to consolidate applications and data warehouses with less cost and complexity and fewer resources to manage
- ▶ The industry's first integrated bitemporal capabilities built directly into the database (as we examine in more detail in Chapter 3, “DB2 for z/OS as an analytics engine” on page 27)
- ▶ Virtual storage improvements that deliver up to 10 times more scalability, providing improved performance, reduced complexity, and cost savings
- ▶ Extensions to built-in security and trace features to provide end-to-end auditing capabilities, which simplifies compliance requirements

2.2.3 IBM industry data models

IBM has created a number of customized data models that combine deep expertise with industry best practices to form a blueprint for customers looking for easy to use, industry-specific warehousing solutions. Application of these models accelerates business analytics processes in the industries for which they have been built, for example:

- ▶ Banking
- ▶ Finance
- ▶ Insurance
- ▶ Health Care
- ▶ Retail
- ▶ Telecommunications

Insurance industry data models

Part of the IBM InfoSphere portfolio, the industry models are based on the experience of more than 500 clients, and more than ten years of development. For more information about the IBM industry models, see the IBM Industry Models Library at this website:

<http://www.ibm.com>

The IBM Insurance Information Warehouse

The IBM Insurance Information Warehouse (IIW) takes insurance industry data models one step further, offering comprehensive data warehouse design models, business terminology models, and analysis templates that accelerate the development of business applications within the insurance industry. The net result is rapid creation and delivery of timely information, resulting in smarter decisions that lead to better overall business performance.

The models provided by the IIW are extremely thorough and contain a multitude of elements designed to address the majority of insurance analytics requirements. For Spiffy, the IIW provided a fast-path means of adjusting the previous data model in use at the company to one that has been thoroughly tested and designed specifically for their industry. The IIW provides the following capabilities:

- ▶ Enables the consolidation of clean, meaningful financial data across multiple channels and products
- ▶ Covers a wide scope of insurance industry areas:
 - Enterprise risk management
 - Finance and compliance reporting (including support for Solvency II: Quantitative Impact Study 5, or QIS5, and Consultation Paper 58, or CP58)
 - Sarbanes-Oxley Act
 - Claims
 - Intermediary performance
 - Basic life and pension actuarial
 - Corporate pensions compliance
- ▶ Enables business users to more effectively control and reduce the time taken to scope their requirements, subsequent customization, and any extension of the data warehouse
- ▶ Provides business solution templates that are based on the Kimball dimensional model approach and enable integration with IBM Netezza as well as online analytical processing (OLAP) functions, such as those found in QMF
- ▶ Provides a solid basis for statutory reporting and relationship management which, in turn, provide input to decision support and performance management applications
- ▶ Helps minimize development costs
- ▶ Reduces the risk of failure by facilitating an incremental approach to delivering an integrated reporting repository

As you can see in later chapters of this book, QMF provides a metadata layer that easily absorbs externally created data definitions (for example, those created in the IIW). It allows you to accept definitions from a particular database and either use them as-is to be immediately productive or modify them for a specific user or set of users. Spiffy found little need to modify the IIW definitions, but did employ the QMF metadata functions to define new data as well as create virtual views containing data from multiple sources.

After installing the IIW, Spiffy used the IIW metadata to feed the IBM InfoSphere Information Server to create a set of self-contained warehouse build and maintain processes.

2.2.4 IBM InfoSphere Information Server

The IBM InfoSphere Information Server is a suite of IBM products and functions that provides a “cradle to grave” approach for data, from definition to delivery. Earlier we noted the basic elements required for a solid enterprise data warehouse foundation. As shown in Figure 2-3, the Information Server contains several functions that enable the rapid definition, creation, and deployment of a data warehouse.



Figure 2-3 Main functions of IBM's InfoSphere Information Server

The InfoSphere Information Server's extract, transform, and load (ETL) capabilities provide a rich environment to maintain and grow a data warehouse (or a series of data marts). It runs under Linux on System z. The InfoSphere Information Server is used at Spiffy to manage the Insurance Information Warehouse data models. Data feeding the Information Server can come from a variety of sources, including IMS, DB2, and VSAM on System z. After the connections to the various data sources are established and the data is accessible, the Information Server maintains the entire cycle of data warehousing processes required to keep the information current, accurate, and in sync.

The InfoSphere Information Server for System z includes the following components:

- ▶ *InfoSphere IBM DataStage® for Linux on System z*: Integrates data on demand with a high-performance parallel framework, extended metadata management, and enterprise connectivity. It supports the collection, integration, and transformation of large volumes of data, with data structures ranging from simple to highly complex.
- ▶ *InfoSphere Information Analyzer*: An exploratory option to derive analytics data and definitions from enterprise data sources. If you want to view source information and use it to create new definitions for analytics, the analyzer provides a wide array of capabilities.
- ▶ *InfoSphere Quality Stage for Linux on System z*: A single set of standardization, cleansing, matching, and survivorship rules for your core business entities, executed in batch, real time, or as a web service. It processes global data on a massively scalable parallel platform for optimal performance.
- ▶ *InfoSphere Metadata Server for Linux on System z*: Offers data profiling and monitoring capabilities that help derive greater value from complex information spread across enterprise systems.
- ▶ *IBM Metadata Workbench for Linux on System z*: Deep metadata definition and exploration to assist in deriving a “single version of the truth” for enterprise data.

- ▶ *InfoSphere Business Glossary for Linux on System z*: Defines and delivers business terms and definitions instead of complex, technical terminology. Users can view in-depth corporate definitions for information presented, such as those where multiple definitions in the past have been confusing and counter-productive.
- ▶ *InfoSphere Information Services Director for Linux on System z*: Allows you to define and publish enterprise data related tasks as part of a service within a corporate SOA.

More detailed information about the IBM InfoSphere Information Server is available at the following website:

http://www.ibm.com/software/data/integration/info_server_system_z/overview/

2.2.5 The IBM InfoSphere Warehouse

The InfoSphere Warehouse provides a highly scalable, highly resilient, lower cost way to optimize a DB2 for z/OS data warehouse. Spiffy decided that the IBM InfoSphere Warehouse solution offered several unique capabilities that would accelerate the company's overall data warehouse restructuring processes as they progressed through their business analytics implementation. Spiffy is working with several user segments that have relied upon OLAP cubes and data marts for departmental copies of data. The IBM InfoSphere Warehouse on System z provides some powerful and easy to configure and use extract, transform, and load (ETL) functions, as well as a feature that creates ROLAP (relational OLAP) cubes without the expense of moving data out of the DB2 for z/OS environment.

The SQL Warehousing Tool

The SQL Warehousing Tool provides data flow, data transformation, and control flow operations required to load data in the proper format into an analytical data store. It provides these services by making use of the power of the DB2 relational database engine and the SQL language. It also provides sequencing and flow control functions, as well as functions to integrate non-database processing. The SQL Warehousing Tool includes these features:

- ▶ Ability to restart control flow from the failure point
- ▶ Ability to load data sets from the server
- ▶ Staging table tuning (creating indexes and range partitions, running RUNSTATS)
- ▶ Order by 'GET FIRST N ROWS'
- ▶ Visual enhancements for flow debugging and execution
- ▶ Virtual columns, expression sorting, filtering enhancements
- ▶ Parallel execution in control flows
- ▶ Sub-process modularizations
- ▶ Variable enhancements, reserved system variables
- ▶ Activity skip during execution
- ▶ Enhanced diagnostics

Access to multidimensional data with no-copy OLAP analytics

InfoSphere Warehouse on System z customers can also build and deploy multidimensional analytics with Cubing Services. Cubing Services empower users to ask intuitive and complex ad-hoc questions about their enterprise, such as “What is my product's profitability for this quarter across my territory?” This type of business query requires the grouping, aggregation, and calculation of information across various business dimensions, such as time, product, customer, and supplier. Cubing Services is designed to resolve just this type of multidimensional query, known as an OLAP query. Because it builds cubes “on the fly” from a relational source, it is also commonly called ROLAP.

Cubing Services provides Spiffy with new insights over traditional outboard cubing technologies. Cubing Services scales to over a terabyte, so Spiffy users are no longer forced to aggregate their data, losing visibility to detail-level activity. It means that any root cause analysis being performed will have drill-down capabilities to the individual transaction or product. Leveraging data directly from the warehouse, Cubing Services does not require the overhead of copying data into a third-party technology. It is fully integrated into the InfoSphere Warehouse on System z design and administrative warehouse tools.

QMF uses Cubing Services as an OLAP source, allowing you to join multiple data sources into a common logical view. Not only can users drill down through increasingly greater levels of detail from their OLAP view, they can also drill through to data held elsewhere if there is a common value to join these views.

For more detailed installation and setup instructions on the InfoSphere Information Warehouse, see the following information center:

<http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp>

In our next chapter, we delve a bit deeper into the features and functions provided within DB2 for z/OS that make it ideally suited to drive enterprise analytics. QMF has co-evolved with DB2 for z/OS and continuously enhances its support as new features, such as bitemporal data, are made available.



DB2 for z/OS as an analytics engine

DB2 10 for z/OS is the leading enterprise data server, designed and tightly integrated with IBM System z. It leverages the strengths of the System z platform, reducing total cost of ownership through process enhancements and productivity improvements for database administrators and application developers. Its continuous availability provides the confidence and reliability required for business analytics as they have evolved from a set of useful tools and functions to taking on a mission-critical role within many enterprises.

This chapter looks at the crucial role that DB2 for z/OS plays in business analytics:

- ▶ It covers the advantages for analytics provided in DB2 10.
- ▶ It provides a brief tour of database management with IBM DB2 Tools.

3.1 Advantages for analytics in DB2 10

DB2 for z/OS has established a new standard for database management systems with its many innovations and enhancements available in Version 10. Business analytics processes are normally more complex than the typical SQL statement. As an analytics engine, DB2 10 for z/OS offers many advantages that provide a solid platform for queries submitted by analytics tools. The following sections describe each of these advantages in more detail.

3.1.1 CPU savings for lower total cost of ownership

Reducing the cost of doing business is always an important issue and challenge for IT. DB2 10 helps to lower total cost of ownership by reducing CPU usage while delivering value. IBM DB2 10 testing and early beta customer results revealed that, depending on the specific workload, customers could achieve “out-of-the-box” DB2 CPU savings of 5% - 10% for traditional workloads, and up to 20% for specific workloads, when compared to running the same workloads on DB2 9. The earlier in the migration a REBIND is performed, the sooner the best performance and memory improvements can be realized.

DB2 reduces its CPU usage by optimizing processor times and memory access, leveraging the latest processor improvements, larger amounts of memory, and z/OS enhancements. Improved scalability and virtual storage constraint relief can add to the savings. Continued productivity improvements for database and system administrators can drive even more savings.

3.1.2 Advancements in scalability

Because memory constraints have been addressed as explained before, virtual memory is no longer critical, allowing 5 to 10 times the number of concurrent threads in a single DB2 10 member. This increase in threads removes a key reason for having additional DB2 data sharing members and therefore allows some consolidation of LPARs and members previously built for handling more users.

3.1.3 Reduced catalog lock contention

The DB2 10 catalog has been restructured to reduce lock contention by removing all links in the catalog and directory. In addition, new functionality improves DB2's lock avoidance techniques, and improves concurrency as well by holding acquired locks for less time and preventing writers from blocking the readers of data. The DB2 10 catalog uses partition-by-growth universal table spaces, with one catalog table per table space and all catalog objects under DFSMS control.

In DB2 10 new-function mode (NFM), you can access currently committed data to minimize transaction suspension. Now, a read transaction can access the currently committed and consistent image of rows that are incompatibly locked by write transactions without being blocked. Using this type of concurrency control can greatly reduce timeout situations between readers and writers who are accessing the same data row.

3.1.4 64-bit virtual storage relief

For many years, virtual storage has been the most common constraint for large customers. Prior to DB2 10, the amount of available virtual storage below the bar potentially limited the number of concurrent threads for a single data sharing member or DB2 subsystem.

DB2 8 provided the foundation for virtual storage constraint relief (VSCR) below the bar and moved a large number of DB2 control blocks and work areas (buffer pools, castout buffers, compression dictionaries, RID pool, trace tables, part of the EDM pool, and so forth) above the bar. DB2 9 provided additional relief of about 10-15%. Although this level of VSCR helped many customers to support a growing number of DB2 threads, other customers had to expand their environments horizontally to support workload growth in DB2, by activating data sharing or by adding further data sharing members, which added complexity and further administration to existing system management processes and procedures.

DB2 10 for z/OS provides a dramatic reduction of virtual private storage below the bar, moving 50-90% of the current storage above the bar by exploiting 64-bit virtual storage. This change allows as much as 10 times more concurrent active tasks in DB2. As a result, customers need to perform much less detailed virtual storage monitoring. Some customers can have fewer DB2 members and can reduce the number of logical partitions (LPARs). The net results for DB2 customers are cost reductions, simplified management, and easier growth.

Because of the database manager address space (DBM1) VSCR also delivered in DB2 10, it becomes more realistic to think about consolidating data sharing members or LPARs. DB2 9 and earlier versions of DB2 use the 31-bit extended common service area (ECSA) to share data across address spaces. If several data sharing members are consolidated to run in one member or DB2 subsystem, the total amount of ECSA that is needed can cause virtual storage constraints for the 31-bit ECSA. To provide virtual storage constraint relief (VSCR) in this situation, DB2 10 more often uses 64-bit common storage instead of 31-bit ECSA. For example, in DB2 10, the instrumentation facility component (IFC) uses 64-bit common storage. When you start a monitor trace in DB2 10, the online performance (OP) buffers are allocated in 64-bit common storage to support a maximum OP buffer size of 64 MB (increased from 16 MB in DB2 9).

Other DB2 blocks have also been moved from the ECSA to 64-bit common storage. Installations that use many stored procedures (especially nested stored procedures) can potentially see a reduction of several megabytes in the ECSA.

3.1.5 Bitemporal queries

DB2 10 provides temporal data functionality, often referred to as time-travel queries, through the `BUSINESS_TIME` and `SYSTEM_TIME` table period definitions. These period definitions are used for temporal table definitions to provide system-maintained, period-maintained, or bitemporal (both system- and period-maintained) data stores. These temporal data tables are maintained automatically and, when the designated time period criterion is met, the data is archived to an associated history table.

Bitemporal queries allow you to more easily measure change, trends, and growth or losses over time. For example, suppose that Spiffy Insurance Corporation agents want to see a history of policy coverage for a particular customer from a point in time to the current day. Prior to implementing DB2 10 for z/OS, agents had to submit a request to IT to retrieve and present these types of changes. With the temporal data features in DB2 10, they are now able to have the system automatically process these changes.

As another example, suppose that Spiffy's agents need to examine policy terms in cases where a client is challenging a claim resolution and there is a need to view the policy terms in effect at the time of the accident. In many cases, these terms might not provide the same coverage as those in currently in place. Bitemporal queries get to these types of answers quickly and easily.

The SQL query interface in QMF for TSO and CICS as well as QMF for Workstation and WebSphere Version 10 fully supports bitemporal SQL. In QMF for Workstation and WebSphere, QMF additionally checks for temporal support when displaying a table within a query and, if enabled, provides the ability for end users to define a specific date context for the query or a date range over which the data retrieval should be applied. Figure 3-1 shows a join in progress in QMF for Workstation/WebSphere's query diagram interface. The **Time Period** icon, highlighted in Figure 3-1, is enabled when a table that has been configured for temporal data is added to the query (in this case, Spiffy's Loss Profiles table).

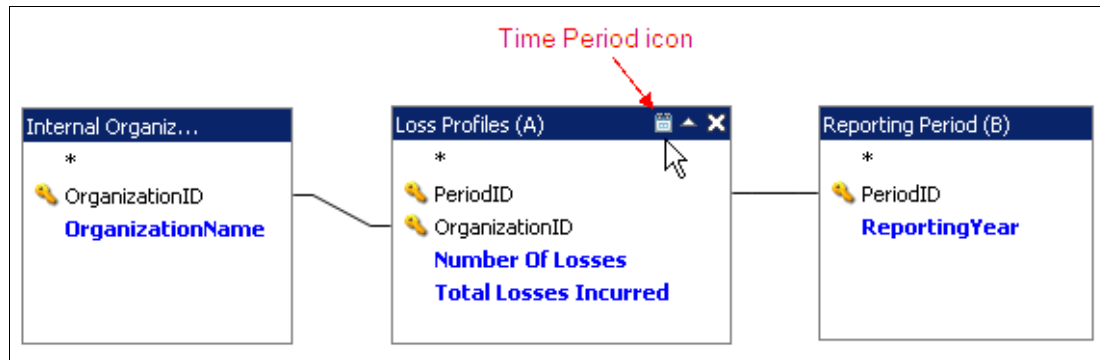


Figure 3-1 Accessing temporal features in QMF 10's query diagram editor

When a table has been configured for temporal data, users can optionally define the date range or point in time over which the data should be retrieved. Clicking the **Time Period** icon displays the dialog shown in Figure 3-2.

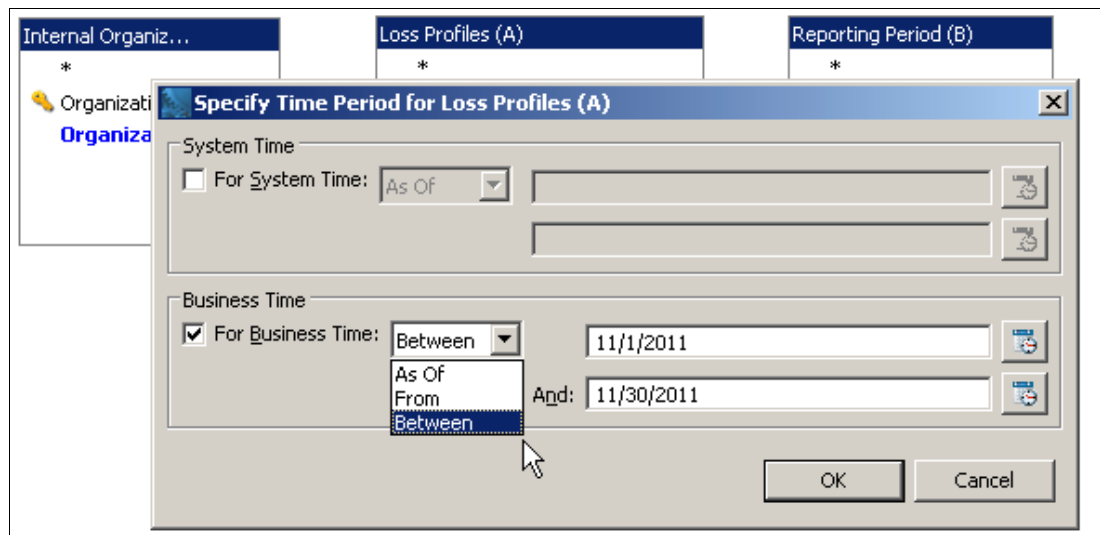


Figure 3-2 Defining temporal constraints on a table within a QMF query

Using this function, Spiffy's analysts can directly answer a wide variety of business questions, such as showing the number of losses or total dollar amount of loss for different periods in time. These selections can also be parameterized, allowing users (or reports or dashboards that embed the temporal query) to supply the dates at runtime.

3.1.6 Integrated XML support

DB2 10 substantially improves DB2 family consistency and productivity for IBM pureXML® users. These enhancements deliver excellent performance improvements:

- ▶ Support for the binary XML format, an external representation of an XML value that can be used for the exchange of XML data between a client and a data server
- ▶ XML schema validation as a built-in function
- ▶ XML date and time data types and functions, including a time zone feature and arithmetic and comparison operators
- ▶ Exploitation of XML indexes for XML joins
- ▶ Simplification of tasks that convert XML values to a SQL data type through the use of the XMLCAST specification
- ▶ The ability of an XPATH expression to return values as a table using the XMLTABLE function
- ▶ XML consistency checking using the CHECK DATA utility
- ▶ Support for multiple versions of XML documents
- ▶ Support for partial update of an XML document

In addition, to enhance DB2 family compatibility and application portability, DB2 10 XML supports the use of the XML data type for IN, OUT, and INOUT parameters and for SQL variables inside the procedure and function logic. These DB2 10 enhancements are only for native SQL procedures and for SQL user-defined scalar and table functions.

In November 2011, support was added for XQuery in IBM® DB2® 10 for z/OS® through APARs PM47617 and PM47618. XQuery extends XPath with new language constructs, adding expressiveness to the IBM pureXML® features of DB2. Like XPath, XQuery is a W3C standard, and its addition is a further step in ensuring compatibility within the DB2 family of products.

3.1.7 Support for OLAP: Moving sums, averages, and aggregates

The OLAP capabilities of moving sums, averages, and aggregates are now built directly into DB2 for z/OS. Improvements within SQL, intermediate work file results, and scalar or table functions provide performance for these OLAP activities. Moving sums, averages, and aggregates are common OLAP functions and involve typical standard calculations that are accomplished using different groups of time-period or location-based data for common criteria, such as product sales or store location.

These OLAP capabilities are further enhanced through scalar, custom table functions or the new temporal tables to establish the window of data for the moving sum, average, or aggregate to calculate its answer set. By using a partition, time frame, or common table SQL expression, the standard OLAP functions can provide the standard calculations for complex or simple analytics requirements. Also, given the improvements within SQL, these moving sums, averages, and aggregates can be included in expressions, select lists, or ORDER BY statements, satisfying any application requirements.

3.1.8 System Management Facility compression

To improve query performance, metrics on how the SQL is behaving are required. These metrics are stored in System Management Facility (SMF) 100, 101, and 102 records, with the amount of SMF data gathered for DB2 being significant at times. Because SMF record volume for DB2 can be very large, significant savings can be realized by compressing DB2 SMF records. Compression can provide increased throughput (as the records written are smaller) as well as savings of auxiliary storage space to house these files. Laboratory measurements show that SMF compression generally saves 60% to 80% of the space for DB2 SMF records and requires less than 1% in CPU for overhead to do so.

This option is enabled in DB2 by a new DSNZPARM keyword.

3.1.9 Dynamic compression with INSERT

Prior to DB2 10, the only way to build the compression dictionary after an ALTER TABLESPACE SQL statement to enable compression was to run the LOAD or REORG utility. However, scheduling a LOAD or REORG can be challenging. In DB2 10 New Function Mode (NFM), after enabling compression with ALTER, the compression dictionary is built when you execute any of the following statements:

- ▶ INSERT
- ▶ MERGE
- ▶ LOAD SHRLEVEL CHANGE

3.1.10 Dynamic SQL EXPLAIN

In the previous version of DB2 for z/OS, collecting explain data for dynamic SQL was challenging. DB2 10 improves that situation considerably. The collection of explain information for dynamic SQL can be enabled by the newly introduced special register CURRENT EXPLAIN MODE. The register can be set to the following values:

- ▶ NO, indicating that no explain information is captured during the execution of an explainable dynamic SQL statement.
- ▶ YES, indicating that explain information is captured in the explain tables as eligible dynamic SQL is prepared and executed.
- ▶ EXPLAIN, indicating that explain information is captured in the explain tables as eligible dynamic SQL is prepared. However, dynamic statements (except SET statements) are not executed and a SQLCODE +217 is returned, indicating a dynamic SQL statement was not executed.

3.1.11 Instance-based statement hints

Instance-based statement hints (also referred to as system-level access path hints) provide a new mechanism for matching hints to a given query. In prior releases, QUERYNO linked queries to their associated hints, which could be error prone because QUERYNO might potentially change, which requires a change to applications for dynamic SQL. With DB2 10, the mechanism uses query text to match with corresponding hints (similar to how statement matching is performed for the dynamic statement caching). Using this mechanism, the hints are enforced based on the statement text for the entire DB2 subsystem (hence the name “instance-based” or “system-level”).

3.1.12 Dynamic SQL information

You can collect information about prepared dynamic SQL statements through audit trace IFCID 145. DB2 10 changes the IFCID 145 to support auditing the entire SQL statement text and to indicate row permission and column mask object usage during the access path selection process.

3.1.13 Query parallelism enhancements

DB2 10 improves several existing access paths through parallelism. These specifically designed enhancements eliminate some previous DB2 restrictions, increase the amount of work redirected to the zIIP processors, and distribute work more evenly across the parallel tasks. These enhancements provide additional reasons to enable parallelism within your environment.

Parallelism improves your application performance, and DB2 10 for z/OS can now take full advantage of parallelism with the following types of SQL queries:

- ▶ Multi-row fetch
- ▶ Full outer joins
- ▶ Common table expressions (CTE) references
- ▶ Table expression materialization
- ▶ Table functions
- ▶ CREATE GLOBAL TEMPORARY tables (CGTTs)
- ▶ Work files resulting from view materialization

The new parallelism enhancements can also be active during the following specialized SQL situations:

- ▶ When the optimizer chooses index reverse scan for a table
- ▶ When a SQL subquery is transformed into a join
- ▶ When DB2 chooses to do a multiple column hybrid join with sort composite
- ▶ When the leading table is sort output and the join between the leading table and the second table is a multiple column hybrid join

Additional DB2 10 optimization and access improvements also help many aspects of application performance. In DB2 10, index look-aside and sequential detection help improve referencing parent keys within referential integrity structures during INSERT processing. This process is more efficient for checking the referential integrity dependent data and reduces the overall CPU required for the insert activity.

3.1.14 Index enhancements

The following index improvements in DB2 10 can enhance the performance of analytics applications:

- ▶ INCLUDE COLUMN clause:

DB2 10 allows you to define additional non-key columns in a unique index. DB2 in NFM includes a new INCLUDE clause on the CREATE INDEX and ALTER INDEX statements. The INCLUDE clause is only valid for UNIQUE indexes. The extra columns specified in the INCLUDE clause do not participate in the uniqueness constraint. You do not need the extra five-column index and the processing cost of maintaining that index.

- I/O parallelism for index updates:

DB2 10 provides the ability to insert into multiple indexes that are defined on the same table in parallel. Index insert I/O parallelism manages concurrent I/O requests on different indexes into the buffer pool in parallel, with the intent of overlapping the synchronous I/O wait time for different indexes on the same table. This processing can significantly improve the performance of I/O-bound insert workloads. It can also reduce the elapsed times of LOAD RESUME YES SHRLEVEL CHANGE utility executions, because the utility functions similar to MASS INSERT when inserting to indexes.

- Range-list index scan:

The range-list index scan feature allows for more efficient access for some applications that need to scroll through data. Queries of this type are most often found in applications that contain cursor scrolling logic where the returned result set is only part of the complete result set.

These types of queries (with OR predicates) can suffer from poor performance because DB2 cannot use OR predicates as matching predicates with single-index access. The alternative method is to use multi-index access (index ORing), which is not as efficient as single-index access. Multi-index access retrieves all RIDs that qualify from each OR condition and then unions the result. DB2 10 can process these types of queries with a single-index access, improving the performance of these types of queries. This type of processing is known as a range-list index scan, also referred to as SQL pagination.

- Index probing:

In DB2 10, the optimizer can use Real Time Statistics (RTS) data and probe the index non-leaf pages to come up with better matching predicate filtering estimates. DB2 10 uses index probing in the following situations:

- When the RUNSTATS utility shows that the table is empty
- When the RUNSTATS utility shows that there are empty qualified partitions
- When the catalog statistics have default values
- When a matching predicate is estimated to qualify zero rows

This new safe query optimization technique, also known as index probing, is only used for matching index predicates with hard-coded literals or when REOPT() is used to supply the literals.

3.1.15 Buffer pool enhancements

Buffer pool prefetch, which includes dynamic prefetch, list prefetch, and sequential prefetch activities, is 100% zIIP eligible in DB2 10 for z/OS. DB2 10 zIIP-eligible buffer pool prefetch activities are asynchronously initiated by the database manager address space (DBM1) and are executed in a dependent enclave that is owned by the system services address space (MSTR). Because asynchronous services buffer pool prefetch activities are not accounted to the DB2 client, they show up in the DB2 statistics report instead. Deferred write is also eligible for zIIP.

3.1.16 Work file enhancements

DB2 10 has three significant work file enhancements:

- ▶ Increased work file record length:

DB2 10 in NFM allows work file records to be spanned, which provides the functionality to allow the work file record length to be up to 65529 bytes by allowing the record to span multiple pages. This support alleviates the issue of applications receiving SQLCODE -670 (SQLSTATE 54010) if the row length in the result of a join or the row length of a large sort record exceeds the 32 KB maximum page size of a work file table space.

- ▶ In-memory work file enhancements:

These enhancements are intended to reduce the CPU time consumed by workloads that execute queries that require the use of small work files. In-memory work file support is available in DB2 10 conversion mode.

- ▶ Partition-by-growth table spaces in the work file database:

DB2 10 in NFM allows table spaces in the work file database to be defined as partition-by-growth universal table spaces. It is the preferred method of defining work files used for declared global temporary tables (DGTT).

3.1.17 Sort enhancements

DB2 10 has five sort enhancements that can have a positive effect on query workloads. They are as follows:

- ▶ Increase in default sort pool storage size (to 10 MB)
- ▶ Hash technique for GROUP BY queries
- ▶ Hash technique for sparse indexes to improve probing
- ▶ Removal of padding on variable-length data fields
- ▶ Improved FETCH FIRST N ROWS processing

3.1.18 Inline LOB support

Prior to DB2 10, DB2 for z/OS stored each LOB column, one per page, in a separate auxiliary (LOB) table space, regardless of the size of the LOB to be stored. All accesses to each LOB, (SELECT, INSERT, UPDATE, and DELETE) had to access the auxiliary table space using the auxiliary index.

A requirement with LOB table spaces is that two LOB values for the same LOB column cannot share a LOB page. Thus, unlike a row in the base table, each LOB value uses a minimum of one page. For example, if some LOBs exceed 4 KB but a 4 KB page size is used to economize on disk space, then it might take two I/Os instead of one to read such a LOB, because the first page tells DB2 where the second page is.

DB2 10 supports inline LOBs. Depending on its size, a LOB can now reside completely in the base table space along with other non-LOB columns. Any processing of this inline LOB now does not have to access the auxiliary table space. A LOB can also reside partially in the base table space along with other non-LOB columns and partially in the LOB table space (that is, a LOB can be split between the base table space and LOB table space). In this case, any processing of the LOB must access both the base table space and the auxiliary table space.

Inline LOBs offer the following benefits:

- ▶ Small LOBs that reside completely in the base table space can now achieve similar performance to similarly sized VARCHAR columns.
- ▶ Inline LOBs avoid all getpages and I/Os that are associated with an auxiliary index and LOB table space.
- ▶ Inline LOBs can save disk space even if compression cannot be used on the LOB table space.
- ▶ The inline piece of the LOB can be indexed using index on expression.
- ▶ Inline LOBs can access small LOB columns with dynamic prefetch.
- ▶ The inline portion of the LOB can be compressed.

A default value (other than an empty string or NULL) is supported.

The DB2 10 LOAD and UNLOAD utilities can load and unload the complete LOB along with other non-LOB columns.

Inline LOBs are only supported with either partition-by-growth or partition-by-range universal table spaces. Reordered row format is also required. Inline LOBs take advantage of the reordered row format and handle the LOB better for overall streaming and application performance. Additionally, the DEFINE NO option allows the row to be used and the data set for the LOB not to be defined. DB2 does not define the LOB data set until a LOB is saved that is too large to be completely inline.

3.1.19 Dynamic statement cache enhancements

In DB2 10, more SQL can be reused in the cache across users. Dynamic SQL statements can now be shared with already cached dynamic SQL statements if the only difference between the two statements is literal values. In the dynamic statement cache, literals are replaced with an ampersand (&) that behaves similar to parameter markers.

Tip: To view IBM success stories on System z and get a glimpse of what others are doing with DB2 for z/OS, see this website:

<http://www.ibm.com/software/data/db2/zos/>

3.2 Effective database management with IBM DB2 tools

There are clearly additional facilities required to manage and maintain an enterprise database environment. The following list provides a brief explanation of the IBM DB2 tools available. For more information and a broader view, see this website:


<http://www.ibm.com/software/data/db2imstools/products/db2-zos-tools.html>

Here is a brief explanation of the IBM DB2 tools:

- ▶ *IBM DB2 Utilities Suite for z/OS* is a comprehensive tool set for managing DB2 data maintenance tasks. Use the DB2 Utilities Suite to quickly and easily build utility jobs while helping to ensure the highest degree of availability, performance, and data integrity.

- ▶ *IBM DB2 Sort for z/OS* is a prime example of ongoing IBM efforts to respond to customer needs. Developed to address specific customer requests, DB2 Sort is designed to help improve performance, reduce elapsed time and decrease CPU utilization during utility sort processing.
- ▶ *IBM DB2 Utilities Enhancement Tool for z/OS* facilitates control over databases by monitoring and managing all thread activity. When combined with DB2 Sort and the DB2 Utilities Suite, the DB2 Utilities Enhancement Tool helps eliminate the need for duplicate utilities from third-party vendors, minimizing the use of system resources and reducing total cost of ownership.
- ▶ *IBM DB2 Automation Tool for z/OS* helps automate common utility tasks, an effective way to streamline database administration. With the DB2 Automation Tool for z/OS, you can define the utility job conditions—stored in easy-to-use profiles—that determine whether, when, and how a utility is run. The DB2 Automation Tool can help you increase database availability by shrinking batch windows, conserving system resources, and maximizing administrative productivity.
- ▶ *IBM DB2 Administration Tool for z/OS* helps simplify a full range of database management functions so administrators at any level of skill and experience can handle key tasks. For example, administrators can quickly navigate the DB2 catalog and build and execute dynamic SQL statements, all without knowing the exact SQL syntax.
- ▶ *IBM DB2 Object Comparison Tool for z/OS* helps accelerate the process of changing DB2 objects throughout the application life cycle. The tool compares objects from different sources and automatically produces reports describing differences. It then generates the information needed to apply changes. DB2 Object Comparison Tool helps synchronize files by staging and propagating changes between environments.
- ▶ *IBM DB2 High Performance Unload for z/OS* provides fast and efficient ways to unload and extract data for movement across enterprise systems or for in-place reorganizations. In many cases, unloading data during an object rebuild phase can slow the change process and negatively affect DB2 application performance. Integrating DB2 High Performance Unload into the change process can help reduce the maintenance time for critical DB2 applications and, consequently, increase database availability.
- ▶ *IBM Tivoli® IBM OMEGAMON® XE for DB2 Performance Expert on z/OS* is a single, comprehensive assessment tool that enables administrators to monitor, analyze, and tune the performance of DB2 and DB2 applications on z/OS. The tool combines real-time monitoring, historical tracking, and batch reporting with buffer pool analysis and expert database analysis to help you identify potential problems and maximize performance.
- ▶ *IBM DB2 Buffer Pool Analyzer for z/OS* is integrated into Tivoli OMEGAMON XE for DB2 Performance Expert. This tool enhances the performance of DB2 by helping to optimize memory usage, CPU cycles, online response times, and batch times. DB2 Buffer Pool Analyzer delivers information about current buffer pool performance and uses statistical data derived from databases to create recommendations for improving application performance.
- ▶ *DB2 Query Monitor for z/OS* allows you to efficiently customize and tune SQL workloads and DB2 objects to improve the effectiveness of DB2 subsystems and enhance overall performance. DB2 Query Monitor provides access to current and historical views of query activity throughout DB2 subsystems from a single console, so you can identify problems quickly and respond immediately.

- ▶ *IBM Optim™ Query Workload Tuner for DB2 for z/OS* can further enhance workload performance by providing expert recommendations to help focus your efforts. You can monitor and manage the performance of SQL queries and query workloads to optimize the physical database design for improved performance. Optim Query Workload Tuner can also help reduce the DB2 total cost of ownership while minimizing specialized skill requirements for database management.
- ▶ *IBM DB2 Change Accumulation Tool for z/OS* helps accelerate recovery of key business functions in the event of a disaster or large-scale outage. It actively collects and stores backup information about the specific objects that administrators identify as most important to the business. By keeping current and consistent data images readily available, the DB2 Change Accumulation Tool facilitates rapid recovery.
- ▶ *IBM DB2 Recovery Expert for z/OS* is a self-managing recovery solution that can help you recover DB2 data with minimal disruption. By automating the process of rebuilding assets to a specified point in time, the tool helps simplify key tasks without interrupting database and business operations. It also delivers comprehensive analysis of altered, incorrect or missing database objects for superior manageability.
- ▶ *IBM DB2 Cloning Tool for z/OS* accelerates the cloning process, enabling you to create DB2 table and index spaces, or entire DB2 subsystems, rapidly for testing, development or maintenance tasks without affecting availability. You can cut the time and costs of traditional methods so you can focus on other tasks while allowing users to query the database on the clone during the time the online system is down.
- ▶ *IBM DB2 Log Analysis Tool for z/OS* helps minimize the need for DB2 recoveries while maintaining data integrity. With detailed reporting, the tool facilitates the identification, isolation, and restoration of unwanted changes to DB2 objects. In addition, it can help your organization meet data change analysis requirements for regulatory compliance.
- ▶ *IBM DB2 SQL Performance Analyzer* helps improve DB2 application design by taking SQL statements from any input source and providing estimated query costs. With this tool, application developers and database administrators can quickly and cost-effectively explore various “what-if” scenarios to evaluate the performance of different database design options and production volumes.
- ▶ *IBM DB2 Path Checker* and *IBM DB2 Bind Manager* enhance the value of DB2 SQL Performance Analyzer. DB2 Path Checker provides information about potential access path changes before they occur and information about changes that occur after a rebind. DB2 Bind Manager automatically processes only the necessary binds, helping to minimize bind operations and conserve resources.
- ▶ *IBM Guardium® for z/OS* helps meet the growing number of requirements placed on companies for achieving regulatory compliance. This solution creates a centralized repository for controlling, summarizing, and reporting data. Controllers can view high-level trending of audit anomalies and generate detailed reports without database administrator involvement. By keeping information in a hardened environment, IBM Guardium helps you better control access and protect audit data.



Maximizing your existing System z investment

More commercial transactions today are processed on mainframes than on any other platform. Many companies rely on System z applications that perform back-office, core functions that are typically hidden from end users, but are critically important to the business. For many mature organizations, substantial capital investment has gone into developing, testing, deploying, and supporting such applications, and the objects that they access and run, over a span of many years. With this significant investment to consider, the ability to leverage existing tooling, knowledge, and skills is an essential part of the planning process for deploying any business analytics solution on an enterprise scale.

You might have personnel in your organization who have developed skills and proficiency with QMF Classic over time and, like many customers, your existing inventory of QMF Classic objects can number in the thousands. With such large inventories, redeveloping, retesting, and redeploying these objects can be so cost-prohibitive that it is not worth serious consideration.

In this chapter, we show you how easy it is to access, edit, use, and run existing QMF Classic objects in QMF for Workstation and WebSphere, allowing you to fully reuse or repurpose these objects for more modern data delivery and presentation. We also cover how you can use the QMF High Performance Option (HPO) to gain efficiencies with existing programs as well as to administer and manage the QMF Classic environment.

Even if you have not used QMF Classic before, you might want to make use of new features that allow you to run any application in TSO directly from the workstation environment and return results to the calling program. These features, which we examine here in more detail, allow you to leverage existing applications in System z even if they are not written specifically for QMF Classic.

This chapter covers the following topics:

- ▶ Overview of QMF Classic capabilities
- ▶ Using QMF Classic perspective within QMF for Workstation
- ▶ The QMF catalog: Accessing QMF Classic objects from QMF for Workstation
- ▶ Directing work to System z from workstation environments

- ▶ Using QMF HPO to manage and administer the QMF Classic environment

4.1 Overview of QMF Classic capabilities

Most of this chapter focuses on how you can maximize your existing QMF Classic investment as you begin to use QMF for Workstation and QMF for WebSphere. However, this topic provides an overview of QMF Classic capabilities in the event you are not already familiar with them.

4.1.1 QMF Classic Version 10 enhancements

Enhancements to QMF Classic Version 10 are summarized next. You can find more information about each of these enhancements in the Information Management for z/OS Solutions information center at the following website:

<http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp>

QMF Version 10 provides the following capabilities:

- ▶ Start QMF for TSO as a DB2 stored procedure.
We cover this enhancement in more detail in 4.4, “Directing work to System z from workstation environments” on page 51.
- ▶ Run multiple SQL statements in a single query.
- ▶ Run SQL queries that are longer than 32 KB, up to 2 MB long.
- ▶ Use 64-bit extended virtual storage in lieu of a spill file.
- ▶ Enable support for DB2 predictive governor settings (SQL code +495) on the QMF DISPLAY TABLE command.
- ▶ Make use of implicit data type casting on the SAVE DATA and IMPORT TABLE commands.
- ▶ Set a default OWNER value for the LIST command by a new global variable.
- ▶ Restrict which commands cause the ‘Last Used’ timestamp on a QMF object to be updated.
- ▶ Pass up to 63 parameters on the CALL statement; 32 were allowed in prior releases.
- ▶ Work more effectively with XML data. Rows that contain XML data can now be up to 2 GB. This increased limit applies to import and export operations with XML data as well.
- ▶ Set support for row lengths up to 2 GB in QMF reports.
- ▶ Set the CURRENT SCHEMA, CURRENT REFRESH AGE, and CURRENT QUERY ACCELERATION special registers in a QMF SQL query.
- ▶ Work more efficiently with decimal floating-point (DECFLOAT) data. This data type is fully supported in QMF Version 10 (where hardware allows it).
- ▶ Work with increased precision on data with the TIMESTAMP data type as well as work with TIMESTAMP WITH TIME ZONE data.
- ▶ Set whether you want the default column heading in QMF forms to be a database label or the column name.
- ▶ Set notification of positive SQL codes at varying levels of detail.
- ▶ Troubleshoot more effectively, with SQLCA information displayed on error message help panels.

- Independently govern database activity generated by user commands.
The QMF packages have been restructured so that modules that handle database activity driven by end user commands are now separate from modules that handle database activity driven by SQL internal to QMF. This new structure allows you to more effectively isolate groups of commands for independent governing.
- Set DB2 for z/OS concurrent access resolution options from within QMF.
- Install QMF more easily. Streamlined installation instructions, combined with the removal of QMF modes, result in a simplified installation process.

4.1.2 The QMF Classic user interface

As mentioned in Chapter 1, “Assessing business analytics solutions” on page 1, QMF Classic Edition consists of the QMF for TSO and CICS component of the QMF product family. QMF for TSO and CICS provides query, reporting, and charting capabilities, as well as procedure and application development functions, for users who work entirely from 3270 terminal emulators.

Navigation between queries, forms, procedures, and reports is quick and easy in QMF for TSO and CICS. When a user begins a QMF session, several named temporary storage areas are created in QMF memory for the various types of objects that can be created. These object types are depicted in Figure 4-1. The report and form objects are created by QMF when query results are returned.

You can navigate between the temporary storage areas by issuing either the SHOW or DISPLAY command, followed by the name of the temporary storage area, or simply press one of the PF keys.

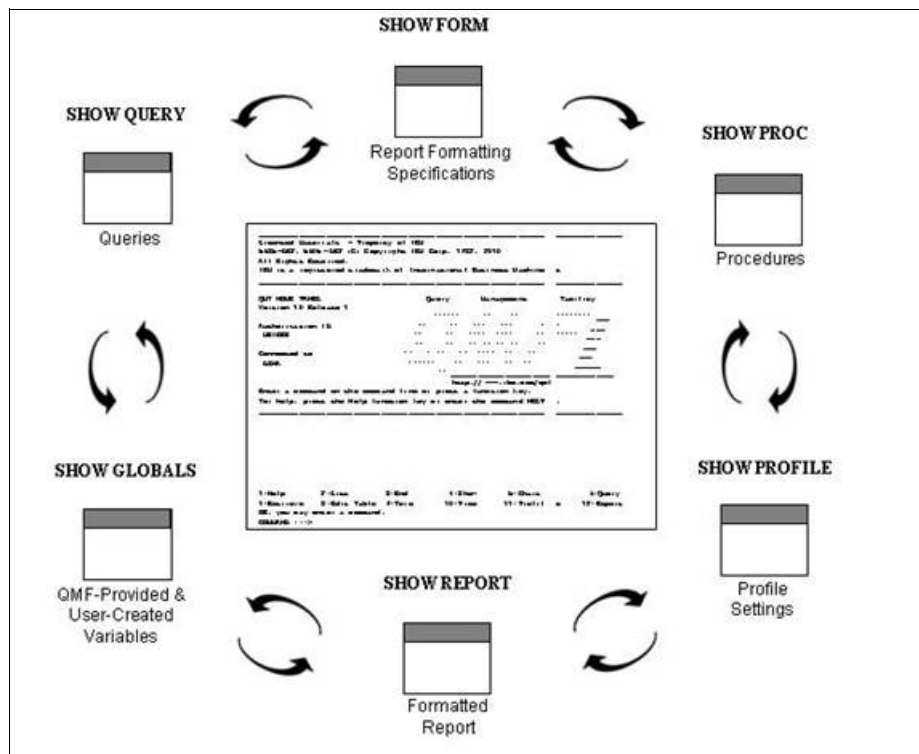


Figure 4-1 Use the SHOW command or function keys to navigate among the various object work areas

Anything created in these work areas is retained in memory for the duration of the QMF session. The SAVE command can be issued to save QMF queries, procedures, forms, and profiles to the database. The SET GLOBALS command can be issued to set global variables that control aspects of the session.

4.1.3 Developing a query in QMF for TSO and CICS

QMF offers the following query methods, which are suited to different audiences depending on level of SQL knowledge. Though there are three query methods, there is a single QUERY work area, storing one query in memory at any given time. Queries can be issued to any database in the DB2 family:

- ▶ Prompted query:

Prompted query is an easy-to-use query method for users who need to simply view data. This method prompts the user to choose tables, columns, row conditions, and sort criteria, then builds the underlying SQL SELECT statements behind the scenes. Users can issue the SHOW SQL command to see the underlying statements at any point along the way, or they can use the CONVERT command to convert the query to SQL when finished.

- ▶ SQL query:

This interface allows those experienced with SQL to issue SQL statements directly to the database. The DRAW command can be used to assist with syntax and column names if necessary.

- ▶ Query By Example (QBE):

This interface graphically represents the columns in the tables referenced in the query, and users type abbreviations for SQL statements beneath the columns to issue different types of queries.

In addition to issuing INSERT, UPDATE, and DELETE statements by the SQL query and QBE interfaces, users can also use the QMF Table Editor to search for and change the rows in question.

4.1.4 Formatting the report

Using QMF for TSO and CICS forms, you can change text, spacing, and alignment on virtually any area of the initially returned report. You can also do the following tasks:

- ▶ Group, aggregate, and summarize data.
- ▶ Define new columns not originally present in the query results.
- ▶ Perform calculations on your data using either simple operators or REXX expressions.
- ▶ Define conditional formatting, which allows you to define data-dependent formatting variations for the report.
- ▶ Fix columns in a large report so that you can easily compare later columns against earlier ones.

When you make changes to a QMF form, you can simply press the Report key or issue the SHOW REPORT command to navigate to the report, seeing formatting results immediately without having to repeatedly fetch information from the database.

Though there is a single QMF FORM work area, there are actually nine form panels, as depicted in Figure 4-2. You can navigate among these form panels by issuing the SHOW command, followed by the fully qualified form name. FORM.BREAK panels are numbered from 1 through 6 because QMF allows you to specify up to six breaks in the report.

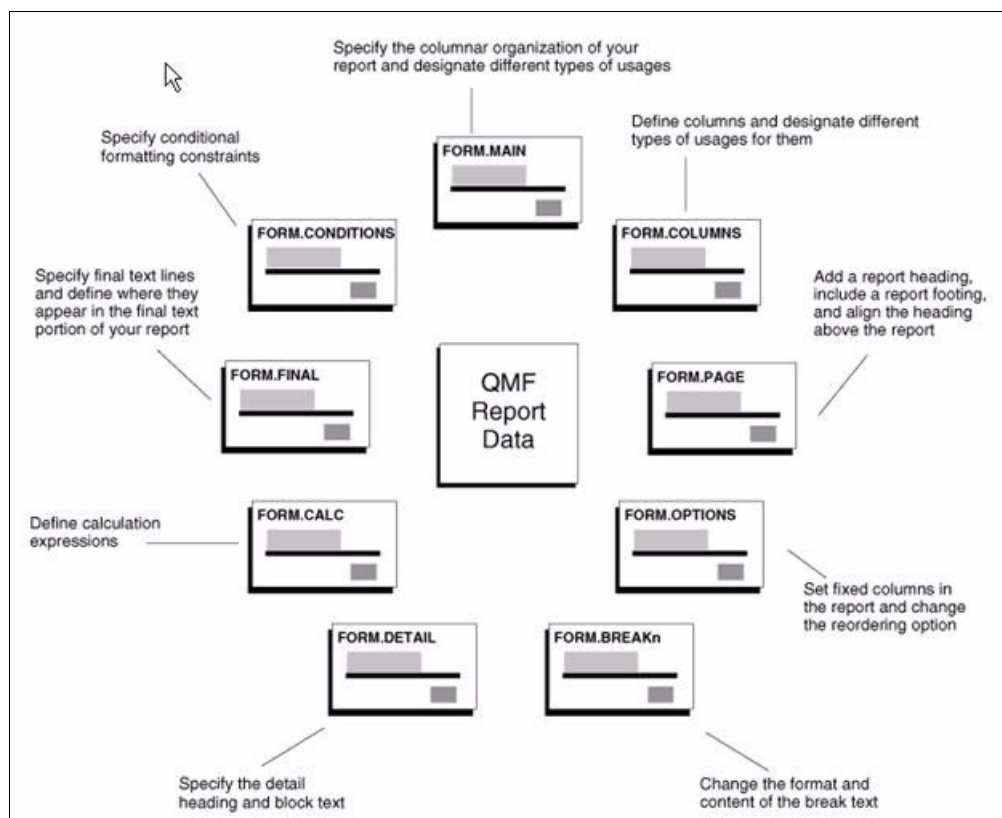


Figure 4-2 The nine QMF form panels

The FORM.MAIN, FORM.COLUMNS, and FORM.CALC panels allow you to specify edit codes. An edit code is a set of characters that tells QMF how to format and punctuate the data in a specific column of a report. Users can specify any of a number of QMF-provided edit codes. An exit interface is also provided for edit codes if you want to create your own formatting routines and associate your own codes with them.

4.1.5 Charting your data

QMF allows you to chart your data using several charting styles, listed next. Charting functions require the IBM GDDM®-PGF Interactive Chart Utility.

- ▶ Bar
- ▶ Histogram
- ▶ Line
- ▶ Pie
- ▶ Polar
- ▶ Scatter
- ▶ Surface
- ▶ Tower
- ▶ Table

4.1.6 Developing procedures and applications

A procedure is a QMF object that enables you to issue multiple commands with a single RUN command. Procedures in QMF for TSO and CICS can take advantage of sophisticated data and object management and can help you make more efficient use of resources. QMF offers *linear procedures*, which can contain only QMF commands, and *procedures with logic*, which can contain QMF commands as well as REXX statements and logic. Procedures with logic can make calls to the operating system or any other environment and are offered only with QMF for TSO.

Figure 4-3 shows a simple procedure that Spiffy Insurance has created. This linear procedure runs a query called SPIFFY_STAFF, which selects all rows from an employee table. The RUN command that runs the query specifies that formatting specifications that have been saved as STAFF_FORM should be used to format the report. STAFF_FORM simply changes the column heading of the employee ID column. The procedure then displays the report. The INTERACT command allows the user to interact with QMF at that point in the procedure, providing a means to browse the report. After the user exits the report, the procedure runs the next command, which displays the query in the event that changes to the query are needed.

```
PROC                                     LINE    1

RUN SPIFFY_STAFF (FORM=STAFF_FORM
-- SELECT * FROM SPIFFY EMPLOYEE TABLE, USE STAFF_FORM TO CHANGE ID COL TITLE
SHOW REPORT
-- DISPLAY RESULTING REPORT
INTERACT
-- ALLOW USER TO BROWSE THE REPORT, RETURN TO PROC ON REPORT EXIT
SHOW QUERY
--REDISPLAY SPIFFY_STAFF QUERY IN CASE USER DECIDES TO MAKE CHANGES

*** END ***

1=Help      2=Run      3=End      4=Print    5=Chart    6=Query
7=Backward  8=Forward   9=Form    10=Insert  11=Delete  12=Report
OK, cursor positioned.
COMMAND ==>                                SCROLL ==> PAGE
```

Figure 4-3 A linear procedure on the QMF PROC panel

QMF Classic also provides the capability to seamlessly incorporate QMF for TSO and CICS features and functions into your business applications, or do the opposite, calling existing applications from within QMF. The following application interfaces are provided:

- ▶ Stored procedure interface:

The stored procedure interface allows any software program that can call a DB2 for z/OS stored procedure, such as QMF for Workstation and WebSphere, to start QMF for TSO, run a predefined QMF query or procedure, and receive up to 20 reports back as result sets. We cover this interface in more detail in 4.4, “Directing work to System z from workstation environments” on page 51.

- ▶ **Callable interface:**

You can build sophisticated application suites by using the QMF callable interface and a variety of supported programming languages. The callable interface lets you integrate QMF functions into Interactive System Productivity Facility (ISPF) applications or applications written in C, COBOL, FORTRAN, High Level Assembler, PL/I, or REXX.

- ▶ **Command interface:**

The command interface allows you to use QMF services from an ISPF dialog. You can integrate QMF for TSO and CICS functions within ISPF dialogs so that users see only ISPF menus.

4.1.7 Authentication methods and security

In QMF for TSO and CICS, users are authenticated through the user IDs stored in the QMF profiles control table. You can configure QMF for these types of authentication:

- ▶ Open authentication, in which users who do not have specific QMF user IDs assigned can use QMF under the generic SYSTEM user ID.
- ▶ Restricted authentication, in which the user ID used to log onto QMF must match one of the unique IDs stored in the QMF profiles control table. In QMF for TSO, you can choose to have QMF authenticate users by using their database authorization IDs or their TSO logon IDs.

QMF can also be used in conjunction with other System z security facilities, such as IBM RACF®.

4.1.8 Customizing the QMF work environment for users and groups

There are many ways that you can customize functions and preferences in QMF for TSO and CICS:

- ▶ Create QMF profiles for individual users or groups of users, which control preferences for printing, query interfaces, and other common QMF functions.
- ▶ Create procedures and applications tailored to your specific business needs and then customize both QMF commands and function keys to allow users to run those applications.

You can also use the MESSAGE command to define a message that appears on a QMF object panel when your application ends:

- ▶ Use QMF program parameters and global variables both at startup time and in your business applications to customize settings for storage, tracing, operating mode (interactive or batch), and other aspects of the QMF operating environment.
- ▶ Create your own edit codes for QMF forms. These user-defined codes format the data in ways that are defined by an underlying data formatting routine that you create.
- ▶ Use ISPF, GDDM, or another panel manager to create application panels that run complex queries and create reports, prompt users for necessary information, and provide customized help.

4.1.9 Connecting to remote databases in the DB2 family

QMF for TSO and CICS allows you to configure access to any database in the IBM DB2 family. When you start QMF for TSO or CICS, the system from which you start the program is known as the *local* system. The DB2 database that resides on this system, where QMF is installed, is known as the *local database*. You can access objects that are stored in databases other than the local database in two ways in QMF for TSO and CICS, as described in the following sections.

Using the QMF CONNECT command

After you have started QMF, you can use the QMF CONNECT command to connect to the remote database. This method is known as *remote unit of work*.

After the connection has been made, you can access and use data as well as QMF objects (queries, procedures, and forms) at the remote database in the same way as you would work with them locally. You can even configure QMF to connect to the remote database before the QMF home panel is displayed to end users, creating a seamless connection.

Using three-part-names

When QMF is connected to a DB2 for z/OS database, you can issue a QMF command that refers to a table or view by a three-part name, which references the name of the remote DB2 for z/OS database in which the data is stored. This method of access is known as *distributed unit of work*.

Three-part names cannot refer to QMF queries, procedures, and forms that are stored in a remote database. To access these objects in a remote database, you must explicitly connect to the database as explained before.

For installation paths that specify how to prepare a remote server for access by the QMF CONNECT command or three-part-names, see *Installing and Managing DB2 QMF for TSO and CICS*, GC19-2886.

4.2 Using QMF Classic perspective within QMF for Workstation

QMF for Workstation now provides QMF Classic users with a view that should be entirely familiar to them. The QMF Classic perspective provides the same look and feel as that found in QMF for TSO and CICS.

To open this perspective, list all perspectives by clicking the Open Perspective icon in the upper left corner of the interface, as shown in Figure 4-4. Select **Other**, then **QMF Classic**, from the resulting context menus.

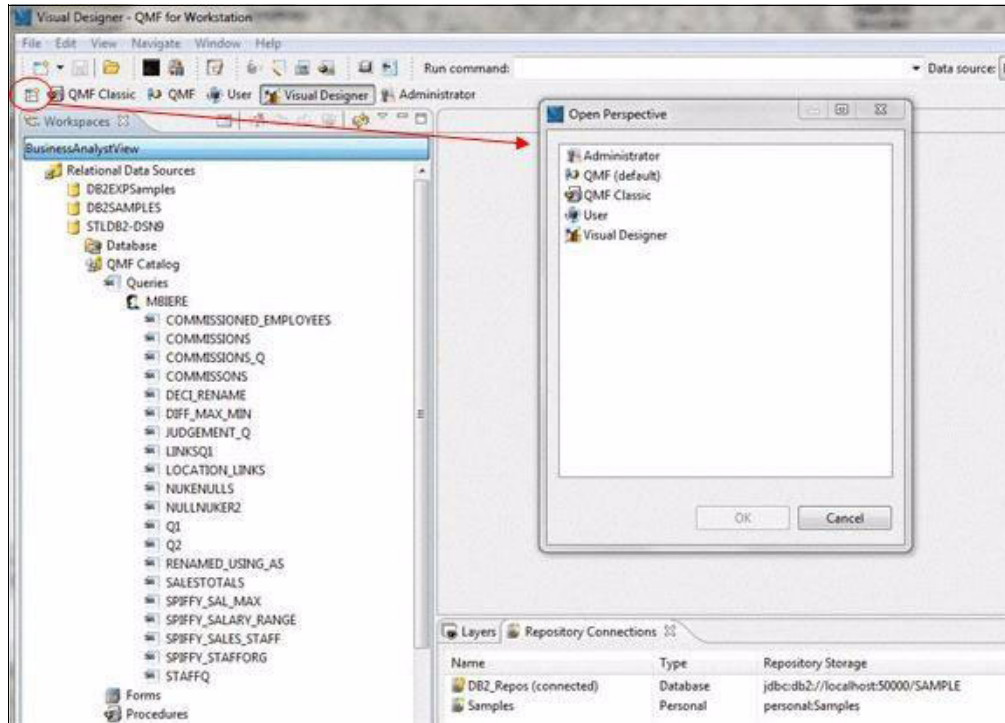


Figure 4-4 Switching to the QMF Classic perspective

Within the QMF Classic perspective, users can interact with QMF for Workstation much as they would in the TSO and CICS environments. For example, in Figure 4-5, the user has entered the command `DRAW Q.STAFF` to display a template SQL query for that table.

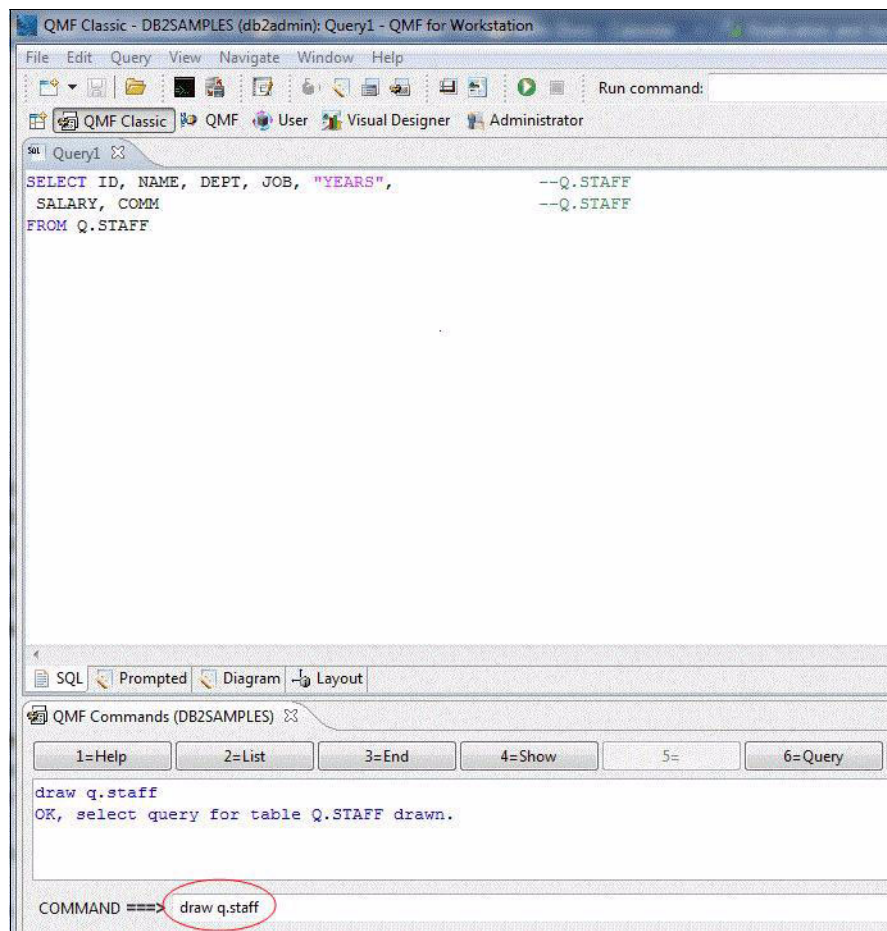


Figure 4-5 Using the `DRAW` command in the QMF Classic perspective

When the `DRAW` command completes, both the prompted and diagram query interfaces are also available for the user to continue working on the query if necessary. These interfaces are explained in further detail in Chapter 9, "Getting to the data you need: Query methods" on page 191.

4.3 The QMF catalog: Accessing QMF Classic objects from QMF for Workstation

In Chapter 6, "Configuring access to data sources and populating user workspaces" on page 89, we cover how to set up data source connections and repository services after you install QMF for Workstation. One of the decisions that you will be making as you move through that process is whether to enable the QMF catalog plug-in. The QMF catalog plug-in enables QMF for Workstation users to work with QMF for TSO and CICS objects (queries, forms, and procedures) that are stored in the QMF object catalog. New QMF for Workstation queries, classic report forms, and procedures can also be listed, accessed, edited, and run from QMF for TSO and CICS when this plug-in is enabled and those objects have been saved to the QMF catalog.

Figure 4-6 shows the repository mapping of some objects in the QMF catalog. The user ID shown in this example has the authority to list all objects owned by any user at any location. Traditional QMF for Workstation deployments would likely provide more stringent filters or restrictions on what end users see.

The query shown in Figure 4-6, SPIFFY_SALARY_RANGE, can be run from here by clicking the **Run Query** icon highlighted in Figure 4-6.

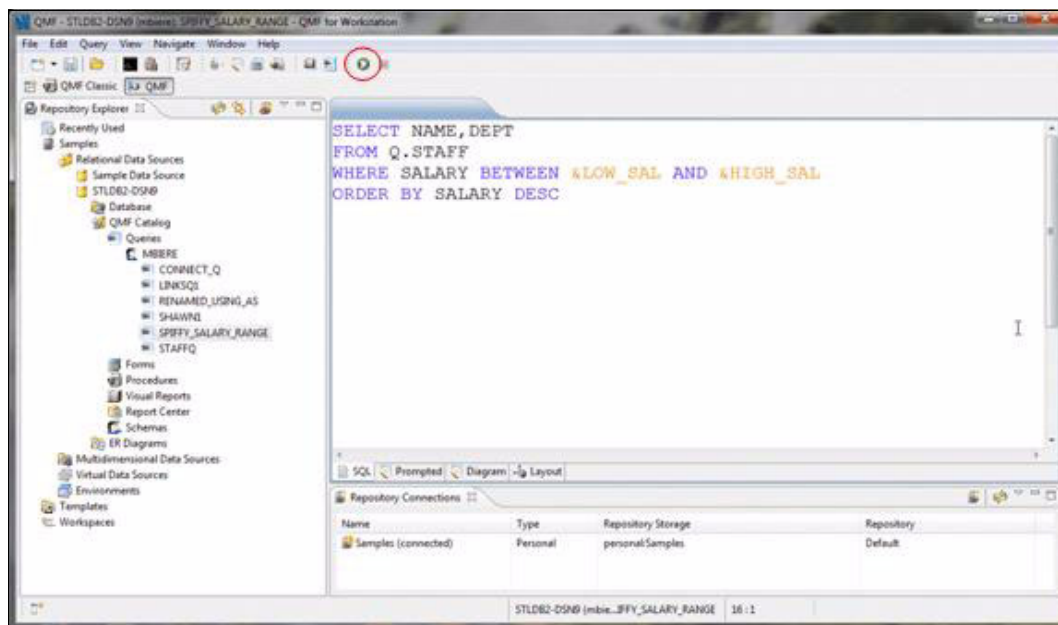


Figure 4-6 Running the SPIFFY_SALARY_RANGE query in QMF for Workstation

When the query is run, QMF for Workstation prompts for variable values, as shown in Figure 4-7.

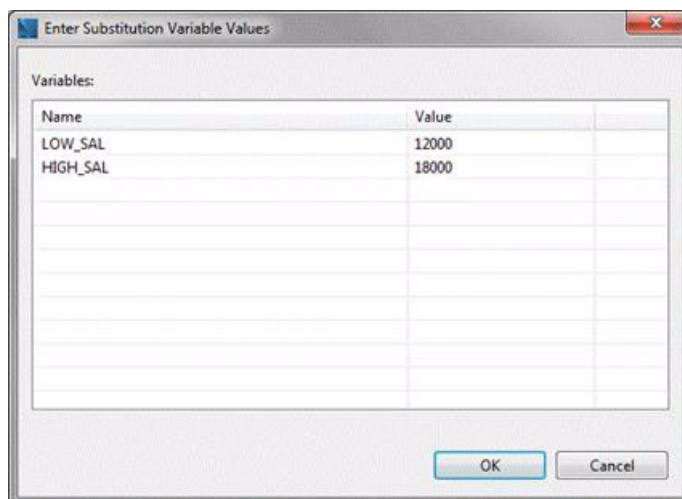


Figure 4-7 Providing variable values for the query

Results are shown in the query results view, identified by the tab labeled Results in Figure 4-8.

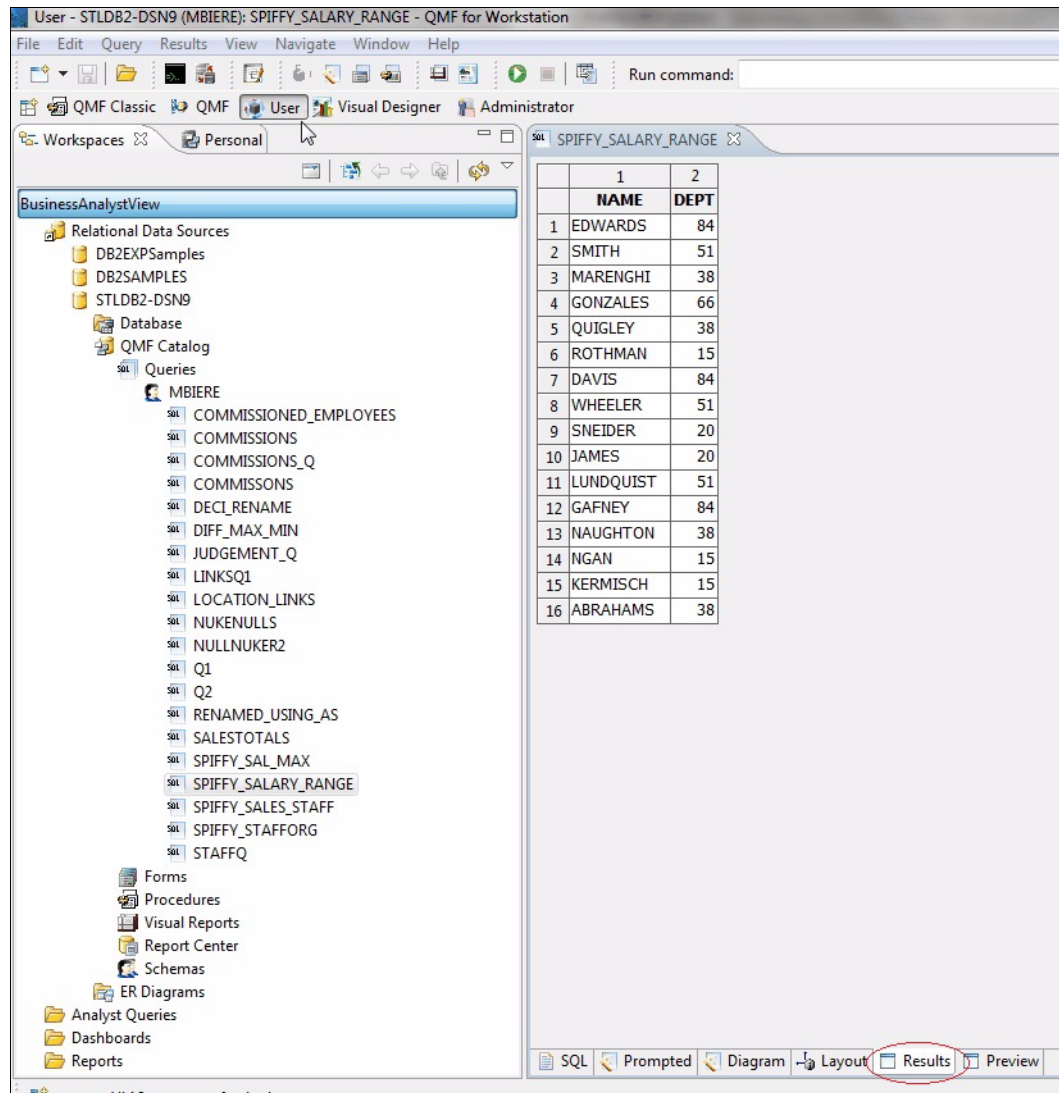


Figure 4-8 Viewing query results in QMF for Workstation

4.4 Directing work to System z from workstation environments

The QMF for TSO stored procedure interface enables users who might be unfamiliar with System z to direct work to TSO from the workstation environment and receive results back to the desktop. This interface thus provides streamlined access to features and resources that might exist only on System z, such as local date and time exits or user-defined edit codes for forms in QMF for TSO. Network traffic, and therefore processor time and total cost of ownership, are kept to a minimum because send and receive operations are reduced to a single CALL and return.

This topic explains:

- ▶ How you can use the RUNTSO command from QMF for Workstation to run a QMF query or procedure on System z.
- ▶ How applications other than QMF for Workstation can start QMF as a DB2 for z/OS stored procedure.

Any software program that can run a DB2 for z/OS stored procedure can start QMF for TSO using the stored procedure interface. We show you the right CALL syntax to use.

- ▶ How to include a command in your procedure to run TSO applications after QMF for TSO has been started as a stored procedure.

Applications do not have to be written specifically for QMF to be run through the QMF stored procedure interface.

- ▶ How to use the stored procedure interface to submit z/OS batch jobs from remote clients in workstation environments.

4.4.1 The RUNTSO command in QMF for Workstation

The RUNTSO command allows QMF for Workstation and WebSphere users to start QMF for TSO as a DB2 for z/OS stored procedure and pass a query or procedure name, as well as any parameters those objects require, on the command. When the RUNTSO command is issued, QMF issues the CALL statement shown in 4.4.2, “How to start QMF for TSO as a DB2 stored procedure” on page 53 behind the scenes. The query or procedure named on the command is then run noninteractively in QMF for TSO, which returns results to the calling application. To return results in a result set, the procedure specified on the RUNTSO command must include the following command for each result set desired:

PRINT REPORT (PRINTER=' ')

Up to 21 result sets can be returned from the procedure, including a result set for trace data. In addition to allowing the user to pass the name of the object to run, the syntax of the RUNTSO command includes parameters to set the desired trace level, location of the trace data, and the national language in which QMF for TSO will run.

Before users can issue the RUNTSO command from either QMF for Workstation or QMF for WebSphere, the stored procedure interface must be installed and configured as explained in *Installing and Managing DB2 QMF for TSO and CICS*, GC19-2886.

Figure 4-9 shows an example of how to run a query named Q1 using the stored procedure interface. Simply type **RUNTSO Q1** in the command bar.

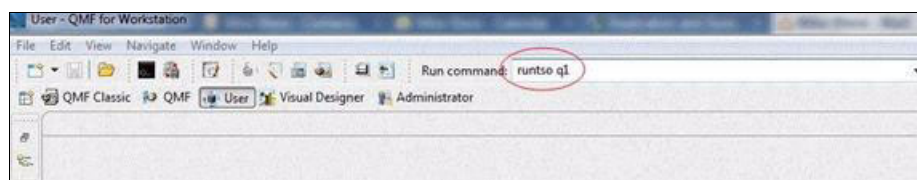


Figure 4-9 Using QMF for Workstation to run a query named Q1 in QMF for TSO

QMF for TSO immediately executes the query and produces the results shown in Figure 4-10.

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
10	SANDERS	20	MGR	7	18357.50	-
20	PERNAL	20	SALES	8	18171.25	612.45
30	MARENGHI	38	MGR	5	17506.75	-
40	O'BRIEN	38	SALES	6	18006.00	846.55
50	HAYES	15	MGR	10	20659.80	-
60	QUIGLEY	38	SALES	0	16808.30	650.25
70	ROTHMAN	15	SALES	7	16502.83	1152.00
80	JAMES	20	CLERK	0	13504.60	128.20
90	ROONITZ	42	SALES	6	18001.75	1386.70
100	PLOTZ	42	MGR	7	18352.80	-
110	NOAN	15	CLERK	5	12508.20	206.60
120	NAUGHTON	38	CLERK	0	12954.75	180.00
130	YAMAGUCHI	42	CLERK	6	10505.90	75.60
140	FRAYE	51	MGR	6	21150.00	-
150	WILLIAMS	51	SALES	6	19456.50	637.65
160	MOLINARE	10	MGR	7	22959.20	-
170	KERMISCH	15	CLERK	4	12258.50	110.10
180	ABRAHAM	38	CLERK	3	12009.75	236.50
190	SNEIDER	20	CLERK	8	14252.75	126.50
200	SCOUTTEN	42	CLERK	0	11508.60	84.20
210	LU	10	MGR	10	20010.00	-
220	SMITH	51	SALES	7	17654.50	992.80
230	LUNDQUIST	51	CLERK	3	13369.80	189.65
240	DANIELS	10	MGR	5	19260.25	-
250	WHEELER	51	CLERK	6	14460.00	513.30
260	JONES	10	MGR	12	21234.00	-
270	LEA	66	MGR	9	18555.50	-
280	WILSON	66	SALES	9	18674.50	811.50
290	QUILL	84	MGR	10	19818.00	-
300	DAVIS	84	SALES	5	15454.50	806.10
310	GRAHAM	66	SALES	13	21000.00	200.30
320	GONZALES	66	SALES	4	16858.20	844.00
330	BURKE	66	CLERK	1	10988.00	55.50
340	EDWARDS	84	SALES	7	17844.00	1285.00
350	GAFNEY	84	CLERK	5	13030.50	188.00

Figure 4-10 Report results from the Q1 query

4.4.2 How to start QMF for TSO as a DB2 stored procedure

The RUNTSO command provides a streamlined way to start QMF for TSO from the workstation for users who might be unfamiliar with stored procedure syntax. However, QMF for TSO can also be started as a stored procedure by any software program that is capable of calling a DB2 for z/OS stored procedure. We cover how to do that in this topic.

About the stored procedure interface to QMF for TSO

The QMF stored procedure interface is made available through a REXX stored procedure named Q.DSQQMFSP, which is external to QMF and must run in a WLM-managed address space (or, in the case of newer System z environments, an address space managed by the Unified Resource Manager).

To start QMF as a DB2 for z/OS stored procedure, first see *Installing and Managing DB2 QMF for TSO and CICS*, GC19-2886, which explains how to install the QMF stored procedure interface after you have completed the QMF for TSO base installation or migration. Be sure to run the installation verification procedure, as instructed, for both the base QMF installation as well as the installation of the interface. The QMF 10 libraries must be available to the WLM or URM environment.

CALL statement syntax for the Q.DSQQMFSP stored procedure

After you have installed and configured the stored procedure interface, simply issue a CALL statement to call the Q.DSQQMFSP stored procedure. Use the following syntax on the CALL statement:

```
CALL Q.DSQQMFSP ('object-name', 'trace-level', 'L2-destination', 'language',
'status-message')
```

object-name	<p>This input parameter names a QMF procedure or query that will run after QMF starts. If the object is a procedure, the procedure can be either a QMF linear procedure or a procedure with logic. The query or procedure that is named in this parameter must exist in the QMF catalog in the subsystem in which the stored procedure interface components are installed.</p> <p>One result set is returned if the specified object is a query; up to 21 result sets can be returned from a procedure (including trace output if indicated). QMF returns one result set to the calling program each time the following command is encountered in the procedure:</p> <p>PRINT REPORT (PRINTER=' '</p> <p>You can set the PRINTER option in the QMF profile as well.</p> <p>Because QMF cannot display panels when it has been started as a stored procedure, no prompting for variable values is available, so all values for the query or procedure you specify in this parameter must be also be supplied.</p>
trace-level	<p>This input parameter specifies the level of trace detail. If the parameter is left blank or is null or NONE is specified, no QMF trace output is generated. Other valid values include:</p> <p><i>L2</i></p> <p>This option traces QMF messages and commands at the highest level of detail. Where the trace output is sent depends on how the L2-destination parameter is set.</p> <p><i>ALL</i></p> <p>This option traces QMF activity at the highest level of detail, including program initialization errors and other errors that might occur before the profile is established. Trace output is sent to the DSQDEBUG DD card defined at the time that you install and configure the interface.</p>
L2-destination	<p>This input parameter specifies the destination for the trace log when trace-level is set to L2. Blank or null values return the trace output as the last result set from the stored procedure run. A value of DSQDEBUG sends the trace output to the DSQDEBUG DD card that is defined at the time you install and configure the interface.</p>
language	<p>This input parameter specifies the national language in which QMF will run. Specify a 1-character national language identifier found in Installing and Managing QMF for TSO and CICS. The default is English.</p>
status-message	<p>This output parameter contains the last message that was issued from the execution of the procedure or query that is passed in the object-name parameter. How the output parameter is defined depends on the software program that issues the CALL statement. For example, in QMF for Workstation, the output parameter containing the status message is defined as a question mark (?).</p>

A sample procedure and CALL statement

The following example procedure with logic, called STAFFPROC, completes these tasks:

1. Makes the SHARE=YES parameter the default on all QMF SAVE commands.
2. Sets the DSQEC_CC global variable to 0 to eliminate carriage control characters from result sets that are returned to the calling program.

- Retrieves the value of the DSQAO_STO_PROC_INT global variable, which indicates whether or not QMF for TSO was started as a DB2 for z/OS stored procedure. See *QMF Reference*, SC19-2889, for more information about this variable.
- Writes message text to the trace log that indicates whether or not QMF for TSO was started as a DB2 for z/OS stored procedure.
- Runs a query named STAFFQUERY, which is as follows:

```
SELECT * FROM Q.STAFF
WHERE NAME = &NAME AND
DEPT = &DEPT AND
JOB = &JOB
```

- Returns the query results to the calling program as a result set. For each result set that must be returned, code one PRINT REPORT command as follows:

```
PRINT REPORT (PRINTER=' ')
```

If you have several PRINT REPORT commands in the procedure, you can issue a SET PROFILE command, as shown in Figure 4-11, to set the PRINTER option of the profile to a string of blanks so that you do not have to specify the PRINTER option on each PRINT REPORT command. The procedure can include up to 20 PRINT REPORT commands that return result sets.

```
/* REXX procedure */
"SET GLOBAL (DSQEC_SHARE = 1"
"SET GLOBAL (DSQEC_CC=0"
"GET GLOBAL ("SPINT"=DSQAO_STO_PROC_INT)"
IF SPINT = 1 THEN
DO
"MESSAGE (TEXT 'YOU ARE RUNNING IN THE QMF STORED PROC INT.')"
"SET PROFILE (PR = ' ')"
END
ELSE
DO
"MESSAGE (TEXT 'YOU ARE NOT RUNNING IN THE QMF STORED PROC INT.')"
END
"RUN QUERY STAFFQUERY (&&NAME = &NAME &&DEPT = &DEPT &&JOB = &JOB"
"PRINT REPORT"
```

Figure 4-11 QMF procedure with logic that is run through the stored procedure interface

To start QMF for TSO as a DB2 for z/OS stored procedure and run the STAFFPROC procedure in Figure 4-11, issue a command such as this one from QMF for Workstation:

```
CALL Q.DSQMFSP
('STAFFPROC(&NAME='''O''''O'BRIEN''',&DEPT=38,&JOB='''SALES'''),'L2','','E',?)
```

This CALL statement then returns to the calling program a result set containing the desired row from the Q.STAFF sample table, as shown in Figure 4-12.

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
---	----	---	----	-----	-----	-----
40	O'BRIEN	38	SALES	6	18006.00	846.58

Figure 4-12 Result set returned as output from the CALL Q.DSQMFSP statement

To pass the entire text of a query when you start QMF, use a variable as a placeholder for the query text. Then create a procedure that runs the query, as shown in the following example.

We call this procedure RUNQPROC:

```
"SET PROFILE (PR = ' ')"
"RUN QUERY RUNQ (&QUERY=&QUERY)"
"PRINT REPORT"
```

The following example CALL statement starts QMF for Workstation and runs the RUNQPROC procedure, passing the entire query text on the first CALL statement parameter:

```
CALL Q.DSQQMFSP ('RUNQPROC(&QUERY=((SELECT CURRENT SERVER FROM
SYSIBM.SYSDUMMY1)))','L2','','E',?)
```

After QMF for TSO has been started as a DB2 for z/OS stored procedure, the procedure whose name you pass on the CALL statement can do any number of tasks, including launching any TSO application or a z/OS batch job, both of which we look at next.

4.4.3 Running TSO applications through the stored procedure interface

You can use the stored procedure interface to QMF for TSO to perform work directly in TSO after the Q.DSQQMFSP stored procedure has started. To do so, simply include the TSO command in the procedure that you pass on the CALL statement. Parameters and values following the TSO command are then sent by QMF directly to TSO and interpreted there. In this manner, you can run any TSO application, CLIST, or REXX exec that exists in your TSO environment, returning any result sets from those applications directly to the workstation environment. For more information about how to use the TSO command in a QMF procedure, see *QMF Reference*, SC19-2889.

4.4.4 Launching batch jobs through the stored procedure interface

Not only can you use the stored procedure interface to launch TSO applications, you can also use it to submit batch jobs directly to z/OS from remote workstation clients. The client can be any software program that can connect to the DB2 for z/OS subsystem where the QMF stored procedure interface is installed.

Follow these steps to use the QMF for TSO stored procedure interface to launch a z/OS batch job:

1. Install and configure the QMF for TSO stored procedure interface as explained in *Installing and Managing QMF for TSO and CICS*.
2. From the client, issue a SQL CALL statement that passes the name of a procedure that launches the batch job.

For example, the following statement starts QMF for TSO as a stored procedure by calling the Q.DSQQMFSP stored procedure and passes a procedure named LAUNCH_STAFF_REPORT (created by user W397754) as the first parameter:

```
CALL Q.DSQQMFSP('W397754.LAUNCH_STAFF_REPORT','L2','','','')
```

See 4.4.2, “How to start QMF for TSO as a DB2 stored procedure” on page 53 or *Installing and Managing DB2 QMF for TSO and CICS*, GC19-2886 for more information about the parameters of this statement.

The LAUNCH_STAFF_REPORT procedure in this example is shown in Example 4-1. The procedure uses the REXX FTP API to launch the batch job. See your z/OS information for details on how to use the REXX FTP API as well as the z/OS FTP and JES interfaces used in the batch job called by this procedure.

Example 4-1 QMF procedure that uses the REXX FTP API to launch a z/OS batch job

```
/* REXX - Launch staff report batch job. */
/*      This procedure uses the REXX FTP API to launch the batch */
/*      job. */
/*****/

/* Initialize REXX program variables */
hostname = "STLMVS1" /* z/OS host name to receive the batch job */
uid       = "W397754" /* z/OS user ID that is used to log into FTP */
uidpw     = "xxxxxxx" /* Password used to log into FTP */
pcode = 0 /* Initialize return code */
TRACEID = 'PAZ' /* Set FTP trace flag */

/* Create REXX FTP environment */
ftp_rc = ftpapi('fcai.', 'create', TRACEID)
if ftp_rc < 0 then
do
  pcode = -10
  exit pcode
end

/* Connect to host. See hostname variable for name of host */
ftp_rc = FtpApi('fcai.', 'init', '-w 300 '||hostname)
if ftp_rc < 0 then exit -21

/* Log user into FTP session */
cmd = "USER " uid
ftp_rc = FtpApi('fcai.', 'scmd', cmd, 'W')
if ftp_rc < 0 then exit -22

cmd = "PASS " uidpw
ftp_rc = FtpApi('fcai.', 'scmd', cmd, 'W')
if ftp_rc < 0 then exit -23

/* Set JES to receive JCL file */
cmd = "QUOTE SITE FILETYPE=JES"
ftp_rc = FtpApi('fcai.', 'scmd', cmd, 'W')
if ftp_rc < 0 then exit -24

/* Send JCL file to JES for execution */
cmd = "PUT 'W397754.STAFF.REPORT.JCL'"
ftp_rc = FtpApi('fcai.', 'scmd', cmd, 'W')
if ftp_rc < 0 then exit -25

/* Quit the FTP session */
cmd = "QUIT"
ftp_rc = FtpApi('fcai.', 'scmd', cmd, 'W')
if ftp_rc < 0 then exit -26

/* Terminate the REXX FTP environment */
ftp_rc = ftpapi('fcai.', 'term')
if ftp_rc < 0 then exit -27

exit pcode
```

Example 4-2 shows the batch job located in data set W397754.STAFF.REPORT.JCL. This job starts QMF in batch mode (using the M=B parameter) and runs a QMF procedure called STAFF_REPORT. This procedure runs a query to create a report that is then exported to a z/OS data set in HTML format. The second step of the job emails the report back to the client.

The QMF stored procedure is executed in a DB2 WLM (or URM) address space. The user ID under which the address space was started must have read access to the data set that contains the batch job.

Example 4-2 Sample z/OS batch job that runs a QMF procedure and emails the resulting report back to the workstation environment

```
//STAFFRPT JOB MSGLEVEL=1,TIME=(,30),REGION=2048K,
//          MSGCLASS=H,NOTIFY=W397754,USER=W397754
/*JOBPARM S=MVS1
//*****
//RUNQMF EXEC PGM=DSQQMFE,
//          PARM='M=B,I=W397754.STAFF_REPORT,P=DSQDEV,S=DB2P',
//          TIME=1440,DYNAMNBR=30,REGION=6M
//STEPLIB DD DSN=QMFDEV.DSQDEV.LOAD,DISP=SHR
//          DD DSN=DSN.DB2P.SDSNEXIT,DISP=SHR
//          DD DSN=DSN.DB2P.SDSNLOAD,DISP=SHR
//ADMGGMAP DD DSN=QMFDEV.DSQDEV.QFMFAPS,DISP=SHR
//DSQDEBUG DD SYSOUT=H,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1210)
//DSQDUMP DD SYSOUT=H,DCB=(RECFM=VBA,LRECL=125,BLKSIZE=1632)
//DSQPRINT DD SYSOUT=H,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//DSQSPILL DD DSN=&&SPILL,DISP=(NEW,DELETE),
//            UNIT=SYSDA,SPACE=(TRK,(100),RLSE),
//            DCB=(RECFM=F,LRECL=4096,BLKSIZE=4096)
//*****
//EMAILRPT EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DSN=W397754.STAFF.REPORT.EMAILCTL,DISP=SHR
//          DD DSN=W397754.STAFF.REPORT,DISP=SHR
//SYSUT2 DD SYSOUT=(A,SMTP),DCB=(LRECL=83,BLKSIZE=4096,RECFM=VB)
//SYSIN DD DUMMY
```

The STAFF_REPORT procedure in this example looks like this:

```
-- QMF procedure to query STAFF table and export report in HTML format
RUN QUERY STAFF_REPORT_QUERY
EXPORT REPORT TO STAFF (DATAFORMAT=HTML,CONFIRM=NO)
```

The STAFF_REPORT procedure runs the following query, called STAFF_REPORT_QUERY:

```
SELECT ID, "NAME", DEPT, JOB, "YEARS", SALARY, COMM
FROM Q.STAFF
ORDER BY DEPT, "NAME"
```

To email the staff report data set back to the client, SMTP control statements are needed to identify the email being sent. This control data set, W397754.STAFF.REPORT.EMAILCTL, which is shown in the batch JCL in Example 4-2, contains SMTP control statements and must precede the staff report in data set W397754.STAFF.REPORT.

Here are the SMTP statements used in this example, where servername is the name of the server where the QMF for TSO stored procedure (Q.DSQMFSP) is installed, and hostname is the fully qualified server name:

```
HELO servername
MAIL FROM:<W397754@hostname>
RCPT TO:<userid@us.ibm.com>
DATA
SUBJECT: Staff Report
MIME-VERSION: 1.0
CONTENT-TYPE: TEXT/HTML
```

For more information about how to use the z/OS SMTP mail server, see your z/OS information.

The staff report that is emailed back to the client is shown in Figure 4-13.

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
240	DANIELS	10	MGR	5	19260.25	-
260	JONES	10	MGR	12	21234.00	-
210	LU	10	MGR	10	20010.00	-
160	MOLINARE	10	MGR	7	22959.20	-
50	HANES	15	MGR	10	20659.80	-
170	KERMISCH	15	CLERK	4	12258.50	110.10
110	NGAN	15	CLERK	5	12508.20	206.60
70	ROTHMAN	15	SALES	7	16502.83	1152.00
80	JAMES	20	CLERK	-	13504.60	128.20
20	PERNAL	20	SALES	8	18171.25	612.45
10	SANDERS	20	MGR	7	18357.50	-
190	SNEIDER	20	CLERK	8	14252.75	126.50
180	ABRAHAMS	38	CLERK	3	12009.75	236.50
30	MARENGHI	38	MGR	5	17506.75	-
120	NAUGHTON	38	CLERK	-	12954.75	180.00
40	O'BRIEN	38	SALES	6	18006.00	846.55
60	QUIGLEY	38	SALES	-	16808.30	650.25
90	KOONITZ	42	SALES	6	18001.75	1386.70
100	PLOTZ	42	MGR	7	18352.80	-
200	SCOUTTEN	42	CLERK	-	11508.60	84.20
130	YAMAGUCHI	42	CLERK	6	10505.90	75.60
140	FRAYE	51	MGR	6	21150.00	-
230	LUNDQUIST	51	CLERK	3	13369.80	189.65
220	SMITH	51	SALES	7	17654.50	992.80
250	WHEELER	51	CLERK	6	14460.00	513.30
150	WILLIAMS	51	SALES	6	19456.50	637.65
330	BURKE	66	CLERK	1	10988.00	55.50
320	GONZALES	66	SALES	4	16858.20	844.00
310	GRAHAM	66	SALES	13	21000.00	200.30
270	LEA	66	MGR	9	18555.50	-
280	WILSON	66	SALES	9	18674.50	811.50
300	DAVIS	84	SALES	5	15454.50	806.10
340	EDWARDS	84	SALES	7	17844.00	1285.00
350	GAFNEY	84	CLERK	5	13030.50	188.00
290	QUILL	84	MGR	10	19818.00	-

Figure 4-13 Staff report emailed from z/OS back to the client

4.5 Using QMF HPO to manage and administer the QMF Classic environment

QMF High Performance Option (HPO) is a multifaceted tool that helps database administrators manage QMF objects and performance in the QMF Classic environment. HPO consists of two components:

- QMF HPO/Manager:

Using QMF HPO/Manager, you can govern (preemptively and in real time) ad-hoc and dynamic query and reporting activities. With easily collected, detailed information, you can more precisely control CPU resource usage at varying levels, according to any number of schedules applied to QMF Classic user groups.

- QMF HPO/Compiler:

With QMF HPO/Compiler, compiled programs that execute static SQL can be rapidly generated from QMF objects, thus saving CPU cycles. It is especially beneficial for reports that will be scheduled at regular intervals.

The remaining topics in this chapter cover the HPO/Manager and HPO/Compiler functions in more detail.

4.5.1 Managing QMF objects and tracking and governing session activity

QMF HPO/Manager is a family of utilities for managing and administering QMF Classic operations. The QMF HPO/Manager comprises the following integrated components:

- ▶ Governor module
- ▶ Activity log
- ▶ Online facilities

Governor module

This module replaces the default governor exit routine provided with QMF for TSO and CICS and is the direct interface to QMF processing for the QMF HPO/Manager. This enhanced governor module is more than a QMF governor because it services the following facilities:

- ▶ Object manager:

The object manager tracks QMF session activity. It records information about the commands and objects and writes this information directly to the activity log. You can also produce lists of QMF objects that are based on the content of a specific object. The object manager has a list filter that allows you to locate queries that contain references to specific table names, column names, SQL verbs, and so on. The object manager supports the migration and copying of objects to and from QMF for TSO and CICS. It recognizes and appropriately handles columns in the QMF object catalog.

- ▶ Governor:

The governor controls QMF session activity. It obtains thresholds and controls from resource groups in the same way as the QMF for TSO and CICS default governor, but provides a wider and more flexible set of controls. These controls enforce the proper use of resources in QMF sessions operating under TSO and CICS.

- ▶ Monitor:

The monitor supplies a real-time user interface to information about QMF session activity in TSO and CICS. It accepts administrator commands and passes them to the HPO governor module.

- ▶ Query analyzer:

The query analyzer provides preemptive governing capabilities. It traps queries before DB2 processes them and estimates their resource consumption. The query analyzer can cancel queries that are estimated to be too resource-intensive:

- Activity log:

The activity log provides a repository for QMF session activity and QMF object usage information. The governor module writes directly to the activity log data sets. You must run a batch job periodically to copy the activity log data sets to the activity log tables. You can use the activity log's JCL function to create the JCL to run this batch job.

- Online facilities:

The online facilities help organize and simplify the administration and management of QMF HPO. You can review and manipulate QMF for TSO and CICS objects by using the object manager's online facilities. Two types of actions are supported: those that operate on one object and those that can operate on a set of objects.

The QMF HPO/Manager helps you isolate production applications from query and reporting activities. A session activity list, shown in Figure 4-14, gives administrators essential facts about database activity, the number of rows that are fetched, and processor time consumption.

```

DB2A -- Session Activity List ----- ROW 1 TO 5 OF 5
COMMAND ==> SCROLL ==> CSR
RAAM018I--monitor data refreshed
Valid Actions Are...
B Browse SQL Text
C Cancel Current Action
rows that are fetched
TSOID : VNRSTRW
Mode : ONLINE C S
QMF Object Object A Q
A Date Time Act Owner Name Rows CPU N L
-----
05/09/10 07:47:52 BEG 0 0
05/09/10 07:47:52 RUN VNRSTRW MODELING 0 0
05/09/10 07:48:01 *** VNRSTRW MODELING 100 00 00 00 23 0 Y
05/09/10 07:48:01 *** VNRSTRW MODELING 1733 00 00 03 69 0
05/09/10 07:48:01 *** VNRSTRW MODELING 3330 00:00:07:20 0
***** BOTTOM OF DATA *****

```

Figure 4-14 The QMF HPO/Manager session activity list

Using the QMF HPO/Manager session activity list, QMF administrators can browse the SQL text associated with a query or cancel an active QMF command that is associated with database activity.

Defining and editing resource groups in QMF HPO/Manager

One of the HPO/Manager functions we look at in detail in this topic is the ability to define resource groups and apply limitations and restrictions on these groups. There are a number of options and restrictions that can be enforced. Figure 4-15 shows the resource allocation options you can apply to an individual or group.

```

DA1A / RS22DA1A -- Edit Resource Group: TECHSUP ----- 10.1.0
Command ==>
RAAP015I--Resource group common controls have been saved in the database
Schedule Number. : 1      Next Schedule Number to Process.... ==> 1
Start Date... ==> 19920101 Stop Date ==> 20991231 (YYYYMMDD)
Start Day... ==> 1      Stop Day. ==> 7      (1=SUN,2=MON,...)
Start Time... ==> 0000 Stop Time ==> 2400 (HHMM)
Active? (Y/N) ==> Y      Use Service Unit Defaults (Y/N).... ==> Y

Max CPU Seconds (Prompt Limit).....Online ==> 0
Max CPU Seconds (Cancel Limit).....Online ==> 0      Batch ==> 0
Max Rows Fetched (Prompt Limit).....Online ==> 0
Max Rows Fetched (Cancel Limit).....Online ==> 1000 Batch ==> 0
Allow GRANT,REVOKE SQL? (Y/N).....Online ==> Y      Batch ==> Y
Allow CREATE,ALTER,DROP SQL? (Y/N)....Online ==> Y      Batch ==> Y
Allow INSERT,UPDATE,DELETE SQL? (Y/N)..Online ==> N      Batch ==> Y
Allow TSO/CICS Commands? (Y/N).....Online ==> Y      Batch ==> Y
Allow EXPORT Command? (Y/N).....Online ==> Y      Batch ==> Y
Allow IMPORT Command? (Y/N).....Online ==> Y      Batch ==> Y
Allow SAVE DATA Command? (Y/N).....Online ==> Y      Batch ==> Y
Allow CONNECT Command? (Y/N).....Online ==> Y      Batch ==> Y
Enable QMF Access? (Y/N).....Online ==> Y      Batch ==> Y
Enable Query Analyzer? (Y/S/N).....Online ==> N      Batch ==> N
Service Units in 100s (Cancel limit)..Online ==> 0      Batch ==> 0

```

Figure 4-15 QMF HPO resource allocations and restrictions panel

Notice that you can even limit the SQL verbs allowed, such as the DROP TABLE statement and more. QMF HPO/Manager provides a very granular means of controlling an enterprise QMF environment. The same features and functions have been extended to the Workstation and WebSphere components of QMF, but simply presented in a different way.

HPO/Manager in action: Restricting the ability of an end user to update

Suppose that you want to limit the ability of your users to issue UPDATE statements. In the example shown in Figure 4-16, this restriction has been imposed on the user trying to issue the query shown, and HPO responds with the error shown on the message line, indicating that the UPDATE has been stopped. Such unauthorized access will be reflected in the HPO information logged and in the many reports available, as HPO provides a wealth of data that can be used to closely monitor a QMF environment.

```
SQL QUERY                                MODIFIED LINE 1
UPDATE GLOBAL.STAFF
SET SALARY = 1000000
WHERE NAME = 'BIERE'

*** END ***

1=Help      2=Run      3=End      4=Print    5=Chart    6=Draw
7=Backward  8=Forward  9=Form    10=Insert  11=Delete  12=Report
RAAG00SE--Not authorized to process this SQL verb: UPDATE
COMMAND ==>                                3 SCROLL ==> PAGE
```

Figure 4-16 With QMF HPO, users can be prevented from performing specific operations

You might have long-running queries in your organization that are prevented from running because they use too many resources. One approach to take with such queries is to compile them into more efficient COBOL programs rather than preventing users from running them. The HPO/Compiler, which we look at next, performs this and other functions.

4.5.2 Optimizing resource-intensive operations

QMF HPO/Compiler provides utilities for generating, preparing, and running report programs in the QMF Classic environment. When an object is compiled using QMF HPO/Compiler, it generates ANSI-standard COBOL that can be executed at any time. It does not require QMF to run; it is totally separate from the QMF environment. You can also modify the source code generated to add new logic or functions that were not included in the original objects.

The QMF HPO/Compiler performs these tasks:

- Provides a stored procedure development environment to create stored procedures.
- Reduces resource contention by optimizing resource-intensive queries, forms, and procedures.
- Converts dynamic SQL to static SQL, which helps reduce DB2 catalog contention and DB2 optimization overhead.

The compiled applications run faster and more efficiently, lowering your production costs.

- Converts queries, reports, and procedures into efficient COBOL programs, generating structured, stand-alone, documented source code that is easy to modify and portable to other platforms:

Using a COBOL precompiler, you can run the program on a personal workstation for use with DB2 for Linux, UNIX, and Windows or other database management systems.

The QMF HPO/Compiler contains a program generator and an end-user facility:

► Program generator:

The program generator is used by programmers to convert QMF reports (queries, forms, and procedures) into compiled programs. In addition, this facility can automatically register report programs with the end-user facility.

CICS programs are pseudo-conversational: they appear to the user as a continuous conversation, but actually consist of multiple tasks. As with TSO programs, you can generate two types of CICS programs: display programs and print programs.

► End-user facility:

The end-user facility is used to run compiled report programs. When programs are generated and prepared for running, they are automatically registered with the end-user facility. After programs are registered, the nontechnical user interface makes it easy to list, locate, and process reports. After selecting a report, you are prompted to supply any run-time variable values. If the report runs in batch mode, JCL is automatically generated and optionally submitted. Otherwise, the interactive report is run online in the appropriate environment, TSO or CICS.

The compiler full dialog mode

You can generate COBOL programs using a short dialog or a full dialog. The full dialog presents more options, but the resulting compiled COBOL program is the same. Figure 4-17 shows some of the options available with the full dialog panels.

```
DA1A / RS22DA1A -- Generate Programs (Full Dialog) ----- 10.1.0
Command ==>

Use % and _ as wildcard characters to produce a list of objects.
Press ENTER to proceed to next panel.
Type CAN or CANCEL or press PF3 to end.

Generate programs from one of the following:

1. Query Name.... ==>
2. Query Name.... ==> STAFFQ
   Form Name..... ==> STAFF_F
3. Procedure Name ==>
```

Figure 4-17 Using the full dialog option to compile COBOL in QMF HPO/Compiler

Additional compiler generation options

Additional generation options are available, such as those shown in Figure 4-18. Notice that there are additional safeguards regarding data that can be generated with QMF. In this example, the program is prevented from saving data or writing any data to a file, thus ensuring no risk of exposure.

```

DA1A / RS22DA1A -- Generation Options ----- 10.1.0
C ^^^^^^^^^^^^^^^^^ Data Processing Options ^^^^^^^^^^^^^^^^^^O
... Command ==>
Q ...
- ... 1. Save Data..... ==> N (Y=Yes, N=No) ...
G ...   a. Table..... ==>
P ...   b. Table Space... ==>
E ...   c. Action..... ==> R (R=Replace, A=Append, F=Prompt) ...
P ...   d. SQL Type..... ==> S (S=Static, D=Dynamic) ...
... 2. Erase Table(s).... ==> N (Y=Yes, N=No) ...
E ...   a. Table(s)..... ==>
/ ... 3. Read Data From File ==> N (Y=Yes, N=No) ...
/ ...   a. LRECL..... ==> *
... 4. Write Data To File. ==> N (Y=Yes, N=No) ...
/ ...   a. LRECL..... ==> *
  ^^^^^^^^^^^^^^^^^ Data Processing Options ^^^^^^^^^^^^^^^^^^O
Write Source Code To...
Data Set ==> 'TSJOEA.SOURCE'
Member.. ==> (blank=Program/Plan Name)

```

Figure 4-18 Generation options available with the QMF HPO/Compiler

Tip: One significant advantage of being able to compile QMF objects is that you have a permanent, executable piece of work that will not change if the owner goes back and changes the underlying query, form, or other objects.



Installing QMF for Workstation, QMF for WebSphere, and touring the interfaces

Part of the picture of total cost of ownership is how easy an analytics solution is to install, deploy, and maintain. In this chapter, we walk you through the installation of both QMF for Workstation and QMF for WebSphere and cover concepts relating to the interfaces. You will use these concepts as you begin to develop and deploy your customized analytics solution.

In this chapter, we consider the following topics:

- ▶ Deciding on your configuration
- ▶ System requirements
- ▶ Installing QMF for Workstation
- ▶ Installing QMF for WebSphere
- ▶ Touring the QMF for Workstation interface
- ▶ Touring the QMF for WebSphere interface

5.1 Deciding on your configuration

Configurations for QMF offer a wide range of possibilities. For QMF for Workstation and QMF for WebSphere, the fundamental choice is between a rich client interface that is full-function and local to the end user (QMF for Workstation) or a thin-client interface that will service mainly data consumers who require less content design function (QMF for WebSphere).

Administrators and rich content authors of dashboards and other objects will, most often, opt for QMF for Workstation, while casual users will typically opt for the thin-client interface. It is easy to mix and match both deployment methods in your configuration.

There are some fundamental differences between QMF for Workstation and QMF for WebSphere that can help you decide on your installation and deployment preferences. Here is a summary of the basic differences between the two configurations:

- Administration:

All administrative functions available in QMF for Workstation are also available in QMF for WebSphere.

- Enhanced visual design:

QMF for Workstation includes a visual designer for both reports and dashboards. With little or no coding experience, users can employ the visual designer to quickly develop graphical reports and dashboards. Users can dynamically place charts, selectors, and controls on a report or dashboard canvas, as well as static content such as text, graphics, hyperlinks, and supporting information (data-driven or static).

QMF for WebSphere is capable of displaying existing reports and dashboards, but cannot be used to author new visual content.

- Accessibility conformance:

QMF for Workstation conforms to accessibility standards and guidelines, including robust support for keyboard shortcuts.

QMF for WebSphere does not include this same full repertoire of keyboard shortcuts.

- Use of the host operating system scheduler:

QMF for Workstation allows users to schedule tasks using the host operating system's scheduler (Windows only), in addition to the scheduler included within QMF. It allows for additional flexibility when scheduling tasks.

QMF for WebSphere users can schedule jobs using only the job scheduler built directly into the product.

- Enhanced drill-down capabilities:

QMF for Workstation provides the ability to “zoom” a column value, displaying its content in a pop-up dialog.

The zoom capability is not included in QMF for WebSphere.

- Customizable perspectives and views:

QMF for Workstation provides the capability to customize workbench views. Users can move views around within the workbench, locating them in different areas to suit their preferences. Additionally, QMF for Workstation users can customize the workbench by adding and removing views from the various perspectives. It allows users to establish standardized views and perspectives that suit their design and data access needs.

In QMF for WebSphere, users find what they need by using the navigation tree in the left pane of the interface. Administrators customize what appears in this tree for specific sets of users or regions.

- ▶ Editing table data:

QMF for Workstation provides the capability to add, edit, and delete data within database tables using a grid-based table editor.

The table editor is not included in QMF for WebSphere.

- ▶ REXX program language support:

QMF for Workstation supports the REXX programming language, allowing power users to develop and implement scripts that address complex and repetitive tasks and functions.

QMF for WebSphere does not support the creation of REXX programs.

- ▶ Targeted printing capabilities:

QMF for Workstation supports targeted printing, allowing users to print a single document of interest. For example, users can perform a File and Print operation on a specific report or query result and get the output for that single piece of data.

In QMF for WebSphere, the capability to print is a function of whatever print capabilities are currently available to the browser.

- ▶ Seamless, procedure-run EXPORT command:

QMF for Workstation provides seamless support for procedure-run EXPORT commands. Users can run procedures to export data in various formats directly to their machines.

In QMF for WebSphere, users can also use procedures to run EXPORT commands, but the command opens the web browser's file download dialog instead.

- ▶ Manipulating data in the results grid:

QMF for Workstation users can apply conditional formatting, as well as add, remove, or edit calculated columns and apply and change column aggregation functions, by right-clicking the column headings in the results grid.

In QMF for WebSphere, these actions are performed using the query's Layout tab.

As you deploy QMF and learn more about how your end users are working with it, you can modify your configuration accordingly. The most important thing to remember is that you are not locked in to a particular interface.

5.2 System requirements

Before you install QMF for Workstation and QMF for WebSphere, refer to the system requirements shown on the QMF website to ensure that you have the recommended disk space and memory, as well as the minimum recommended versions of prerequisite software:

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

5.3 Installing QMF for Workstation

QMF for Workstation is typically installed on Windows by a database administrator, so we show a Windows installation in this walkthrough. For a complete installation, you must install the QMF for Workstation core product first and then apply the most recent fix pack.

To install QMF for Workstation, proceed as follows:

1. Load the file on the system where it will be installed.

The QMF installation uses the third-party InstallAnywhere application. Because the size of the setupwin32.exe file is approximately 400 MB, it is best practice to load the file on the system where it will be installed rather than installing from a remote file server.

Table 5-1 shows the names of the setup executables by platform.

Table 5-1 Setup executables

Platform	Name of setup executable
Windows	setupwin32.exe
Macintosh	setupMac.zip
Red Hat Linux	setupLinux.bin
SuSE Linux	setupLinux.bin

2. Run the executable file.

To install the software, run the installation executable on the target platform and simply follow the installation prompts. In this walkthrough, we will show a Windows installation, which we begin by running the setupwin32.exe file.

When you start the executable, QMF displays an installation wizard that walks you through the steps of the installation.

3. Choose the language in which you want to install QMF.

QMF for Workstation supports installation in the following languages:

- Arabic
- Brazilian Portuguese
- Czechoslovakian
- Danish
- French, Belgian French, Canadian French, and Swiss French
- German and Swiss German
- Hebrew
- Italian and Swiss Italian
- Japanese
- Korean
- Portuguese
- Spanish
- Swedish
- Traditional Chinese

This walkthrough installs English as the national language, as shown in Figure 5-1.

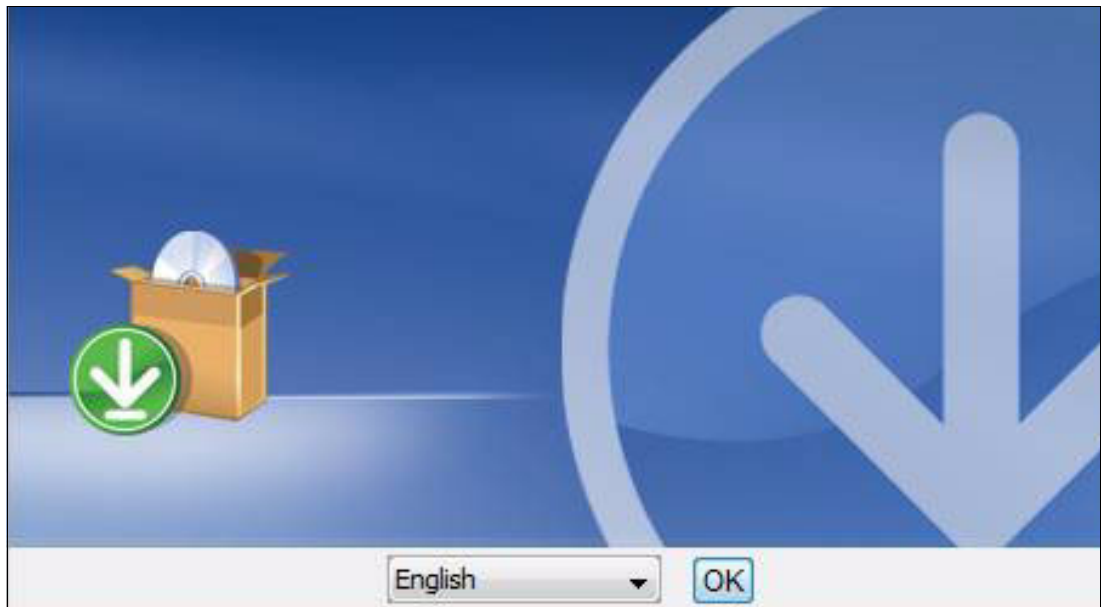


Figure 5-1 Specifying the language to be used

4. Accept the license agreement.

You see an introduction followed by the license agreement, as shown in Figure 5-2.

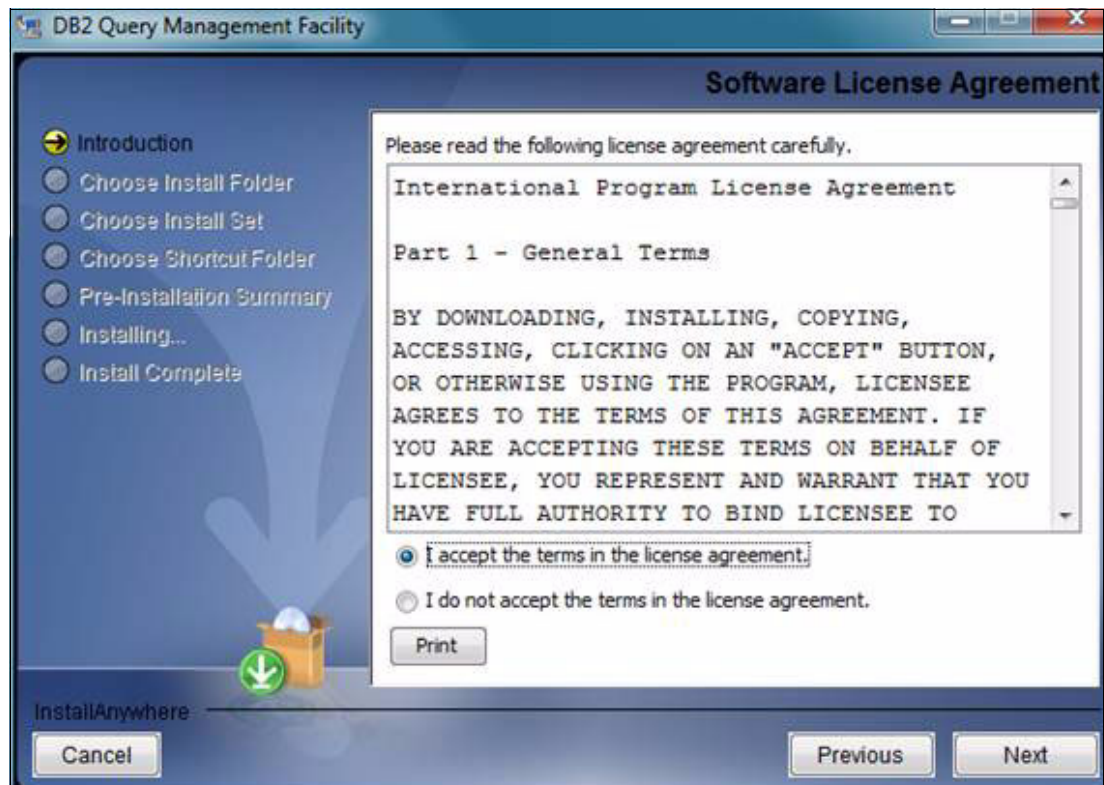


Figure 5-2 License agreement acceptance

5. Specify the location of the installation directory.

We recommend that you accept the default installation directory if you are installing on a single-user system, as shown in Figure 5-3.

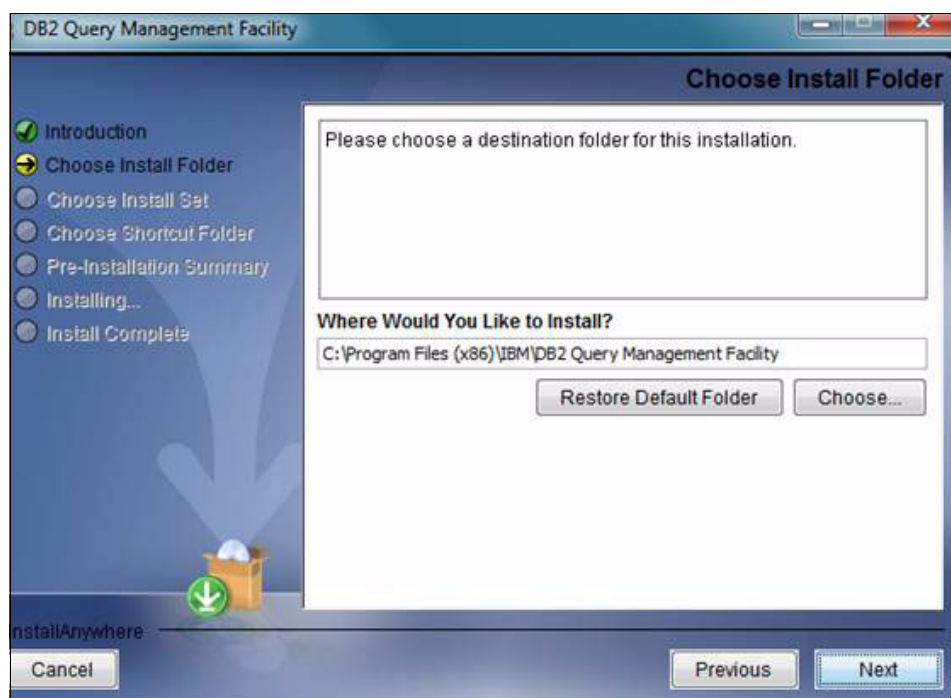


Figure 5-3 Specifying the installation directory

6. Specify which product components to install.

QMF displays a list of components, as shown in Figure 5-4. If performing an installation for an administrator or a single user, we recommend that you take all of the options.

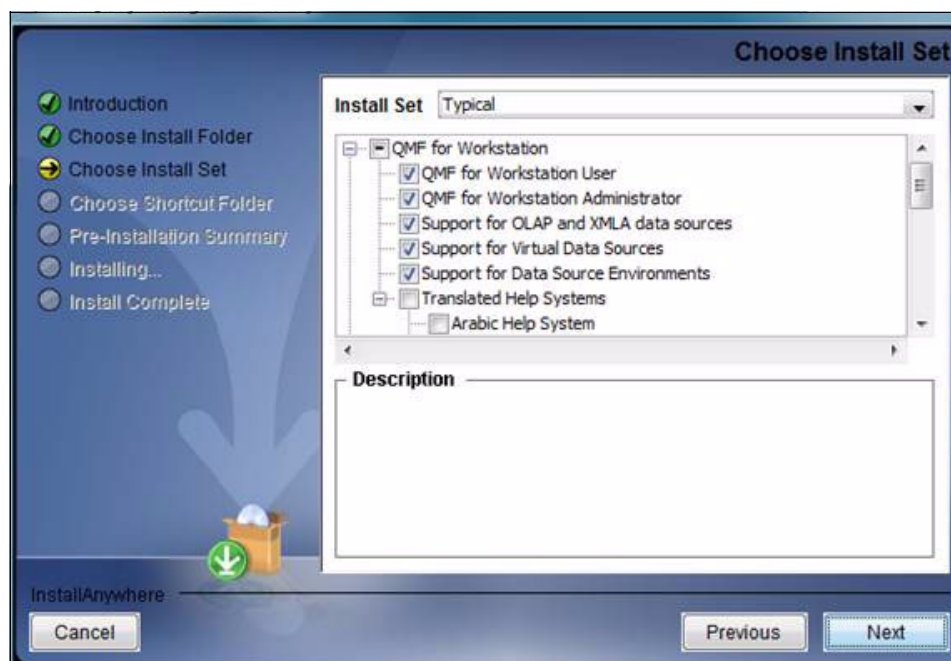


Figure 5-4 Selecting QMF components to install

Tip: A deployment for end users must omit the QMF for Workstation Administrator and the QMF for WebSphere options. When you check the QMF for WebSphere option, the installation automatically unpacks the installable web archive (.war) file that you will use to install QMF for WebSphere. Thus, the QMF for WebSphere option needs to be checked only for administrators who will be deploying it to end users, as discussed later in this chapter.

7. Specify shortcuts.

Next you can customize the Shortcut folder and icons, as shown in Figure 5-5.



Figure 5-5 Customizing the Shortcut folder and icons

8. Review the pre-installation summary and click **Next** to install the product.

As shown in Figure 5-6, you have an opportunity to review the options you have selected before you begin the installation.

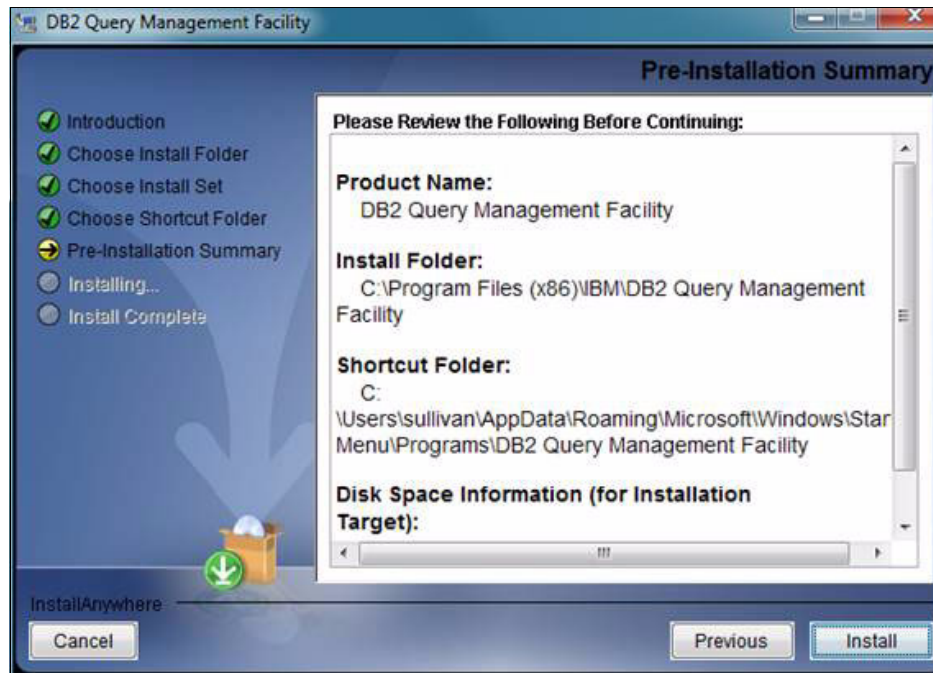


Figure 5-6 Pre-installation summary

If you click **Install**, the process will begin and will run until completion, as shown in Figure 5-7.

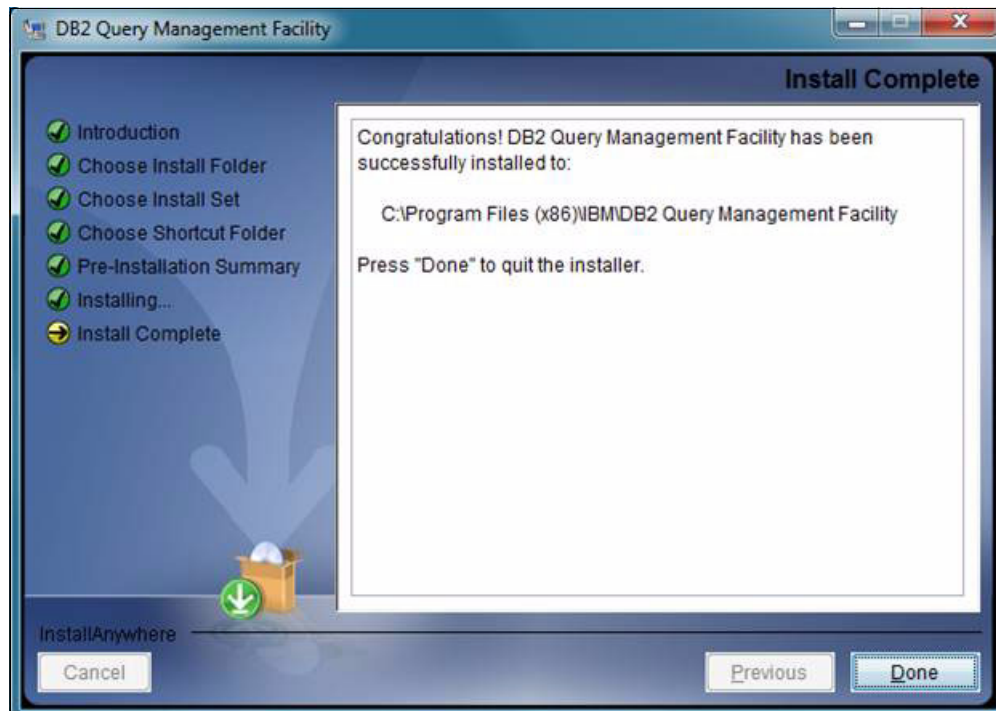


Figure 5-7 Successful installation of QMF for Workstation

5.4 Installing QMF for WebSphere

There is a huge movement toward thin-client, server-oriented architectures in business analytics today. Many companies want the majority of their end users to access QMF through a web browser to reduce the overall maintenance and complexity of setting up analytics at an enterprise level. This topic walks you through a typical QMF for WebSphere installation on Windows.

When you check the **QMF for WebSphere** option in the Choose Install Set dialog shown in Figure 5-4 on page 70, the installation process automatically unpacks the installable web archive (.war) file and enterprise archive (.ear) files for QMF for WebSphere. These files can be installed under the web application server.

The QMF for Workstation and WebSphere installation documentation, located at the following website, explains how to install QMF for WebSphere under IBM WebSphere Application Server. So, here we just cover how to install QMF for WebSphere under Apache Tomcat:

<http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.qmf10.doc.imw/imqmf10home.htm>

The .ear and .war files are unpacked to the directory you specified in the Choose Install Folder dialog, as shown in Figure 5-8.

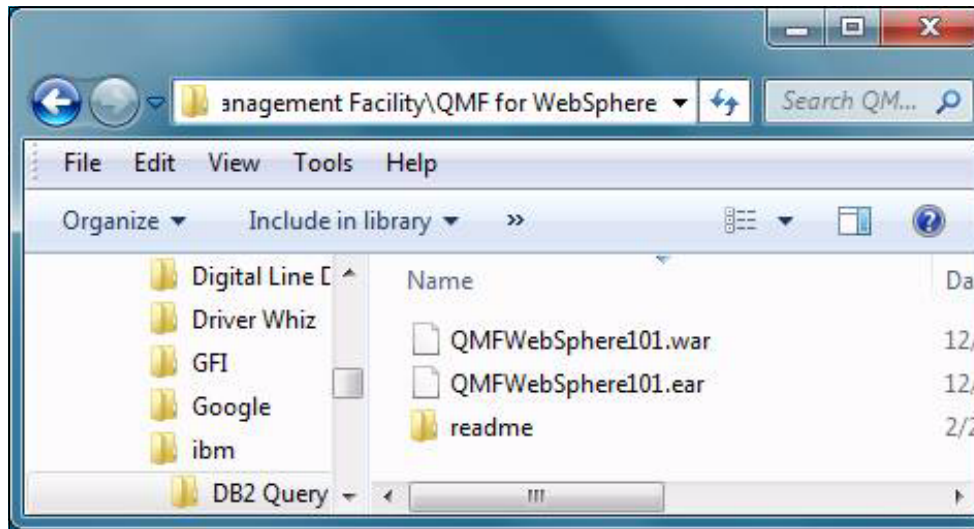


Figure 5-8 Unpacking the QMF .ear and .war files

To install QMF for WebSphere under Apache Tomcat:

1. Download and install Apache Tomcat from the Apache Tomcat Foundation website. Follow the installation wizard. We are taking the default settings to install as a service and using port 8080.
2. Shut down the Apache Tomcat service if it is running. To do so, open the Windows Control Panel for Services, shown in Figure 5-9, right-click the service name, and select **Stop**.

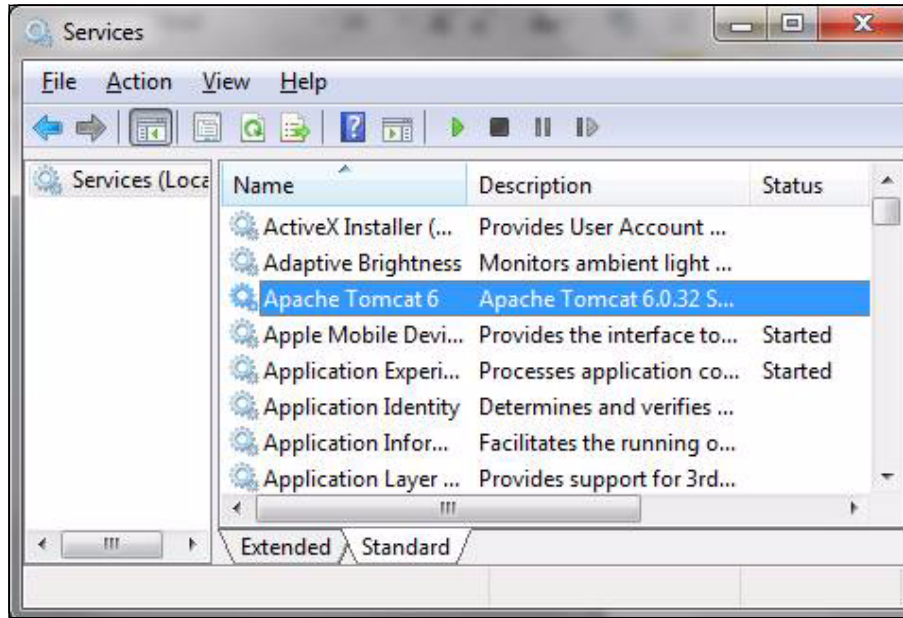


Figure 5-9 Shutting down Apache Tomcat if it is running

3. Copy the JDBC driver files for your data sources to the `../lib` directory in the Apache Tomcat folder, as shown in Figure 5-10.

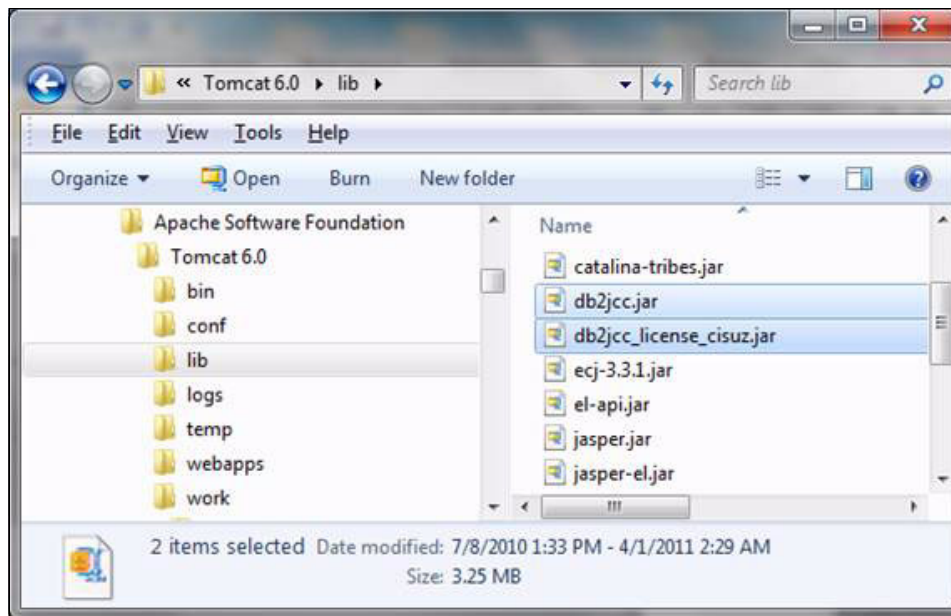


Figure 5-10 Adding JDBC drivers to the Apache Tomcat directory

4. In the ../conf directory, edit the tomcat-users.xml file, shown in Figure 5-11.

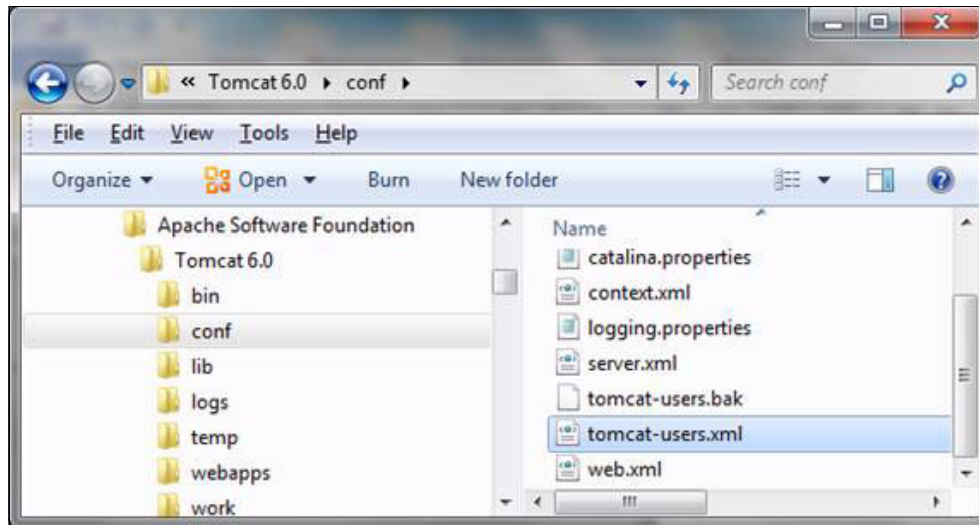


Figure 5-11 Editing the tomcat-users XML file

5. Add a rolename of "qmfadmin" and add that role to the list of roles for administrative users. For example:

```
<?xml version='1.0' encoding='cp1252'?>
<tomcat-users>
  <role rolename="qmfadmin"/>
  <role rolename="role1"/>
  <user name="admin" password="admin" roles="manager-gui,qmfadmin" />
  <user username="tomcat" password="tomcat" roles="tomcat"/>
  <user username="both" password="tomcat" roles="tomcat,role1"/>
  <user username="role1" password="tomcat" roles="role1"/>
</tomcat-users>
```

6. Copy the QMFWebSphere101.war file (not the QMFWebSphere101.ear file) to the ../webapps directory, as shown in Figure 5-12.

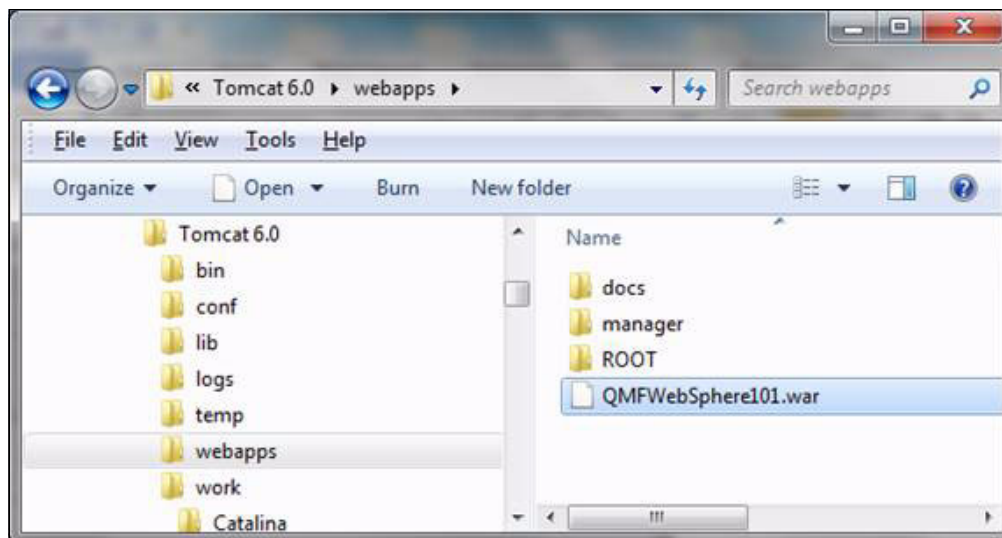


Figure 5-12 Copying the QMFWebSphere101.war file

Tip: The name of the .war file will be in the URL for QMF for WebSphere users. Thus, the default URL will be as follows:

<http://<hostaddress>:8080/QMFWebSphere101/user>

If you rename the .war file at this point, you can control the URL name.

7. Start the Tomcat service. Even after it is started, it might take a few minutes to internally deploy QMF for WebSphere.
8. After a few minutes, go to the URL (or its equivalent if you altered the URL as explained in the foregoing tip):

<http://<hostaddress>:8080/QMFWebSphere101/user>

The QMF for WebSphere user interface is displayed, as shown in Figure 5-13.

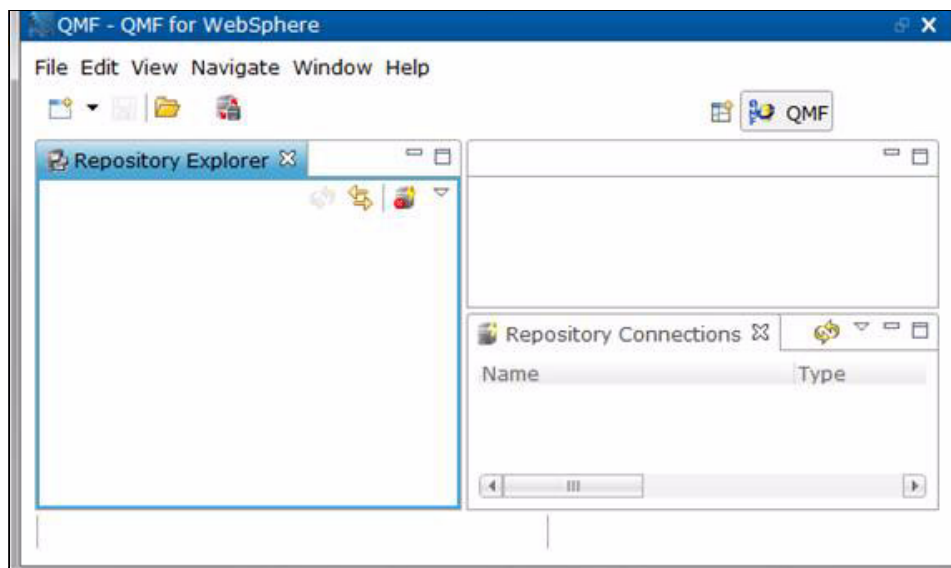


Figure 5-13 Starting QMF for WebSphere on your system

5.5 Touring the QMF for Workstation interface

Now that you have installed QMF for Workstation and WebSphere, we take a moment to familiarize you with some of the features and options of the interface so that you can navigate immediately to what you need.

5.5.1 Eclipse-based architecture

The majority of programming, content development, and data access applications in the market are now built on the Eclipse platform. It not only provides interface consistency across vendor tools but also allows disparate tools to work together on the workstation. For example, users might elect to run a third-party Eclipse-based data schema editor directly within the QMF for Workstation application user interface.

The same Eclipse-based interface is used in both QMF for Workstation and QMF for WebSphere, allowing users to seamlessly transition between the workstation and web-based products. Because the product is based on Eclipse, the Eclipse-based concepts of views and perspectives are used in the interface, as explained in the following topics.

5.5.2 Views

After installation, QMF for Workstation presents the user interface shown in Figure 5-14.

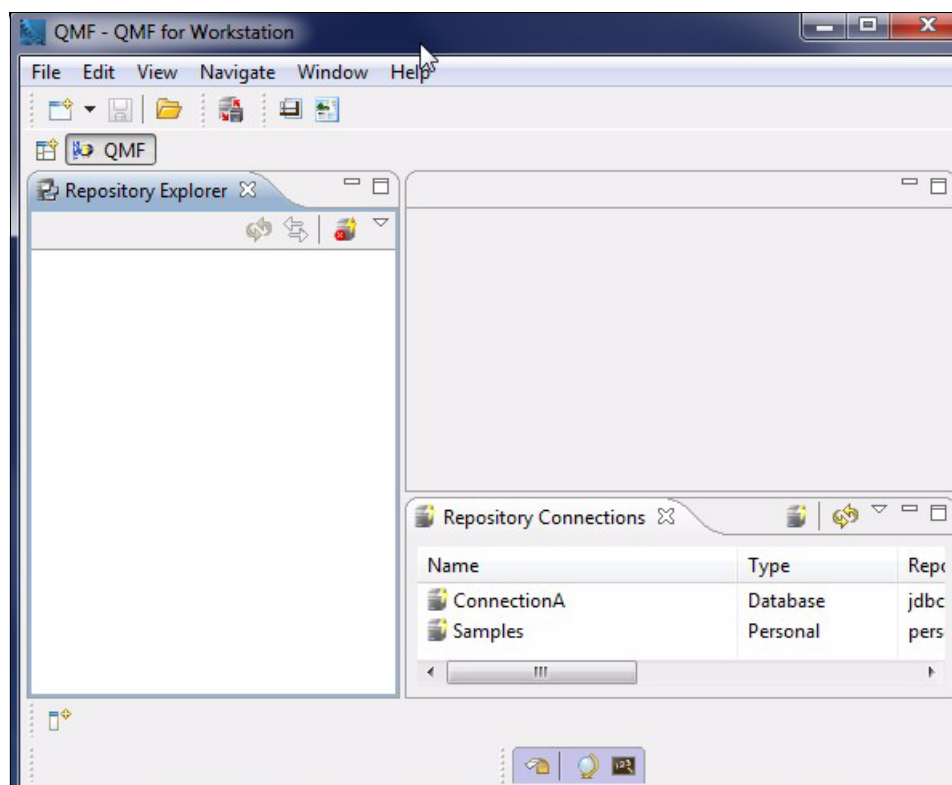


Figure 5-14 The QMF for Workstation interface after installation

The interface includes numerous tabbed panes that can be detached and relocated (“docked”) around the main content area. Collectively, these tabbed panes are called views. On the Eclipse platform, views are used to navigate a hierarchy of information, open an editor, or display properties for the active editor. Views in QMF operate in the same manner.

The Repository Explorer view, shown at the upper left of Figure 5-14, provides access to data sources and saved queries, while the Repository Connections view, shown at the bottom of Figure 5-14, gives the user a summary of the repositories that are known to QMF.

The number and nature of views visible depends on the task you are performing. For example, dashboard development requires visual design views that are not required when building ad-hoc queries and reports. QMF for Workstation therefore includes a number of perspectives, each of which provides different views depending on the purpose of the perspective.

5.5.3 Perspectives

Different menus, buttons, and views tend to be used by different types of QMF users, so QMF provides perspectives that are tailored for different user roles. A perspective is a collection of views appropriate for the activities typically performed by an individual within one of these roles. For example, the Administrator perspective is shown in Figure 5-15. This perspective provides views and menus commonly needed by QMF administrators.

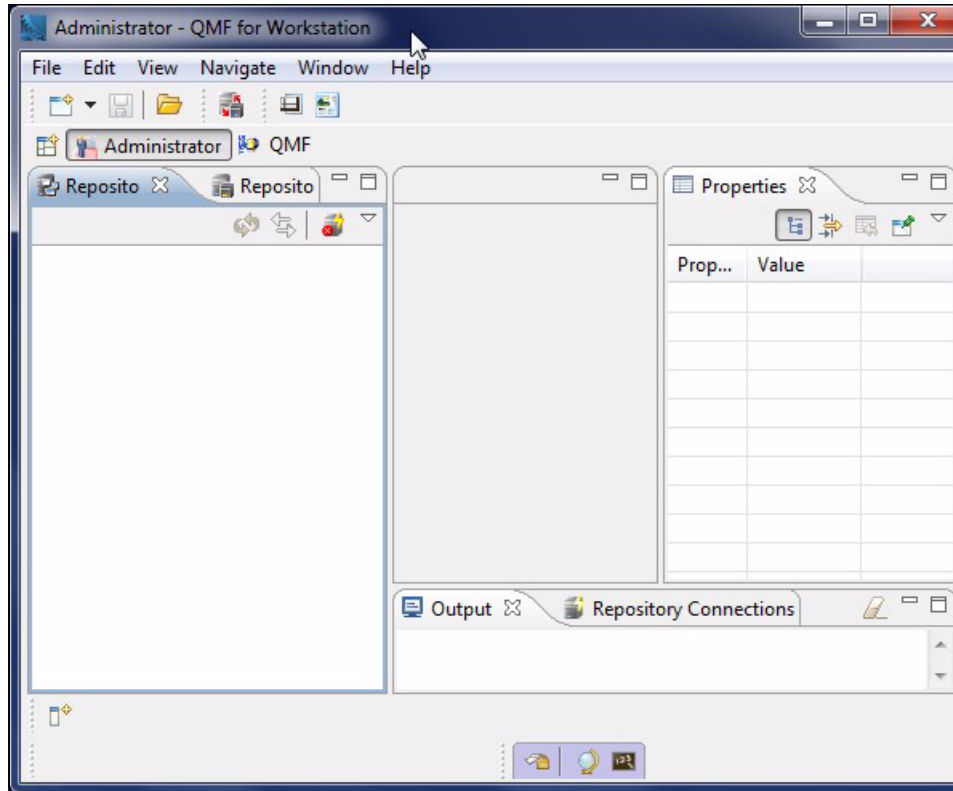


Figure 5-15 The Administrator perspective in QMF for Workstation

The User perspective looks more like what you see in Figure 5-16.

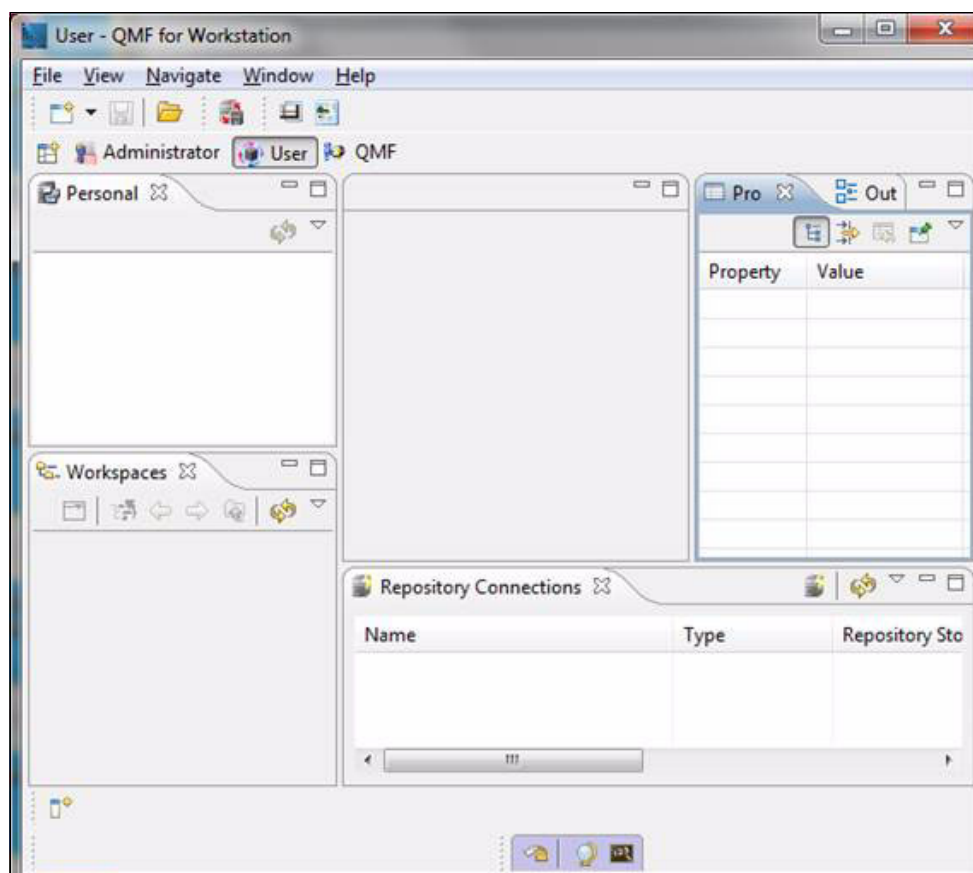


Figure 5-16 The User perspective in QMF for Workstation

QMF for Workstation and WebSphere include the following perspectives:

- ▶ Administrator:
Displays the commonly used views appropriate to administering the QMF for Workstation/WebSphere products, such as adding repositories and data sources and managing user and group permissions.
- ▶ QMF (default):
Displays a small collection of views to provide a user interface that resembles the QMF for Windows Version 8 environment.
- ▶ QMF Classic:
Displays a collection of views that provide a familiar and productive environment for users who have historically used QMF for TSO and CICS.
- ▶ User:
Displays a number of views appropriate for end users who are creating and running queries, procedures, reports, and dashboards.
- ▶ Visual Designer:
Arranges the user interface in a form suitable for dashboard and graphical report authoring.

The application starts in the QMF perspective when first installed. Users can switch between perspectives by clicking the **Open Perspective** icon highlighted in Figure 5-17.

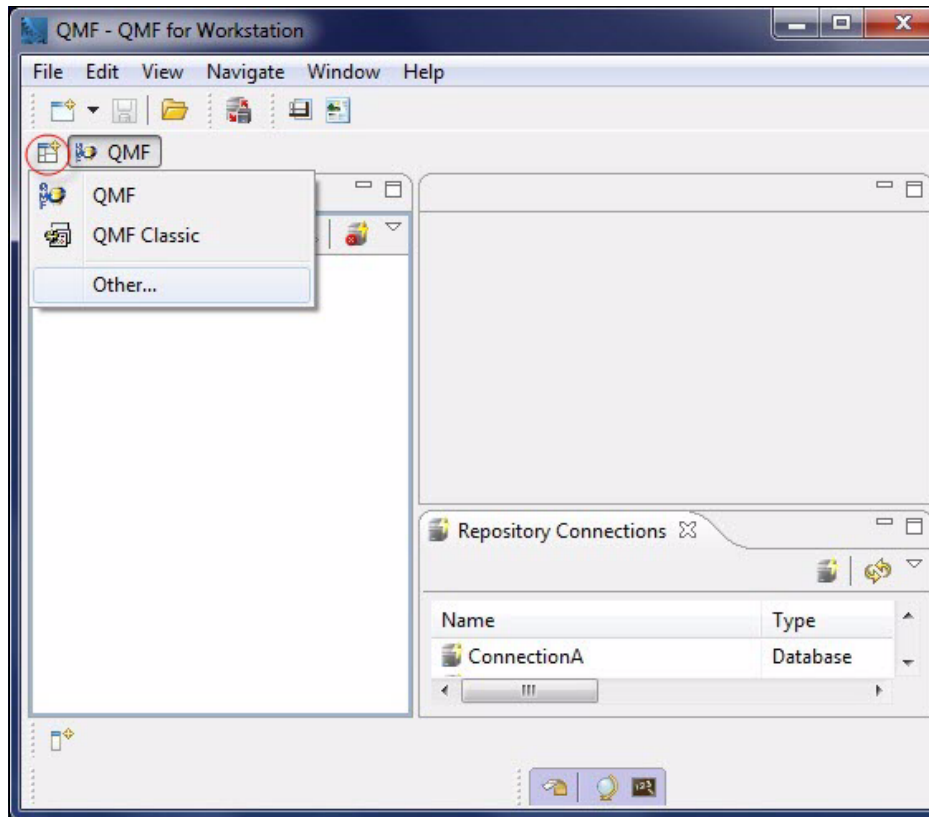


Figure 5-17 Opening a new QMF perspective

To see the many perspectives, click the **Other** option to display the Open Perspective window, which is shown in Figure 5-18.

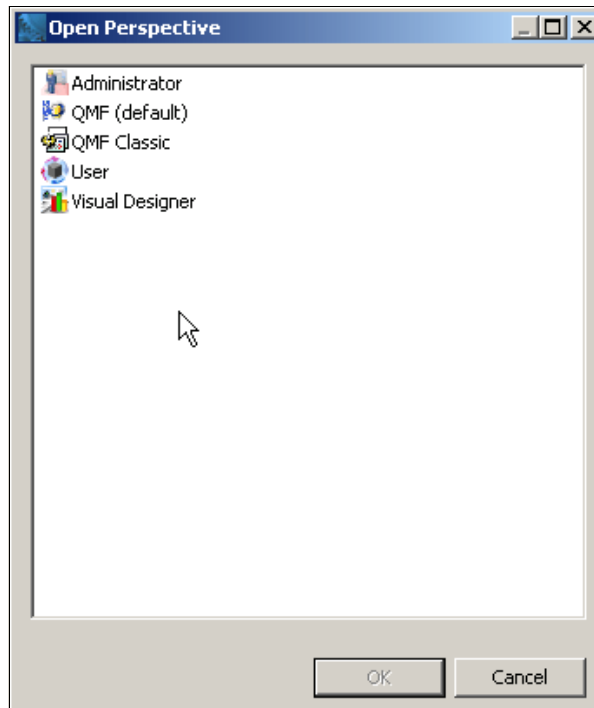


Figure 5-18 Open Perspective window

Other perspectives can be accessed by selecting an item in the dialog and clicking **OK**. After it has been opened, each perspective is listed thereafter in the toolbar (as shown in Figure 5-19), making it easier to navigate among the different perspectives while you work.

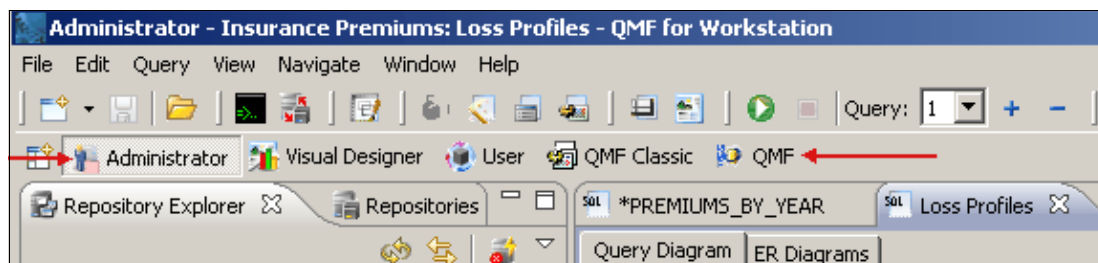


Figure 5-19 QMF perspectives toolbar, showing all perspectives accessed by the user

Perspectives can be customized with your own choice of views and layout. For example, you might want to see a list of database connections, along with the REXX console, when developing and running queries. It can be done as follows:

1. Select the **User** perspective by clicking the **Open Perspective** toolbar icon (Figure 5-17 on page 80) or by choosing **Window** → **Open Perspective** → **User**.

2. Choose **Window** → **Show View** → **Other**, as shown in Figure 5-20.

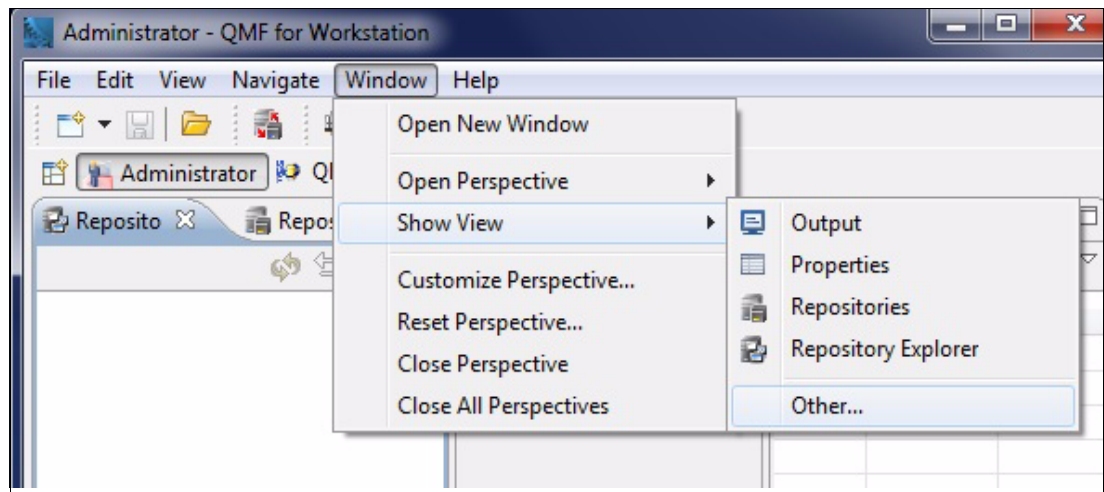


Figure 5-20 Accessing the full list of available views

3. The Show View window, shown in Figure 5-21, presents a list of all available views that can be displayed. Views are organized into four categories, and the text box at the top of the dialog allows users to locate a given view by typing part of the name. For example, you can type REXX to locate the REXX Console view.

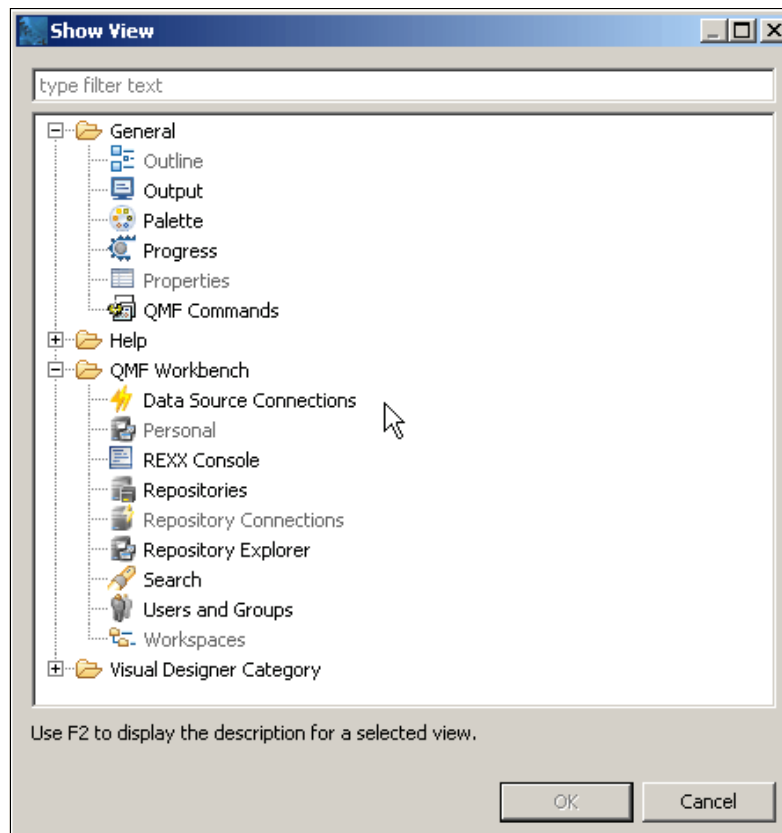


Figure 5-21 The Show View window lists all available views

4. Choose Data Source Connections from the list and click **OK** to display the view. Repeat this process to display the REXX Console view.

The two new views are added to your User perspective. You can then drag and drop these new views (or any of the existing views) into other areas around the application content area. In addition, default views can be closed in a given perspective, allowing for full customization of the user interface. Figure 5-22 shows an example of the user perspective with the following changes applied:

- ▶ REXX Console, Data source connections, and Output views have been added and docked to specific locations by dragging and dropping them. The Output view traces all SQL executed, as well as status and error messages.
- ▶ The default Properties view has been closed.
- ▶ The default Personal view has been dragged into the same docked area as the Workspaces view, creating more space at the left of the main content area.

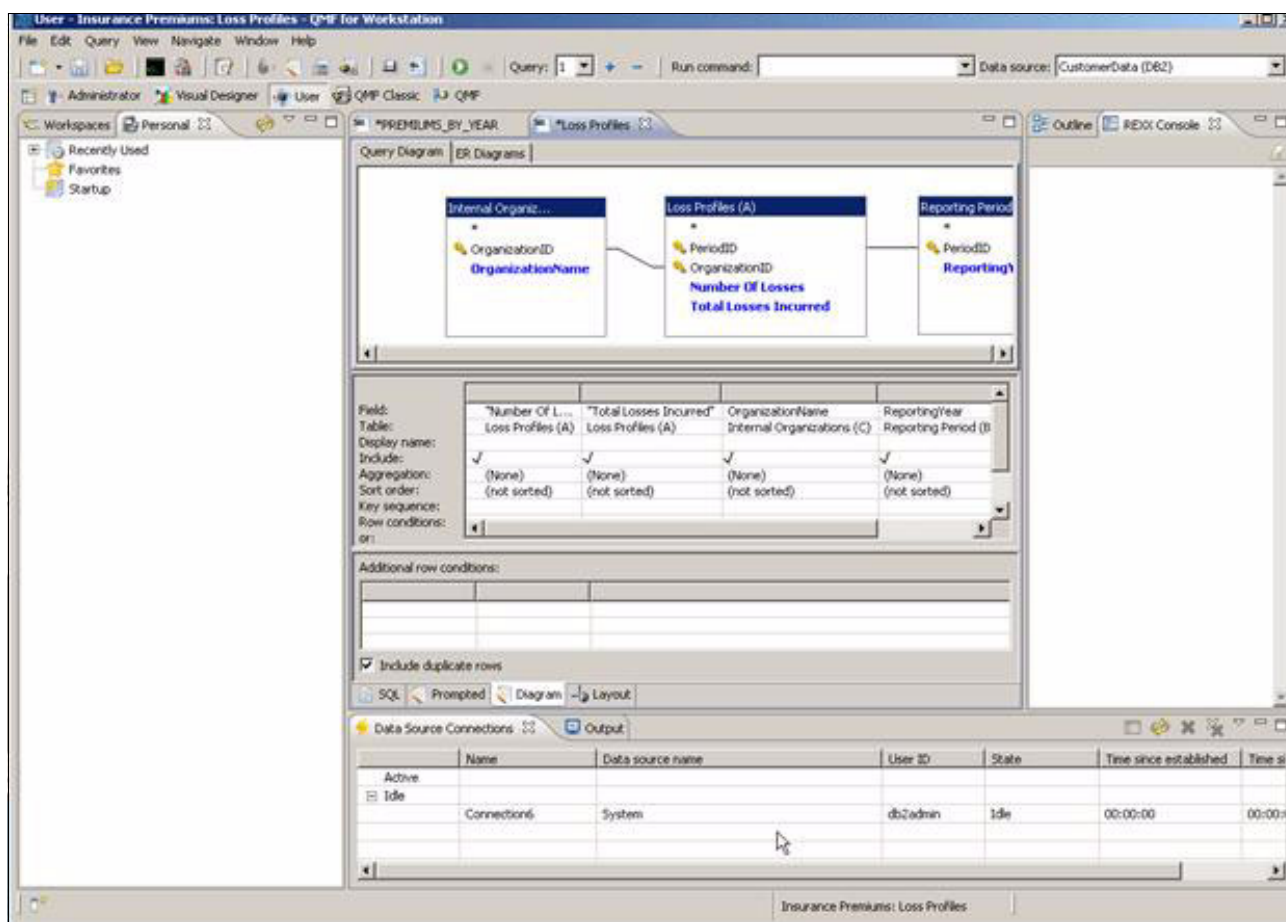


Figure 5-22 Customized QMF for Workstation user interface layout

Whenever you run QMF for Workstation and select the User perspective, your user interface layout will return to your customized layout.

Tip: Think of the five perspectives as five different user interface layouts you can start with, customize if you choose, then restore at the click of a button if needed. A customized perspective can be reset to “factory defaults” by choosing **Window → Reset Perspective**. It will return the perspective to the original layout, removing all modifications that you subsequently applied.

5.5.4 Shared repository storage

The most common deployment of QMF assumes that end users will want to share work and collaborate. QMF objects to be shared must reside in a shared repository. Shared repository storage is a set of database tables that stores objects and information grouped by qualifying names called repositories. Each item held in the tables has a field designating the repository name associated with the content.

5.5.5 Repositories

A *repository* is a logical grouping of QMF objects (queries, procedures, forms, reports, and dashboards) developed according to business needs for a specific set of data sources. The repository stores the connection information for the data sources.

There are two types of repositories:

- Shared repositories:

A shared repository is designed to support multiple users, with a single user as the administrator. Shared repositories can reside in any JDBC-compliant database.

Shared repositories store the connection information for the data sources associated with the repository's objects. (If you are familiar with older versions of QMF, the shared repository concept replaces the Server Definition File as well as a set of rules setting limits and permissions for users at each data source.)

- Personal repositories:

A personal repository serves a single user. QMF for Workstation installs with an internal Derby (open source) database engine. Derby is an embedded Java database for applications to store application data, allowing QMF for Workstation end users to create repository storage on their local machines. Personal repositories can be set up with the connection information for remote databases as well.

Personal repositories are accessed from the personal settings directory of the machine that is running QMF. For example, in Windows, the personal repository is saved under the following path (shown here on two lines):

```
C:\Documents and Settings\[UserName]\Application Data\IBM\QMF for  
Workstation\Personal Repositories
```

Sample data that is shipped with QMF is stored by default within the personal repository when it is installed, providing a way for users to learn QMF without affecting production data. Personal repositories are well suited for end-user training or for database administrators doing proof of concept. However, they are typically limited to these types of functions, as the traditional methods of securing data do not apply.

The Administrator perspective provides the views, menus, and wizards that enable you to create, manage, and maintain repositories. Additional granularity is available within a repository, allowing you to create subgroups of content that you can then choose to present to end users as workspaces.

Tip: A typical site needs only one shared repository storage space and one repository within it for production use. You might want to set up an additional repository for development purposes. A repository will likely have many workspaces. One programmatic option allows the automatic generation of a home workspace for each user. You can learn more about these workspaces in Chapter 6, “Configuring access to data sources and populating user workspaces” on page 89 and Chapter 8, “Configuring security” on page 145.

5.5.6 Workspaces

QMF administrators create workspaces to logically group content from a repository so that it can be presented to end users as unified group of resources. The workspace is the means by which repository content can be viewed and used by end users. Thus, only content in a repository that is assigned by an administrator to a workspace will be available to end users in the User perspective. Figure 5-23 shows a workspace named TWSHAW2 that has been populated with content.

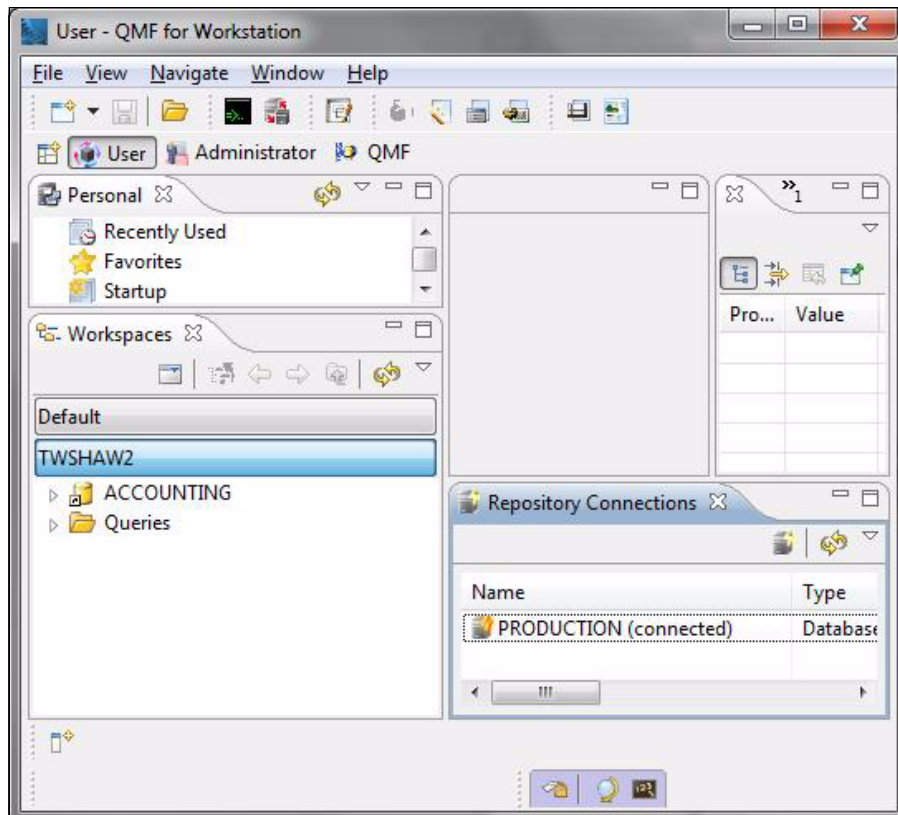


Figure 5-23 A workspace, defined and available

5.5.7 Repository connections

A repository connection is a view item that bundles the connection information for a particular shared repository storage space with the name of a particular repository stored within it.

The repository connection is the end user's point of entry into the QMF infrastructure. The repository connection layer shields the end user from having to specify a number of steps to access a particular data source or collection of QMF objects:

- ▶ When an administrator creates a repository connection, the connection combines selection of the shared repository storage and the selection of a repository into a single tree item.
- ▶ When the end user selects the repository connection, a connection is made to the appropriate database containing the tables that make up the shared repository storage. The rows containing table names within a given repository are presented to the end user. They are grouped into the workspaces to which they have been assigned.

Thus, when starting a work session, a user will first make a connection to the database holding the shared repository storage and will have the option of connecting to the data source whose connection information has now been retrieved.

5.6 Touring the QMF for WebSphere interface

QMF for Workstation provides users with a modern, productive user interface within which queries, reports, and dashboards can be created, edited, or executed. Frequent users will benefit from the convenience of a locally-installed desktop application. However, occasional QMF users might prefer to access QMF by a web browser. Accessing QMF in this way precludes the need for a locally-installed application and provides a means to access powerful analytics capabilities through a thin-client architecture.

QMF for WebSphere has been specifically designed with two objectives in mind:

- ▶ Serve as a desktop replacement offering for users that access QMF infrequently.
- ▶ Serve as a runtime environment for dashboards, reports, and queries authored in QMF and distributed across the enterprise.

QMF for WebSphere users typically want to view the content (such as a dashboard or report) with no accompanying QMF user interface elements. To address this requirement, QMF for WebSphere has two user interface modes: it can be accessed as both a web-based application and as a content server, whereby a specific dashboard or report is returned in the web browser.

Nearly all aspects of the workstation interface are present in QMF for WebSphere, with the exception of the visual designer. By way of comparison, the QMF for Workstation user interface is shown here in Figure 5-24. Connecting to the very same repository content, the QMF for WebSphere user interface is shown next in Figure 5-25.

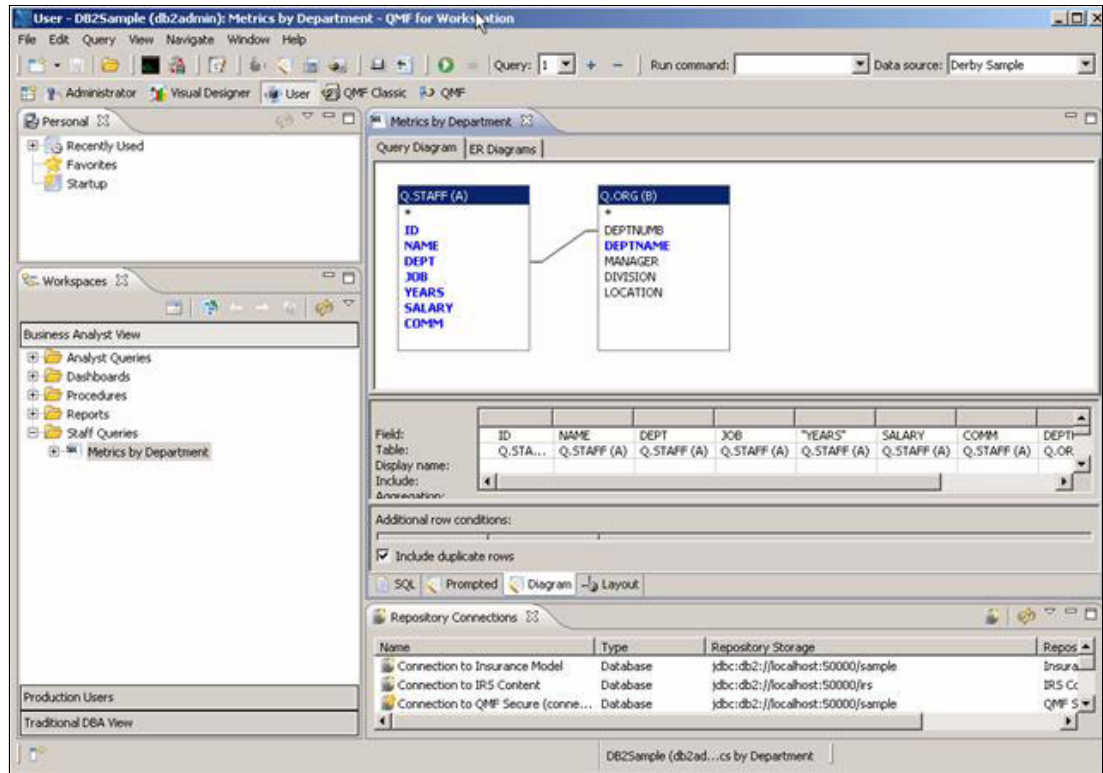


Figure 5-24 The QMF for Workstation interface

As shown in Figure 5-25, the QMF for WebSphere user interface is truly a desktop application within a web browser. The interface is pure HTML and JavaScript and requires no client-side plug-ins. Because both QMF for WebSphere and QMF for Workstation connect to the same repositories, content created in either application will immediately appear across all applications, desktop or web-based.

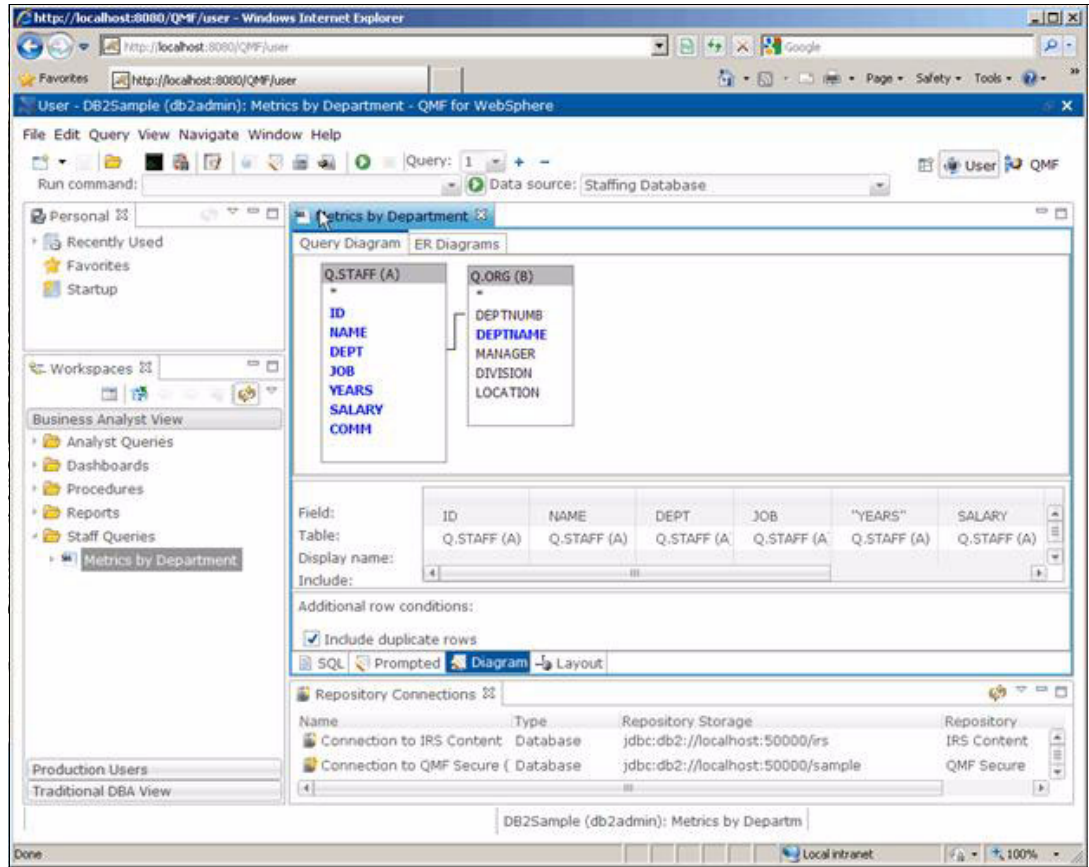


Figure 5-25 The QMF for WebSphere interface



Configuring access to data sources and populating user workspaces

Before you can add data sources that users can work with (whether they are relational, multidimensional, hierarchical, or semi-structured), you must first create shared repository storage. As explained in Chapter 5, “Installing QMF for Workstation, QMF for WebSphere, and touring the interfaces” on page 65, shared repository storage holds the repositories that contain data and objects that end users work with in their workspaces. Typical installations include a single shared repository storage space and one repository within it.

This chapter shows you how to do the following tasks:

1. Create shared repository storage. At the time shared repository storage is created, application control tables and structures required to operate QMF are installed.
2. Create a repository to hold data and objects.
3. Configure access to data sources.
4. Populate the Repository Explorer tree in the interface by defining connections to data sources.
5. Populate users’ workspaces with content from the connected repository.

As you move through this process, you will make some decisions about your configuration. These decisions are recorded in configuration files that are stored in the operating system, which is Windows in the case of the example installation shown in Chapter 5, “Installing QMF for Workstation, QMF for WebSphere, and touring the interfaces” on page 65. When the initial configuration is finished, the configuration files are collected and placed in the same folder as the setupwin32.exe file.

6.1 Creating shared repository storage

As explained in Chapter 5, “Installing QMF for Workstation, QMF for WebSphere, and touring the interfaces” on page 65, shared repository storage provides a way to organize QMF objects and data into repositories. Repositories can be further divided into workspaces, which allow you to group content from a repository in ways that are meaningful to different sets of end users. So the first step in making content accessible to end-user workspaces is to create shared repository storage.

In addition to holding repositories that contain data users will work with, shared repository storage also contains:

- ▶ The QMF catalog, in which objects that need to be shared among products in the QMF family are stored
- ▶ Application control information accessed and used by QMF for Workstation, including connection information for defined data sources

To create shared repository storage, here is what you need to do:

1. Establish connectivity to the data source that will host the shared repository storage.

Because application control structures and catalogs are established upon creation of shared repository storage, the hosting database must be able to accommodate the SQL statements that create this infrastructure. These SQL statements will be shown in a dialog when you get to Step 8 on page 101 in 6.1.2, “Creating and configuring shared repository storage” on page 96. The examples in this topic will show you how to establish a direct IBM DRDA® connection to a DB2 for z/OS database and create shared repository storage in this database.

2. Create and configure the shared repository storage.

The following topics examine each of these steps in detail.

6.1.1 Defining connectivity to the database that will host the shared repository storage

There are two ways to establish connectivity to a data source: JDBC or direct DRDA. For DB2 for z/OS, you can use either method. For all data sources other than DB2 for z/OS, you must use JDBC to make the connection. Both methods are explained next.

Configuring a JDBC connection to a data source

QMF for Workstation is a Java application using JDBC as the most common means to connect to data sources. Before any end-user specific configuration can begin, you must load the appropriate JDBC driver on the desktop. You then must point QMF to the location of the driver. You do this by entering the location of the driver file and the name of the Java class inside that file, as specified by the driver vendor.

For example, for DB2 for z/OS, the following parameters apply:

- ▶ The JDBC driver to be used is the DB2 Type 4 UDB driver.
- ▶ The file name is db2jcc.jar.
- ▶ The class name is com.ibm.db2.jcc.DB2Driver.
- ▶ The configuration requires a license file, which is called db2jcc_license_cisuz.jar.
- ▶ An auxiliary driver is also required, and that driver is called sqlj.zip.

Tip: The DB2 JDBC driver files just mentioned are commonly acquired through an IBM DB2 Connect™ or DB2 LUW installation from the `.../SQLLIB/java` directory.

The class names for commonly used drivers for a number of other databases (such as Oracle, SQL Server, and Informix) are preconfigured within QMF, but you must supply the driver and the location of the driver and any required license files. In later topics, we explain how you make these drivers available to end users.

To configure a JDBC connection to a data source, follow these steps:

1. Start QMF for Workstation. If it is the first time that you have started the product, a Welcome window is displayed. The Welcome window is a valuable quick reference that you can return to in the future by selecting **Help** → **Welcome** from the application menu. Click the curved arrow in the upper right corner of the Welcome window, as shown in Figure 6-1, to navigate to the user interface.

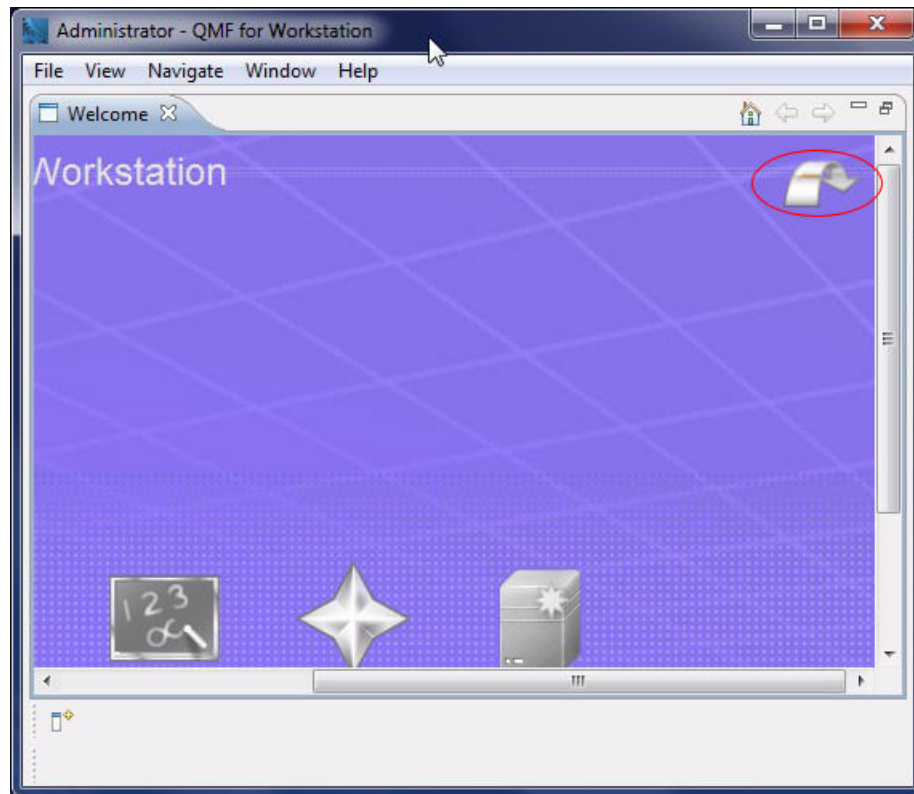


Figure 6-1 The QMF for Workstation Welcome window

2. When the user interface is displayed, as shown in Figure 6-2, the default perspective shown is called the QMF perspective. (See Chapter 5, “Installing QMF for Workstation, QMF for WebSphere, and touring the interfaces” on page 65 for an explanation of this and other perspectives.) Switch to the Administrator perspective by clicking the Open Perspective icon to the left of the QMF perspective tab and selecting **Other** → **Administrator**.

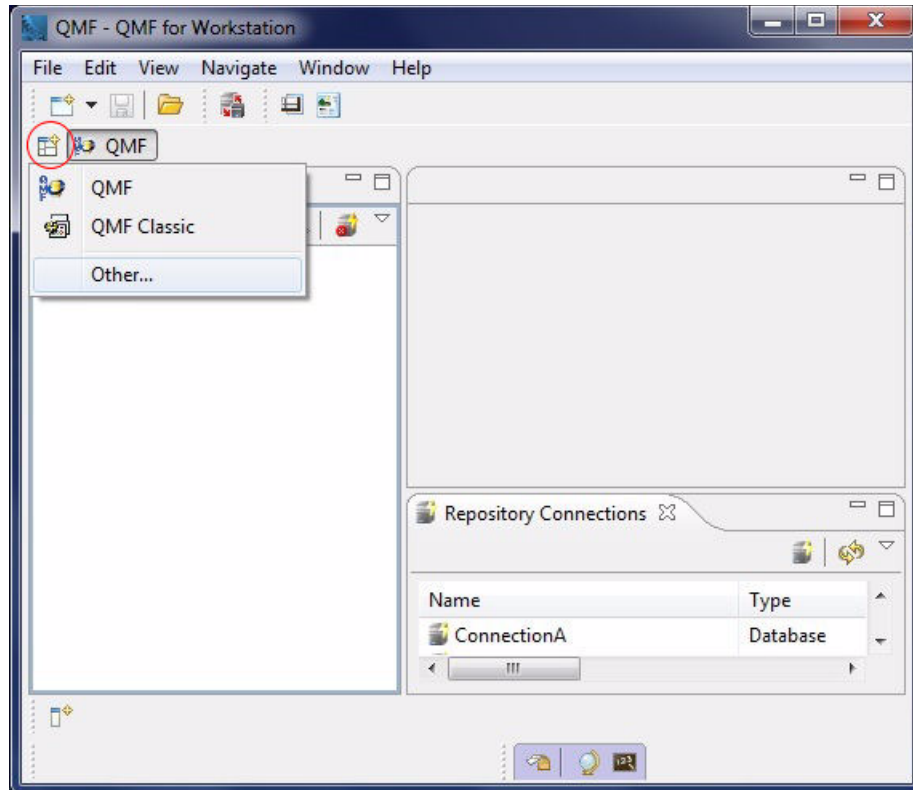


Figure 6-2 The Open Perspective icon allows you to access the various perspectives available

3. Select the tab for the Repositories view, shown in the left pane in Figure 6-3.

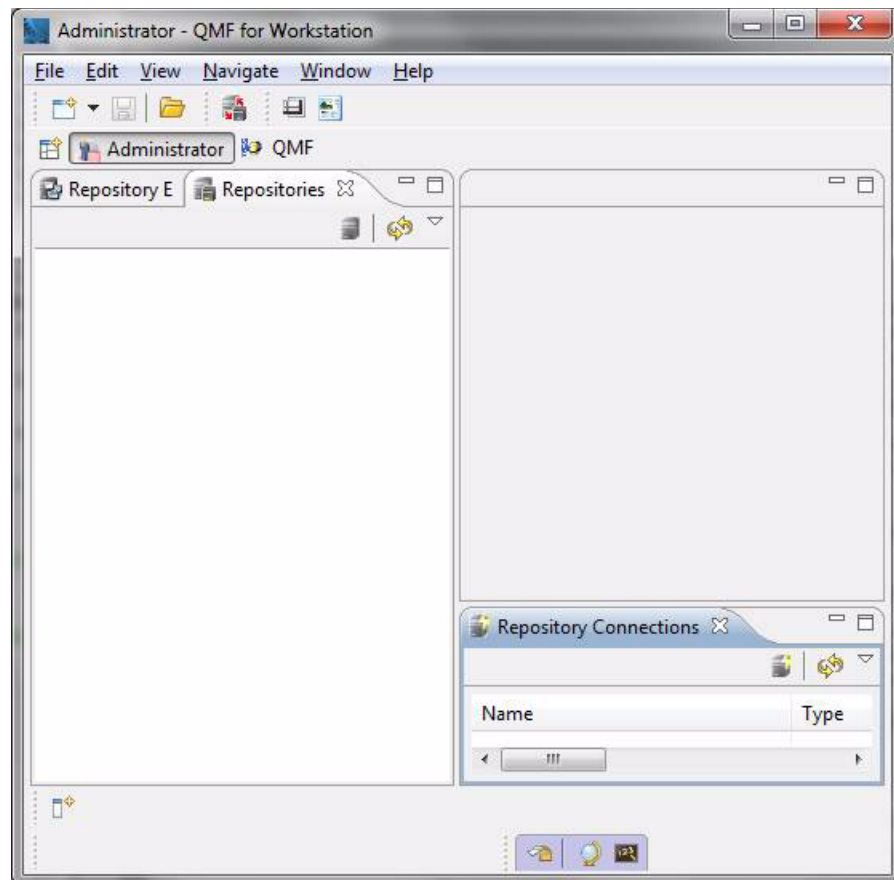


Figure 6-3 Displaying the Repositories view

4. Select **View** → **Preferences**.

- From the list on the left, select JDBC Libraries, as shown in Figure 6-4. Some commonly used drivers are listed on the right. The vendor-specific class names are included for each driver name, but the actual drivers and their locations are not included with QMF for Workstation. The exceptions to this are the Sun ODBC and Derby drivers, which are embedded in the product.

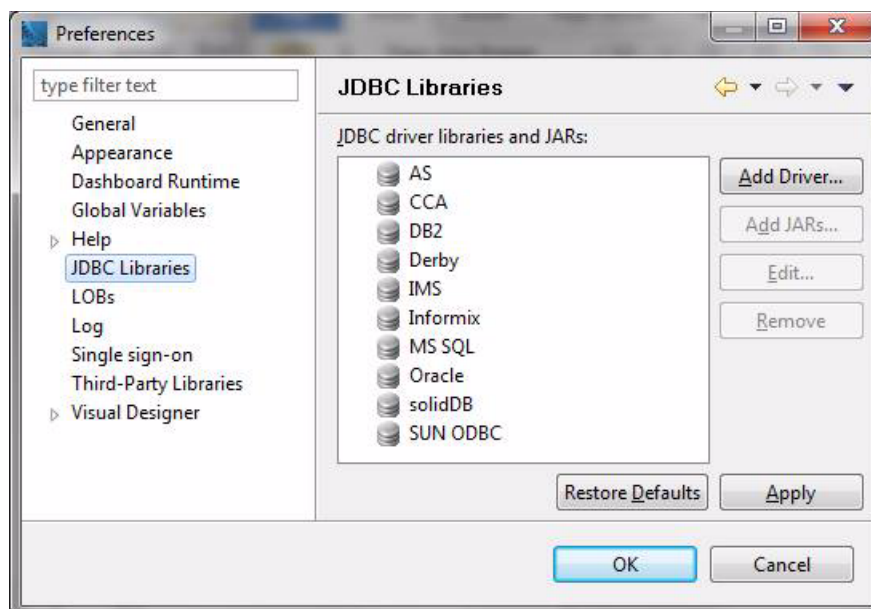


Figure 6-4 Selecting JDBC libraries

- Expand the tree for DB2 and expand the branch for DB2 Type 4 UDB driver, as shown in Figure 6-5. As mentioned in the prior step, the class name is listed, but you will need to supply the drivers and license files. You will point to the location of these in a later step.

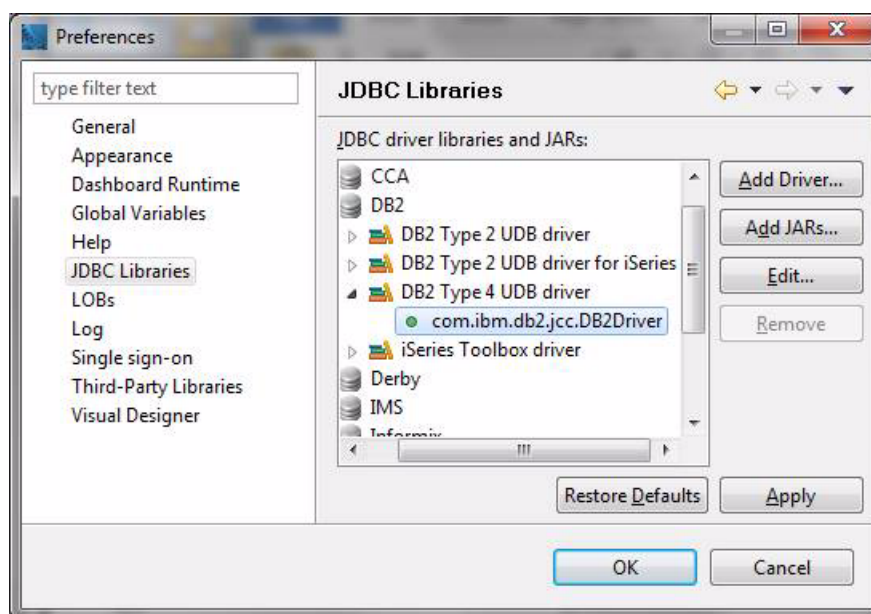


Figure 6-5 Specifying the DB2 Type 4 UDB driver

- Click the class name **com.ibm.db2.jcc.DB2Driver**, as shown in Figure 6-5.
- Click **Add Jars**. The jars referred to here are the driver and license files.

9. Navigate to the location of the drivers and license files. For DB2 for z/OS, you need three files, db2jcc.jar, sqllib.jar, and db2jcc_license_cisuz.jar, as shown in Navigating to the driver files. To select all three, hold down the Ctrl key while selecting, then click **Open**. See Figure 6-6.

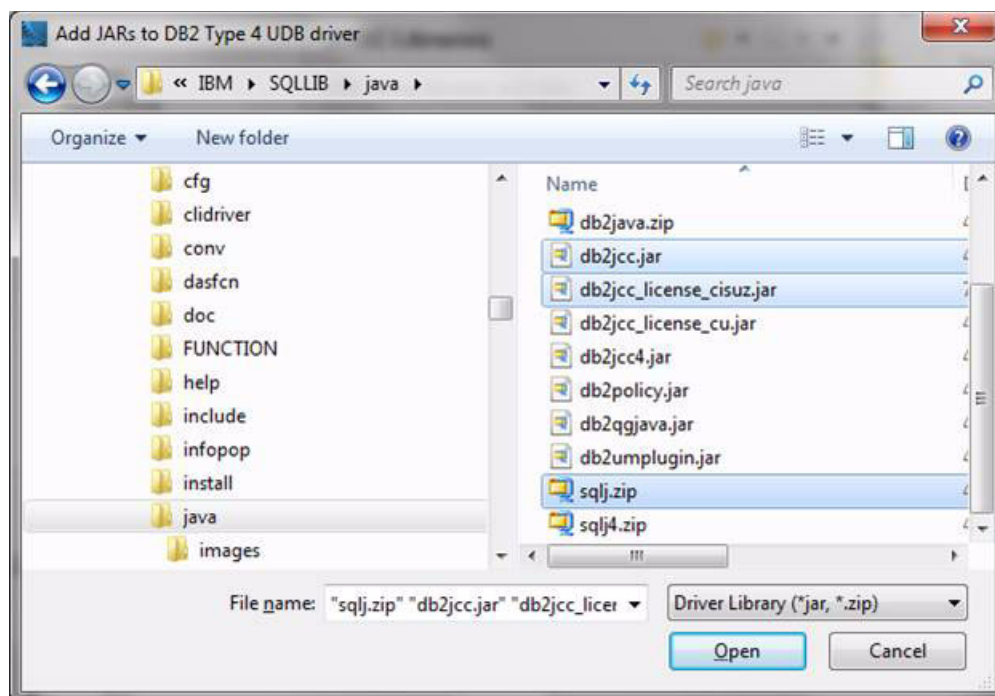


Figure 6-6 Navigating to the driver files

Tip: The db2jcc_license_cu.jar license file will not allow access to DB2 for z/OS.

10. When you return to the Preferences window, click **OK**.

Configuring a direct DRDA connection to a data source

QMF for Workstation provides an option to make a direct DRDA connection to DB2 for z/OS. This connection is built into the product and requires no additional drivers or external files, nor does it require DB2 Connect.

Because it is built into the product, you can configure a DRDA connection at the time that you create shared repository storage, as shown in the next topic. Although configuring a direct DRDA connection is similar to the JDBC driver setup explained in the previous topic, notice the following key differences:

- ▶ Direct DRDA is only supported for DB2 for z/OS, so all other data sources must use the JDBC driver setup.
- ▶ The direct DRDA driver uses a different JDBC URL that can be easily confused with the URL for configuration by JDBC, so care must be taken to configure it properly, as shown in the next topic.
- ▶ The direct DRDA connection requires an additional “bind packages” step to set up the supporting infrastructure for DB2 for z/OS.

6.1.2 Creating and configuring shared repository storage

As mentioned in Chapter 5, “Installing QMF for Workstation, QMF for WebSphere, and touring the interfaces” on page 65, shared repository storage is a set of database tables that stores QMF for Workstation objects, such as queries and dashboards, as well as database connection information, QMF configuration information, and application data. Configuring shared repository storage provides a way for QMF users to share objects.

To set up shared repository storage, follow these steps:

1. Select **File** → **New** → **Shared Repository Storage**. You can also right-click anywhere in the white space of the **Repositories** view to make this selection, as shown in Figure 6-7.

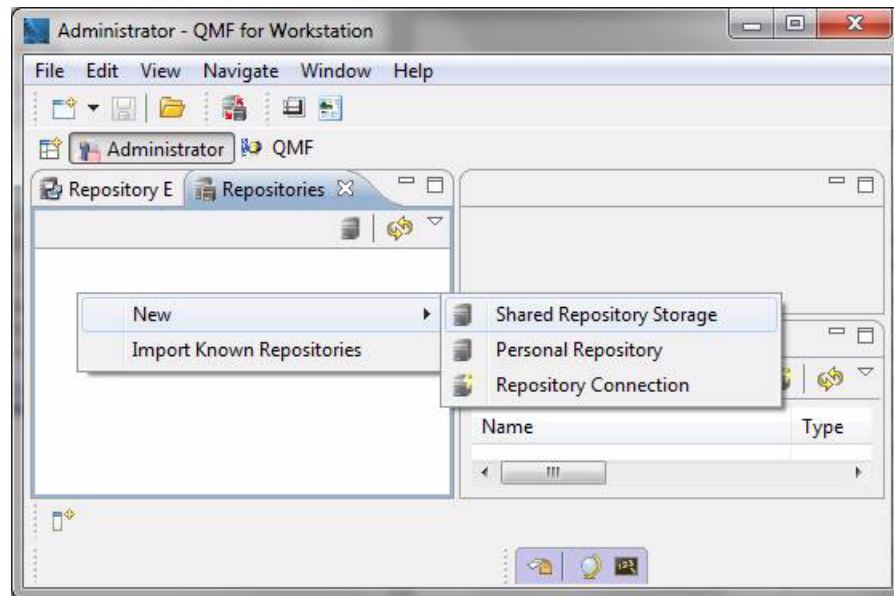


Figure 6-7 Setting up shared repository storage

2. In the Create New Shared Repository Storage wizard, you can specify connection parameters for the new shared repository storage. Figure 6-8 shows the selections to make for connection type and JDBC driver to configure a direct DRDA connection. If you are configuring a JDBC connection, select **DB2 Type 4 UDB driver**.

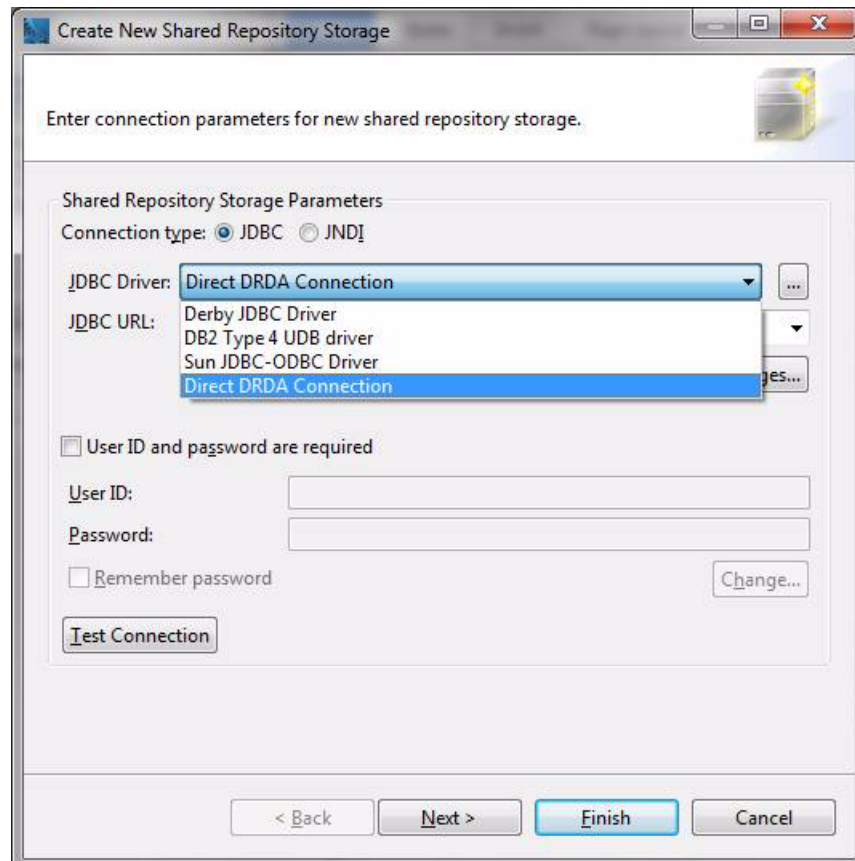


Figure 6-8 Create New Shared Repository Storage wizard

3. The JDBC URL field is prepopulated with the URL `jdbc:db2://host:50000/database` in our example, as shown in Figure 6-9. This URL is the default for the DB2 Type 4 UDB driver. Type over this URL with your proper host name port and database.

For the direct DRDA connection, you must change the URL to be of the following form:

`jdbc:rsbi:db2://{server}:{port}/{database}`

The screenshot shows a window titled "Create New Shared Repository Storage". Inside, there's a section "Shared Repository Storage Parameters". Under "Connection type", "JDBC" is selected. The "JDBC Driver" dropdown is set to "Direct DRDA Connection". The "JDBC URL" field is highlighted, and a dropdown menu is open showing three options: "jdbc:db2://host:50000/database", "jdbc:rsbi:db2://{server}:{port}/{database}" (which is highlighted in blue), and "jdbc:db2://host:50000/database". Below this, there's a checkbox for "User ID and password are required" which is unchecked. There are input fields for "User ID:" and "Password:", and a "Remember password" checkbox. A "Change..." button is next to the "Remember password" checkbox. At the bottom left is a "Test Connection" button. At the bottom right are navigation buttons: "< Back", "Next >", "Finish" (highlighted in blue), and "Cancel".

Figure 6-9 Entering the URL for the database connection

4. Check the **User ID and password are required** check box, as shown in Figure 6-10, then enter the user ID and password required for authentication to the database that will hold shared repository storage.

Create New Shared Repository Storage

Enter connection parameters for new shared repository storage.

Shared Repository Storage Parameters

Connection type: ☒ JDBC ☐ JNDI

JDBC Driver: Direct DRDA Connection

JDBC URL: jdbc:rsbi:db2://rs25:3970/RS25Q91A

Build URL... Advanced... Bind Packages...

☒ User ID and password are required

User ID: twshawnn

Password:

☐ Remember password Change...

Test Connection

< Back Next > Finish Cancel

Figure 6-10 Providing database authentication information

Tip: The **Change** button shown in Figure 6-10 allows you to change your DB2 password when necessary without starting a terminal emulator and logging into z/OS.

5. Click **Test Connection**. Dismiss the resulting confirmation dialog by clicking **OK**.
6. For direct DRDA connections only: Click **Bind Packages** to bind the appropriate application packages to DB2. Bind progress is shown as the operation completes, as shown in Figure 6-11.

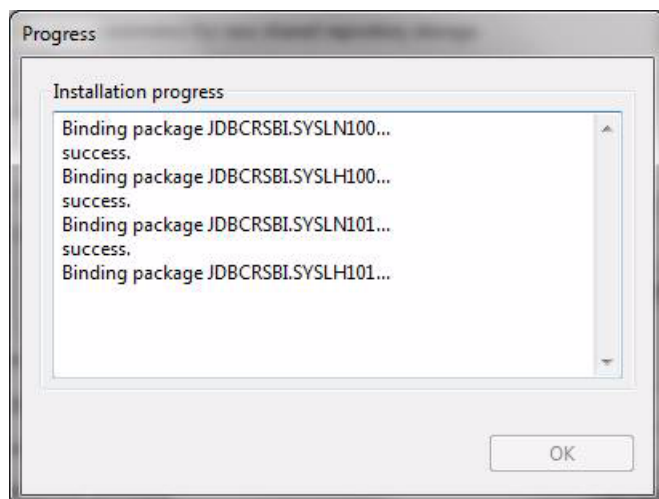


Figure 6-11 Binding the application packages for direct DRDA connections

7. Click **Next** in the Create New Shared Repository Storage wizard. Do not click **Finish** at this point, as this will leave the configuration only partially complete.

As shown in Figure 6-12, select **Create or upgrade repository storage** tables to generate SQL statements to create stogroups, databases, table spaces, views, and indexes.

If the user ID that you specified in Step 4 does not have the privileges to execute these CREATE statements, you can enter a secondary authorization ID in the Owner ID field. QMF for Workstation will then issue a SET CURRENT SQLID statement before running the rest of the SQL. If you do not need to switch IDs, leave the Owner ID field blank.

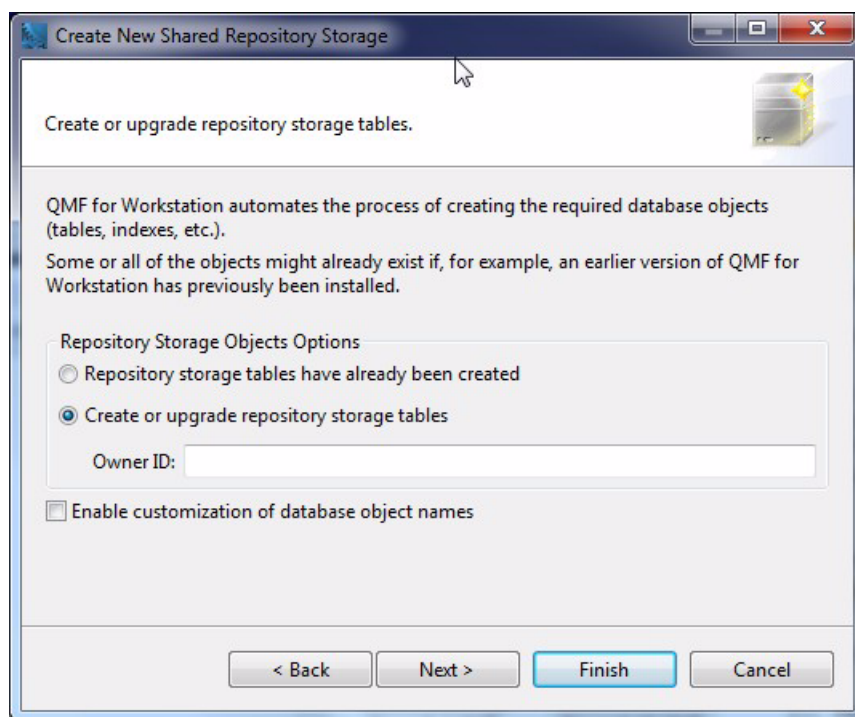


Figure 6-12 Step to create or upgrade repository storage tables

8. Click **Next**. The SQL to be run is displayed. As a backup or documentation option, right-click the SQL and select Copy from the resulting context menu. You can then paste the SQL statements into a document of your choice for future reference.

Click **OK** to run the SQL, then click **Finish** when the SQL has completed to establish the repository storage tables.

9. Protect shared repository storage tables.

Because shared repository storage contains the application information, database connection information, and other data required to operate QMF, QMF allows you to protect access to these tables using either stored procedures or static SQL packages. When protected, these tables in shared repository storage are then accessed only through “hard-coded” SQL during QMF operations. If a user issues queries with statements such as SELECT, UPDATE, INSERT, or DELETE through dynamic SQL directed at these tables, such queries are then rejected.

The database hosting the repository storage must, of course, support the method you choose; two methods are offered, as explained next, because some database management systems do not support both.

To protect shared repository storage, follow these steps:

- a. Check the **Protection method** check box on the Protect Repository Storage Tables page of the wizard.

In cases where the database that is hosting the repository storage does not support either stored procedures or static SQL, you can leave this box unchecked. For DB2, however, protection should always be used.

Select the protection method. Figure 6-13 shows protection by static SQL packages.

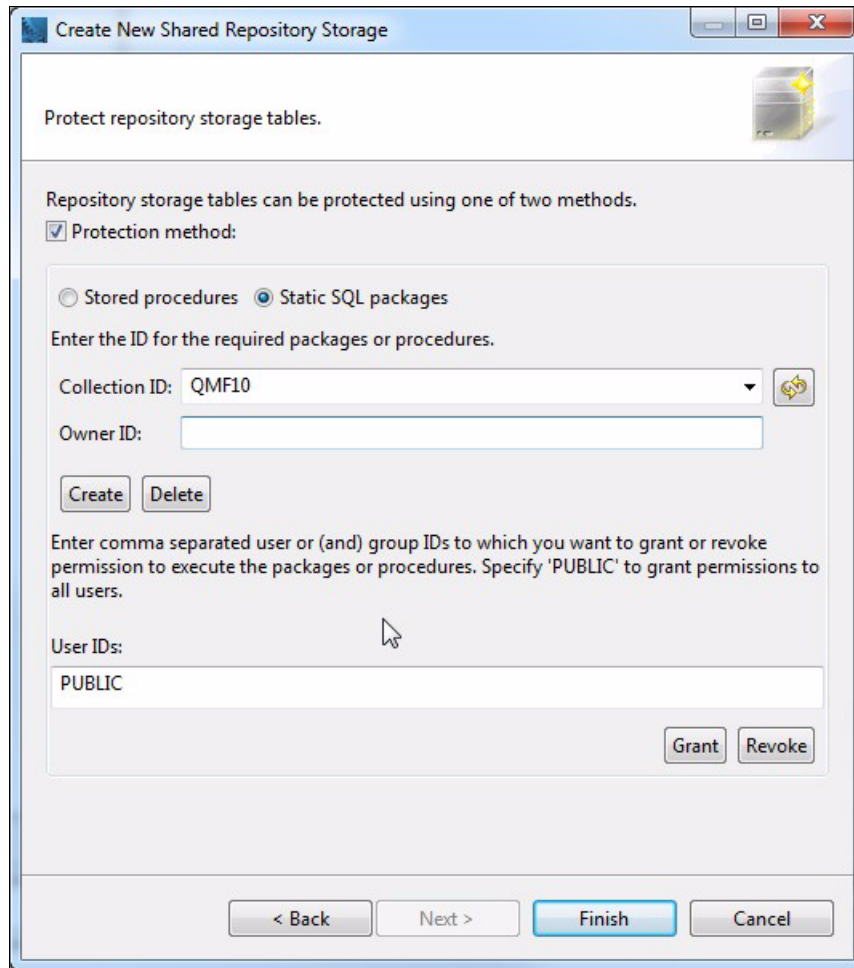


Figure 6-13 Protecting repository storage tables

- b. In the Collection ID field, enter the collection ID that you want to associate with the packages to be created.

Tip: In the past, when fix packs were applied, packages needed to be bound. If the fix pack needed to coexist with a prior fix pack, the collection ID had to be changed and the packages rebound. In this case, it had been a recommended practice to name the collection with a name that clearly identified the fix pack. QMF for Workstation now uses package versioning so that the collection name no longer needs to be changed for fix pack coexistence.

- c. Click **Create** to bind the packages.
- d. When the packages are bound, click **Grant** to grant EXECUTE privileges on these packages to PUBLIC.
- e. Click **Finish**.

The repository storage tables to be used for QMF for Workstation and WebSphere operations are now in place, and access to these tables has been granted to PUBLIC. However, there is no connection information in these tables yet. We look at how to define connection information next.

6.2 Creating a repository within shared repository storage

After creating shared repository storage, you need to create at least one repository that will hold both data source definitions and users' content. Most QMF installations typically have only one shared repository storage and one repository within it.

In a traditional QMF setting, the shared repository storage tables will hold connection information for the data sources, as the QMF server definition file did in prior QMF versions, only now the object storing the connection information is a repository.

A shared repository is centrally located in a database, DB2 for z/OS in these examples, and holds connection strings for the various DB2 data sources and other data for production.

To set up a shared repository, follow these steps:

1. In the Administrator perspective, navigate to the **Repositories** view. Expand the tree for the shared repository. If it is a database that is new to QMF for Workstation, there is an indicator that there are no repositories, as shown in Figure 6-14.

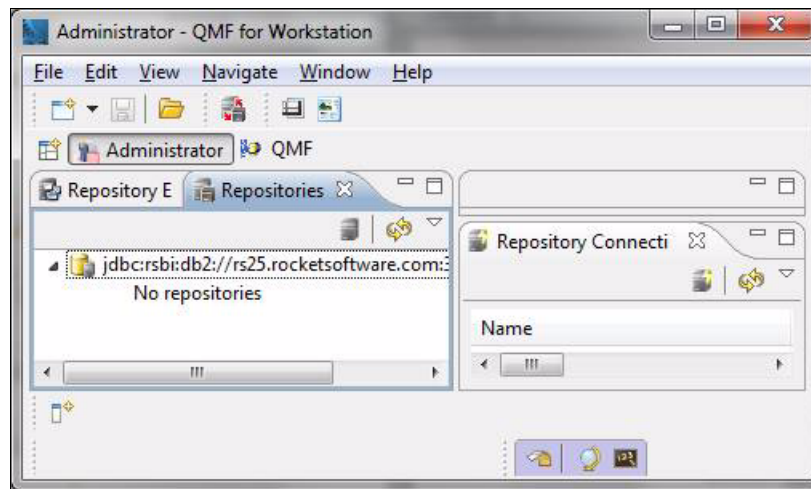


Figure 6-14 Setting up a shared repository

2. Right-click the JDBC URL associated with the shared repository storage and select **New** → **Shared Repository**, as shown in Figure 6-15.

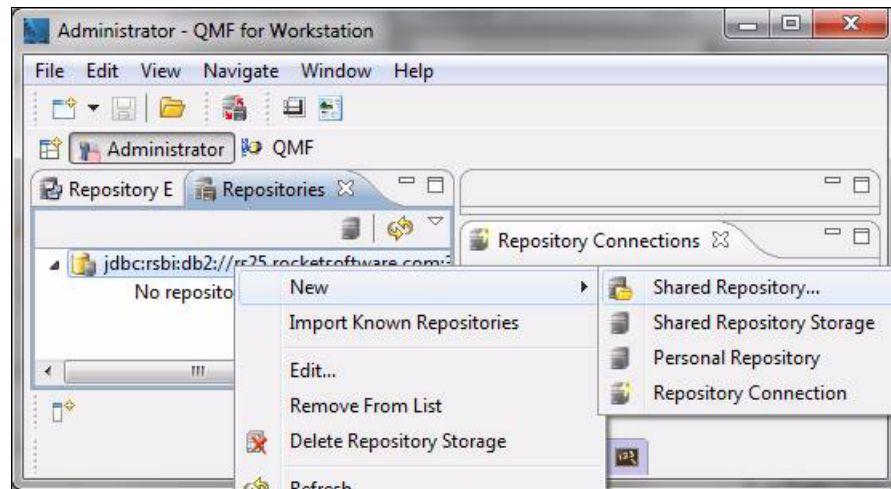


Figure 6-15 Specifying the shared repository JDBC URL

3. Give the repository a name. End users will not see this name, so it can be as technically meaningful as you want. An example is shown in Figure 6-16.

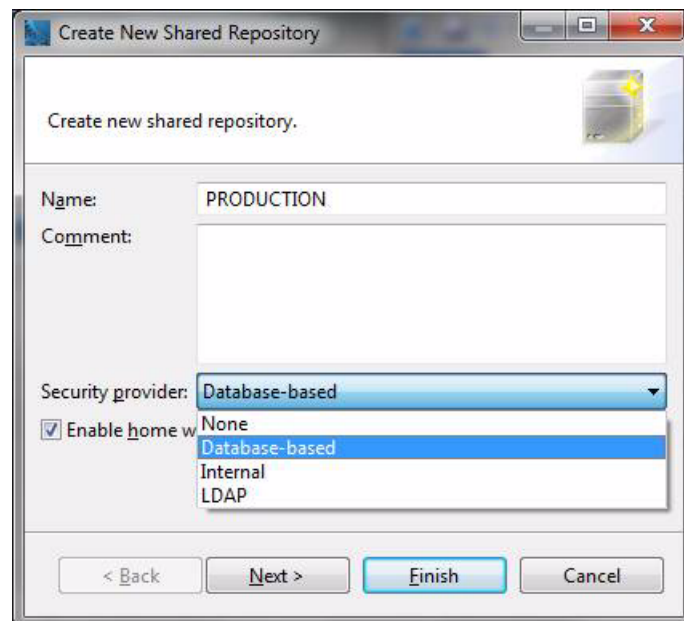


Figure 6-16 Naming a repository

4. From the **Security provider** drop-down list, select **Database-based**, which specifies that the security settings of the host database are used to protect the shared repository.

The security provider setting allows you to assign privileges such as Browse, Edit, and Full to users of the repository content. (Chapter 8, “Configuring security” on page 145 explains the security provider settings in more detail.)

Important: You cannot change this security provider setting after it has been established.

Leave **Enable Home Workspace Support** checked, then click **Next**. The Enable Home Workspace Support option specifies that a home workspace will be automatically created every time a new user logs in. A home workspace is created automatically when a user connects for the first time; it is owned by the user and is created with no permissions for PUBLIC. Other workspaces are created by users and are owned by them but have Browse set as the default privilege for PUBLIC.

5. In the DB Security Provider Options page of the wizard, leave **Import User names from QMF Catalog** checked and click **Finish** to create the shared repository, as shown in Figure 6-17.

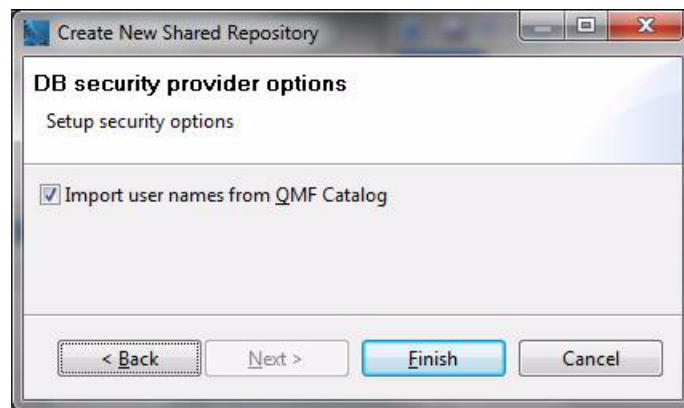


Figure 6-17 Importing user names from the QMF catalog

Tip: Whereas the creation of shared repository storage installs a large set of database objects, creation of the shared repository inserts just a few rows in these tables. As previously mentioned, the shared repository will function as the server definition file formerly did in older QMF versions.

- Expand the tree for the shared repository, as shown in Figure 6-18. There is a slight delay while some configuration takes place. User IDs from QMF are downloaded and container names for different types of data sources are loaded into the sections of the repository storage tables labeled with the repository name. The shared repository name can be thought of as a schema, or a high-level qualifier or name tag.

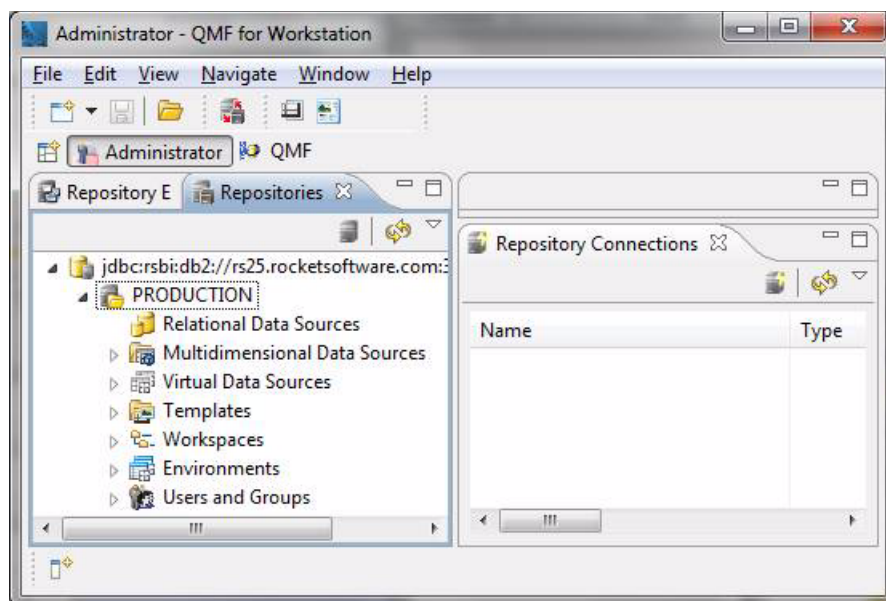


Figure 6-18 Expanded Repository Explorer tree

6.3 Configuring access to data sources

After you create a repository within shared repository storage, it is initially empty. To begin to populate the repository with content, you must first add configuration information for each data source that will be accessible from the repository.

You can configure access to the following types of data sources:

- ▶ Relational
- ▶ Multidimensional
- ▶ Hierarchical
- ▶ Semi-structured

The topics that follow provide steps for how to configure access to each of these types of data sources. Default settings for data source definition are shown throughout, as these defaults are sufficient in most cases.

6.3.1 Adding a relational data source

To configure QMF access to a relational data source, follow these steps:

1. If you have not done so already, add database driver information for the data source, as explained in 6.1.1, “Defining connectivity to the database that will host the shared repository storage” on page 90.

2. Right-click the Relational Data Sources branch of the repository and select **New** → **Relational Data Source**.

The Create New Relational Data Source wizard is displayed, as shown in Figure 6-19. In this wizard, you specify connection information for the database and test the connection before proceeding. This wizard is similar to the Create New Shared Repository Storage wizard previously explained.

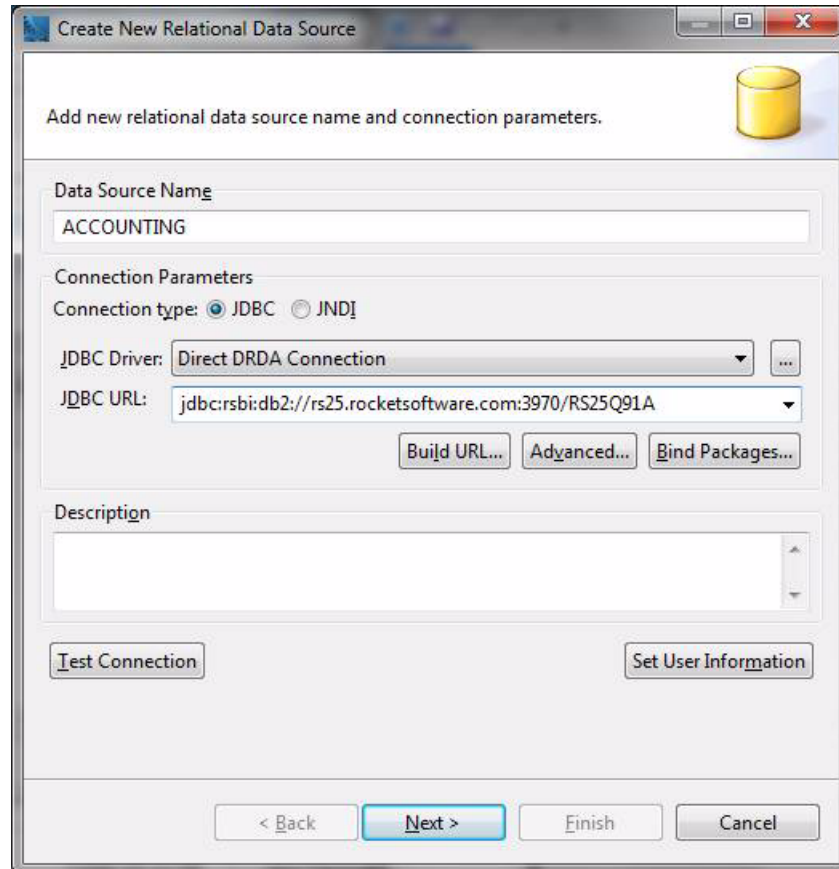


Figure 6-19 Specifying database connection in the Create New Relational Data Source wizard

3. Enter a data source name. This name can be anything you want, but keep in mind that end users *will* see this name. Because it can be useful at times to refer to this data source in scripts, it might be easier if the name does not contain spaces or special characters, such as single quotes.
4. Select a connection type of JDBC.
5. Select a JDBC driver from the drop-down list. For DB2 for z/OS, select either **DB2 Type 4 UDB driver** or **Direct DRDA connection**. If you select **Direct DRDA connection**, ensure the JDBC URL field contains a URL that conforms to the following syntax:

```
jdbc:rsbi:db2://{server}:{port}/{database}
```

The DRDA packages do not need to be bound again if they were already bound when shared repository storage was created. If it is a data source for which packages have not yet been bound, click **Bind Packages** at this time. There are no other differences between the setup for the DB2 Type 4 UDB driver and the Direct DRDA connection.

6. Click **Set User Information**. The User Information page of the wizard is displayed, as shown in Figure 6-20. Here you can specify user authentication credentials for the data source on the User Information page of the wizard.

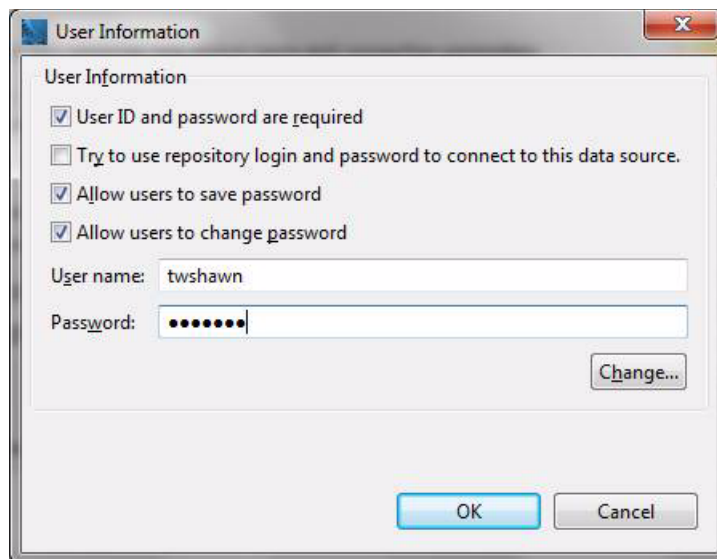


Figure 6-20 Specifying user authentication credentials on the User Information page of the wizard

On this page, accept the default choices for the check boxes and specify the appropriate user name and password. Then click **OK**.

For future reference, notice the **Try to use repository login and password to connect to this data source** check box. If the user ID and password are the same for this database as they are for shared repository storage, checking this box will cause the login dialog for this data source to be suppressed and the credentials that were used for shared repository storage will be reused.

When the Create New Relational Data Source wizard is again displayed, click **Test Connection**. When the connection test succeeds, click **Next**.

The Enable Data Source Plug-ins page is displayed. This dialog, shown in Figure 6-21 on page 109, is very important for QMF for TSO and CICS users. The QMF catalog plug-in enables QMF for Workstation to access any QMF product family objects (queries, forms, and procedures) that are stored in the QMF catalog. New QMF for Workstation queries, classic report forms, and procedures can also be accessed from QMF for TSO and CICS when this plug-in is enabled and those objects have been saved to the QMF catalog.

7. Under QMF Catalog Plug-in, check the **Enable plug-in** check box to enable the QMF catalog, as shown in Figure 6-21.

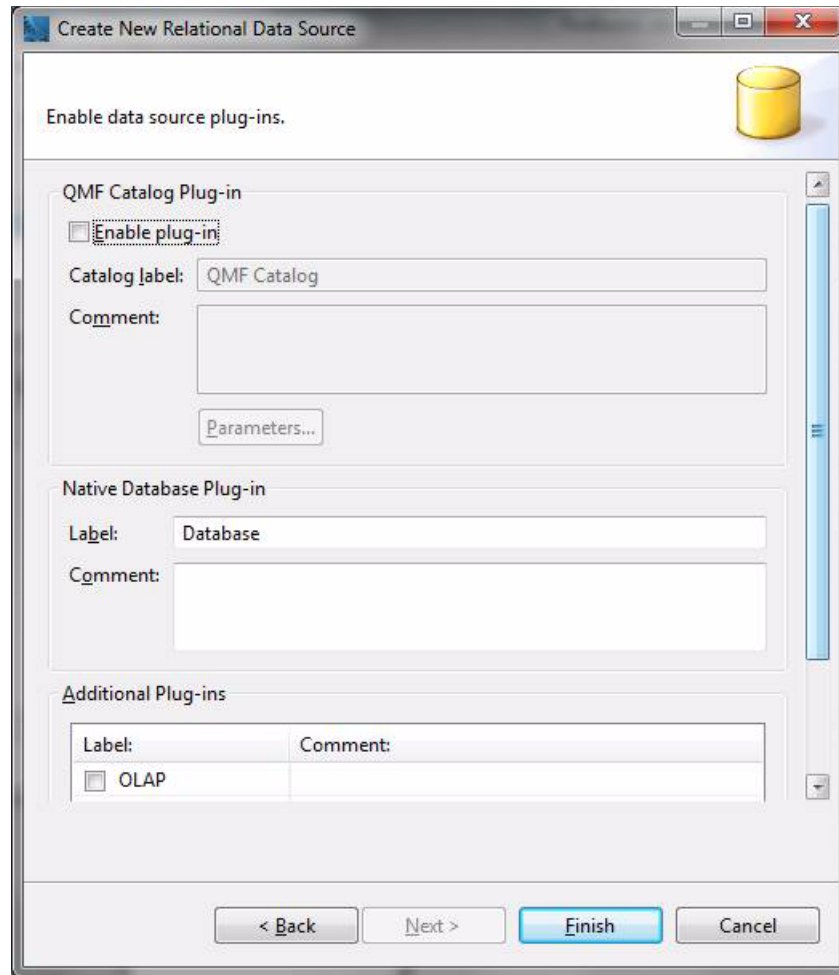


Figure 6-21 Enabling the QMF catalog plug-in

Selecting this check box automatically launches the QMF Catalog wizard.

8. Select the **Create or update catalog tables to support long names** radio button, as shown in the QMF Catalog wizard dialog in Figure 6-22. Leave the **Enable customization of database** object names check box unchecked and click **Next**.

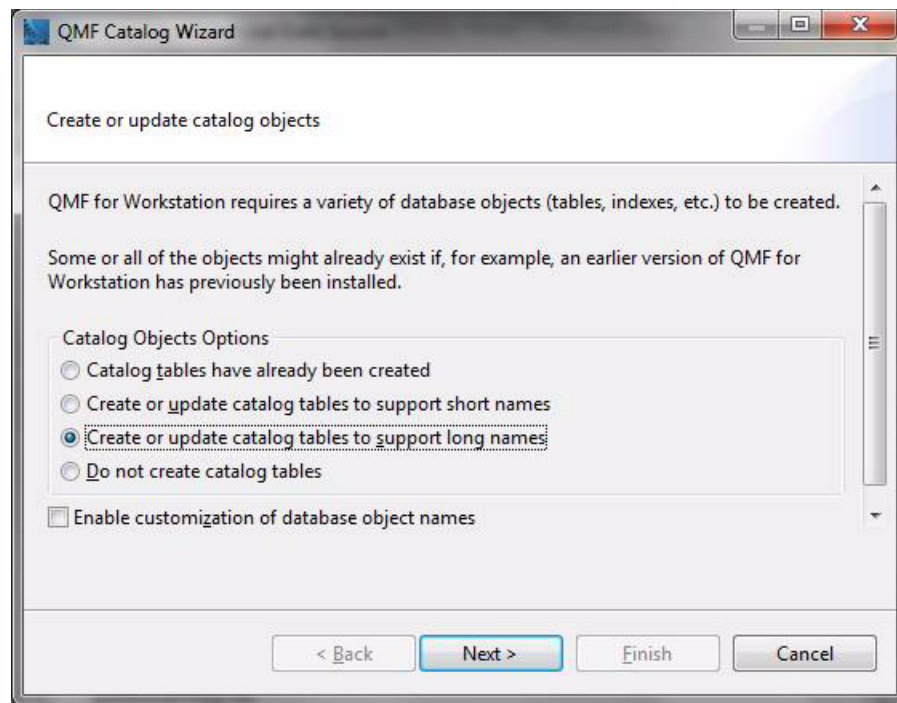


Figure 6-22 Setting up the QMF catalog to support long names

The next dialog covers choices you need to make for object lists.

9. Read the options on the Choose Object Listing Option page in Figure 6-23, carefully, as they explain tradeoffs in database resources used when generating object lists.

The **Include all objects** option, selected in Figure 6-23, simply queries SYSIBM.SYSTABLES for all tables and views matching the filter specified by the user. With this option, the list will be larger but will be returned quickly.

The other options include joins to SYSIBM.SYSTABAUTH and SYSIBM.SYSUSERAUTH to determine the tables for which the current user has SELECT access.

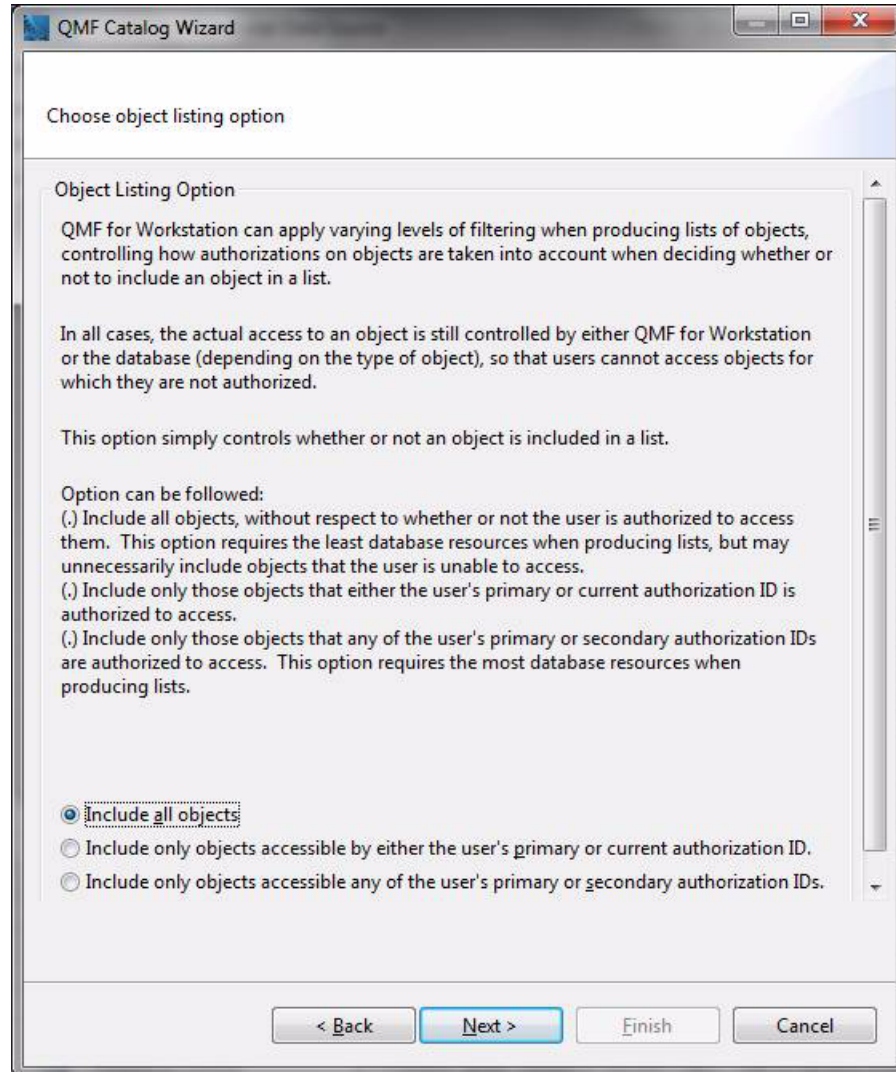


Figure 6-23 Descriptions of object list options

Click **Next**. QMF for Workstation now checks to see if all the necessary control tables, views, and other objects are in place. If not, it displays the SQL that will create whatever is missing. Again, it is a good practice to copy and paste the SQL into a document that you can use for future reference if necessary.

Click **OK** to run the SQL.

10. Protect the QMF catalog tables.

The QMF catalog tables, like the shared repository storage tables, need end user access granted to them and should be protected from ill-informed end user updates.

- a. To protect the tables, select **Always** under Connect Using Protected Mode, as shown in Figure 6-24.

QMF Catalog Wizard

Protect QMF Catalog tables

Packages

Connect using Protected Mode

☐ Never ☐ If possible ☒ Always

☒ Protect

QMF Catalog tables can be protected using one of two methods.

☐ Stored procedures ☒ Static SQL packages

Enter the collection ID for the required packages or procedures.

Collection ID: QMF10

Owner ID:

Create Delete

Enter the user IDs to which you want to grant or revoke permission to execute the packages or procedures. Specify "PUBLIC" to grant the permission to all users.

User IDs:

PUBLIC

Grant Revoke

< Back Next > Finish Cancel

Figure 6-24 Specifying protection for the QMF catalog tables

- b. Make sure the **Protect** check box is checked and select the protection method. Step 9 of the procedure for setting up shared repository storage, in “Protect shared repository storage tables.” on page 101, explains these methods.
- c. Enter a collection ID to be associated with the packages to be bound. It does not have to be the same collection as the one used for the repository storage.
- d. As with repository storage, enter a secondary authorization ID in the Owner ID field only if the primary ID in use does not have the authority to bind packages.
- e. Click **Create**. This operation takes longer than binding the packages for the repository storage protection.
- f. When the binds are complete, click **Grant** to grant EXECUTE privileges on the packages to PUBLIC.
- g. When the grants are complete, click **Next**.

On the Select QMF Catalog page (Figure 6-25), select the name of the data source that will host the QMF catalog. Only DB2 data sources can host the QMF catalog.

Also on this page, leave the name of the QMF catalog as Default and click **Finish**.

QMF Catalog Wizard

Select QMF catalog

Catalog

Choose the data source that hosts the QMF catalog. It can be the same as or different from the current data source. Only DB2 data sources can host a QMF catalog.

There is no restricted catalog information.

Data source name: ACCOUNTING

Refresh List

ID	Name	CCSID
0	Default	37 (U.S. English EBCDIC)

☐ Require the use of this catalog server when accessing the current server

< Back Next > Finish Cancel

Figure 6-25 Selecting the data source that will host the QMF catalog

11.If you want, set up the resource limits that will govern user sessions.

In the past, these resource limits have been stored in the QMF catalog in the DB2 data source. QMF for Workstation now allows options for the resource limits to be stored in the repository storage tables instead. However, notice that when the limits are stored in repository storage, QMF for TSO and CICS cannot share resource limits with QMF for Workstation sessions. It also means that all data sources with repository resource limits will have the same limits.

To preserve sharing of resource limits between QMF for TSO and CICS as well as QMF for Workstation, select **QMF Catalog** from the **Resource limits provider** drop-down list, as shown in Figure 6-26.

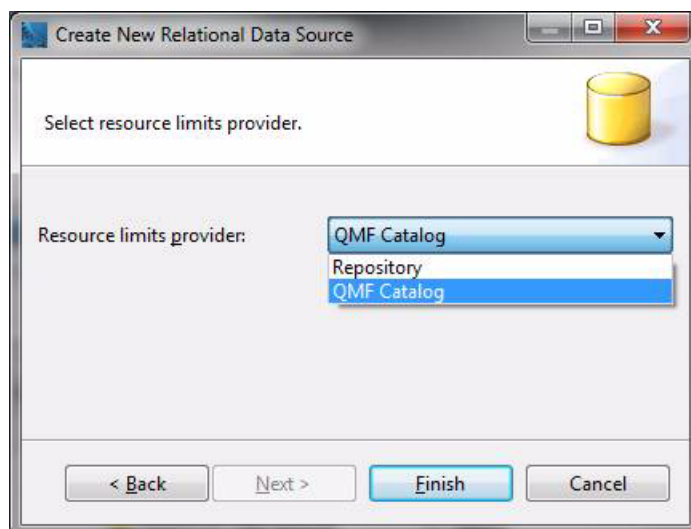


Figure 6-26 Specifying the location of QMF resource limits information

Important: You cannot change this resource limits provider setting after it has been established.

12.Click **Finish** to complete the setup for the new relational data source.

6.3.2 Adding a multidimensional data source

As with relational data sources, data source configuration information must be added to the repository for any multidimensional data sources users will need to access from their workspaces. Follow these steps to add an OLAP data source.

1. Switch to the Administrator perspective, if it is not already displayed, by selecting **Window** → **Open Perspective** → **Administrator** from the menu.
2. If you have not done so already, add database driver information for the data source, as explained in 6.1.1, “Defining connectivity to the database that will host the shared repository storage” on page 90.

3. In the **Repositories** view, right-click Multidimensional Data Sources and select **New** → **Multidimensional Data Source** from the context menu. The window shown in Figure 6-27 is displayed.

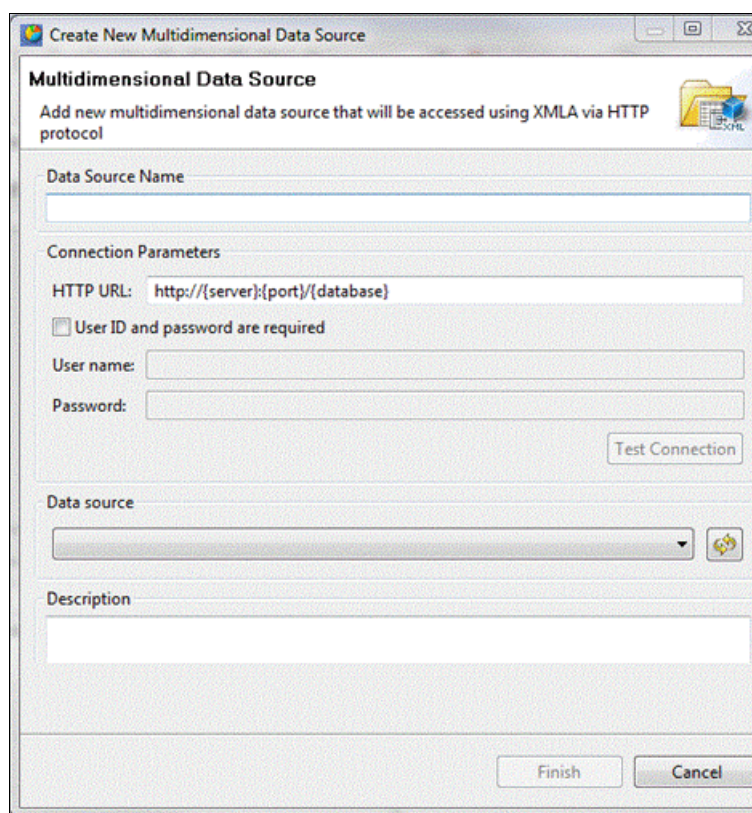


Figure 6-27 Creating a new multidimensional data source

4. Enter a meaningful name in the Data Source Name field.
5. Enter the appropriate URL.

Most OLAP servers use XML for Analysis (XMLA), an http-based protocol that does not require a driver. The XMLA URL format depends on the location of the OLAP server's web interface, the port, and the database that is to be accessed.
6. Click the **Refresh** icon to the right of the Data source field to populate the field with data sources, then select the data source that contains the OLAP data.
7. Click **Finish**. A connection to the OLAP data source has now been defined and can be used in creating queries and dashboards or any other operations where it is required.

6.3.3 Adding a hierarchical data source

Operational analytics is a term often used synonymously with “real time” analytics. The traditional data warehouse is often built at a level of aggregation that provides a very high-level view of information (for example, summaries by territory over time), but this view leaves out detailed information.

Operational analytics serves two purposes:

- It provides access to information that is as current as possible, such as the information a customer service representative might need.

- It provides data at the lowest, most granular level of detail for those who need to deal with the data at a micro level.

A large percentage of operational data is based in IMS. IMS installations typically extract the data periodically or use change data capture (CDC) technologies to get the IMS data out and replicated as quickly as possible. Sometimes the turnaround is simply not adequate for specific purposes and therefore direct access is desired. However, direct access to IMS data must be done with care.

Beginning with IMS 11 and continuing with IMS 12, a new set of possibilities for providing analytics directly within IMS was introduced in the form of open database access with JDBC, which is how QMF supports IMS directly.

The traditional IMS installation is very leery of allowing direct access to data from any business analytics tool due to the nature of ad-hoc queries. However, approaches that place constraints on the queries and embed the function directly within a QMF dashboard allow IMS enterprise accounts creative access to IMS data while keeping the appropriate controls in place.

The IMS Open Database environment

There are two universal JDBC drivers with IMS (IMS Universal DB Resource Adapter and IMS Universal JDBC Driver) that offer a greatly enhanced JDBC implementation with the following benefits:

- Local commit/rollback support
- Keys of parent segments are included in tables as foreign keys, which allows standard SQL implementation for the SQL subset supported
- Updatable result sets
- Metadata discovery API implementation, which uses metadata generated by the DLIModel Utility as “catalog data” and also enables JDBC tools to work with IMS databases just as these tools work with DB2 databases

The IMS Open Database environment, which is present in IMS 11 and later, is graphically illustrated in Figure 6-28.

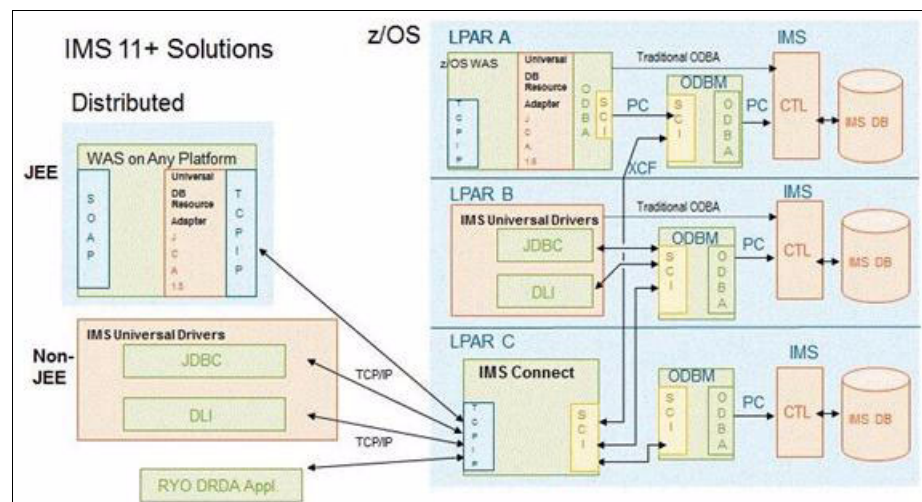


Figure 6-28 The IMS Open Database environment

Attention: Customers migrating from IMS 9 to IMS 11 must migrate off of the Security Maintenance Utility before migrating to IMS 11.

IMS 12 extends the capabilities of IMS even more. Here is a short list of benefits to consider if you are currently running a pre-12 version of IMS:

- ▶ IMS catalog support:
The IMS catalog is a comprehensive view of IMS database metadata fully managed by IMS. By externalizing this metadata, IMS can participate in solutions that require the exchange of metadata, such as business impact analysis. The IMS Universal drivers are enhanced to take advantage of the IMS catalog.
- ▶ TCP/IP support connects two IMS systems in a Multiple IMS Systems Coupling (MSC) network.
- ▶ Expanded communication between IMS systems.
- ▶ Additional RACF security codes.
- ▶ A centralized data store for dynamic resource definitions.
- ▶ Eclipse-based graphical tool that simplifies IMS application development tasks.
- ▶ New IMS and IMS Connect QUERY and UPDATE commands improve availability.
- ▶ Better problem analysis and enhanced RAS increase data quality.
- ▶ Fast Path 64-bit buffer manager support alleviates bottlenecks.
- ▶ Database Recovery Control (DBRC) enhanced LIST commands offer support for greater than 32KB output.
- ▶ Dynamic full-function database buffer pool and constraint relief.
- ▶ Fast Path (FP) secondary index enablement.

Adding an IMS data source

An IMS database consists of various segments and provides a child/parent relationship view of data. For example, Figure 6-29 shows a table with three levels in its hierarchy. The way it is mapped back to QMF as virtual tables allows the user to join them in any sequence, provided it makes logical sense. The three ovals represent three different ways of joining the data in QMF.

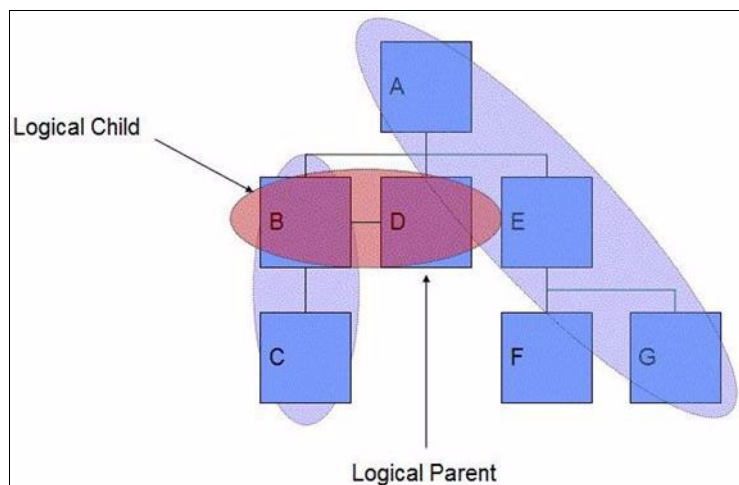


Figure 6-29 Join options for IMS data in QMF

Setting up access to IMS follows the same basic process as adding a relational data source and uses the Create New Relational Data Source wizard.

To configure access to an IMS data source, follow these steps:

1. If you have not done so already, add database driver information for the data source, as explained in 6.1.1, “Defining connectivity to the database that will host the shared repository storage” on page 90.
2. Select **File** → **New** → **Relational Data Source** or right-click **Relational Data Sources** in the **Repository Explorer** tree and select **New Relational Data Source**.

The Create New Relational Data Source wizard is displayed, as shown in Figure 6-30.

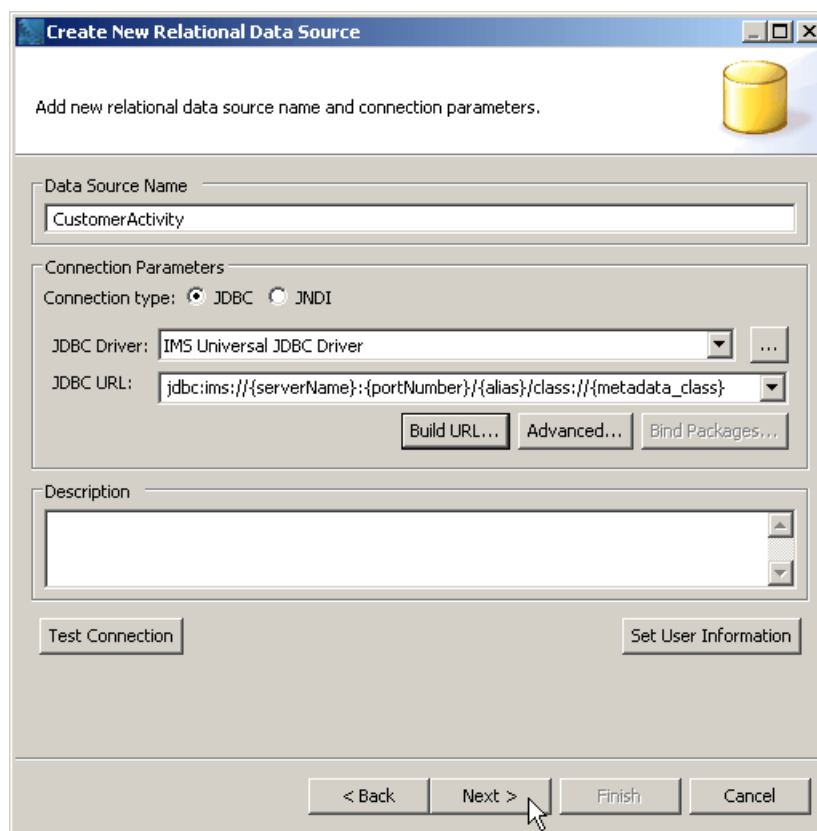


Figure 6-30 Adding an IMS data source using the IMS Universal JDBC driver

3. Select **IMS Universal JDBC Driver** from the **JDBC Driver** drop-down list, as shown in Figure 6-30.

QMF for Workstation and QMF for WebSphere both include support for the IMS JDBC driver.

4. In the JDBC URL field, specify the location of the IMS JDBC driver in the format shown in Figure 6-31.

To have QMF for Workstation build the URL for you, click **Build URL**. The Build URL By URL Template window is displayed, as shown in Figure 6-31. System-specific values are supplied in the Value column.

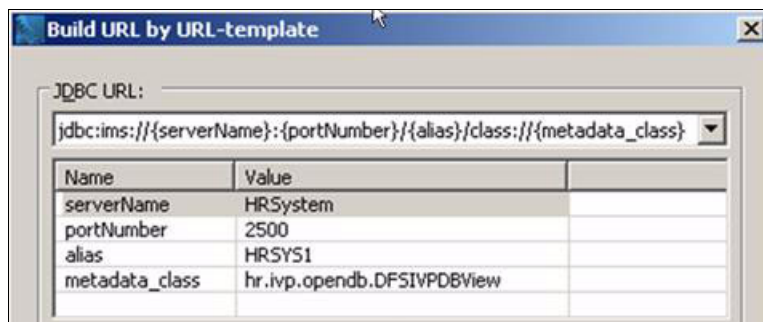


Figure 6-31 Building the IMS URL with assistance from QMF

5. Select the appropriate template from the JDBC URL drop-down list and supply the values to the right of the parameter names. Each field value is specific to your particular IMS environment.
6. Click **OK**.

QMF for Workstation automatically forms the JDBC URL that will be used to connect to IMS. IMS data is now available much like any other data source.

After the IMS data source is defined, QMF sees the IMS system as a relational data source. IMS tables appear in the navigation tree and queries against them can be built using the SQL, prompted, or query diagram view interfaces, which are explained in Chapter 9, “Getting to the data you need: Query methods” on page 191.

6.3.4 Adding a semi-structured data source

QMF for Workstation and QMF for WebSphere, when running on Windows, support ODBC connectivity. Semi-structured data sources, such as Lotus Notes, appear to QMF as ODBC-based data sources, and thus connectivity to these sources can be configured from the QMF side by the JDBC-ODBC driver included with the products.

To connect QMF to an ODBC data source, follow these steps:

1. Open the Create New Relational Data Source wizard, shown in Figure 6-32, by selecting **File → New → Relational Data Source** on the application menu or by right-clicking **Relational Data Sources** in the Repository Explorer and selecting **New Relational Data Source** from the resulting menu.

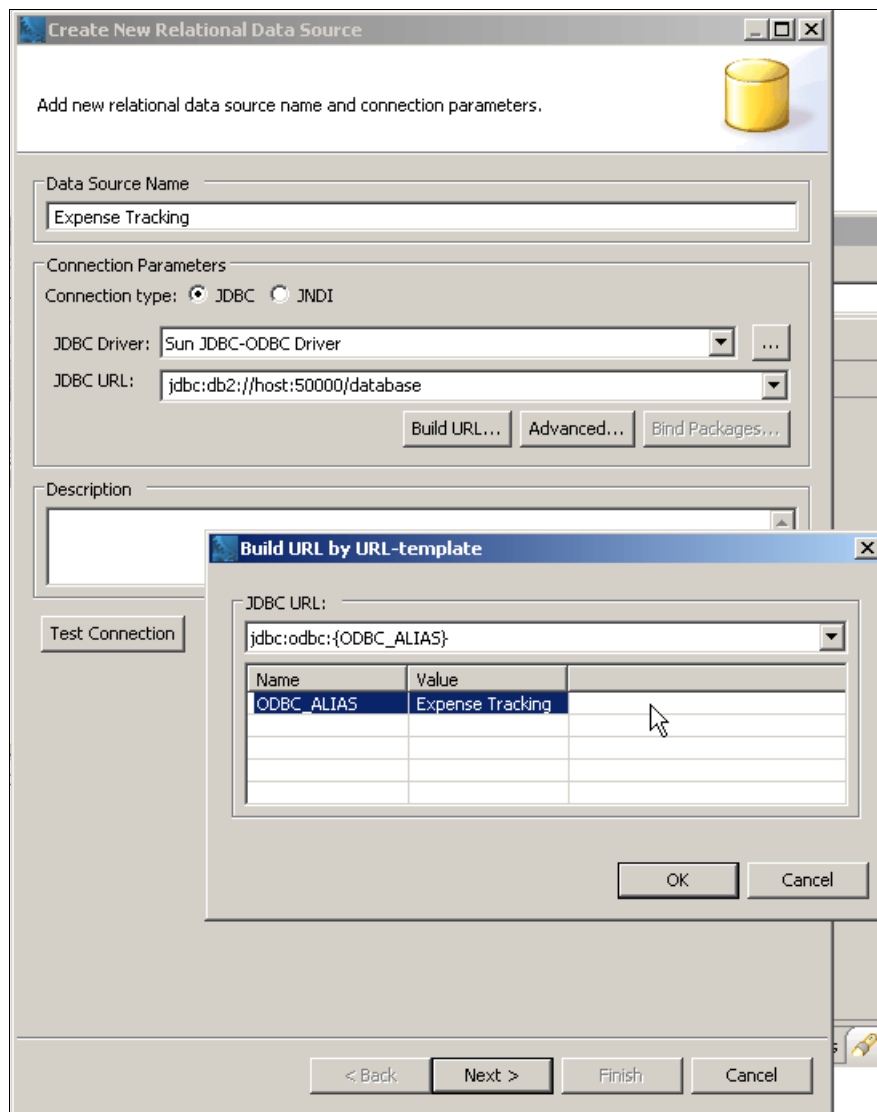


Figure 6-32 Connecting to an ODBC data source using QMF for Workstation

2. Enter the name of the ODBC data source. This name should be predefined in the Windows ODBC Administrator.
3. From the JDBC Driver drop-down list, select the JDBC-ODBC driver.
4. Click **Build URL** to have QMF build the JDBC URL for you.
5. Complete the remaining pages of the Create New Relational Data Source wizard, accepting all the default values.

6.4 Populating the Repository Explorer

After you initially define your data sources, they appear in the Repository Explorer tree, but are empty because they are not populated until a repository connection is established.

At this point, there is a connection to the repository storage database in our example DB2 subsystem, RS25Q91A. We set up this connection earlier in the chapter in 6.1, “Creating shared repository storage” on page 90. It is a traditional client-server connection from a client application (QMF for Workstation) to a database. However, this administrative connection can only execute configuration-related tasks. It cannot issue queries that pull result sets.

While, in general, there is no need for more than one repository, you are not restricted to one. There can be a number of repositories configured in the Repositories view. A key part of the architecture is that, within the Repositories view in the Administrator perspective, many repositories can be configured and viewed at once; in the end user perspectives (QMF, User, and Visual Designer), only one repository can be in play at a time.

To enforce this rule, QMF for Workstation has the concept of the repository connection. The repository connection designates the single repository that will be made available to the end user perspectives. It is necessary in order to keep the end user from being overwhelmed when there are myriad data sources available.

Because the RS25Q91A data source in our examples will function as both the repository storage data source and a production data source, the following steps show you how to create a repository connection to the relational data source in this subsystem (named ACCOUNTING) that we defined to QMF earlier in 6.3.1, “Adding a relational data source” on page 106.

To create a repository connection:

1. In the Repositories view, right-click the repository name and select **New** → **Repository Connection**, as shown in Figure 6-33.

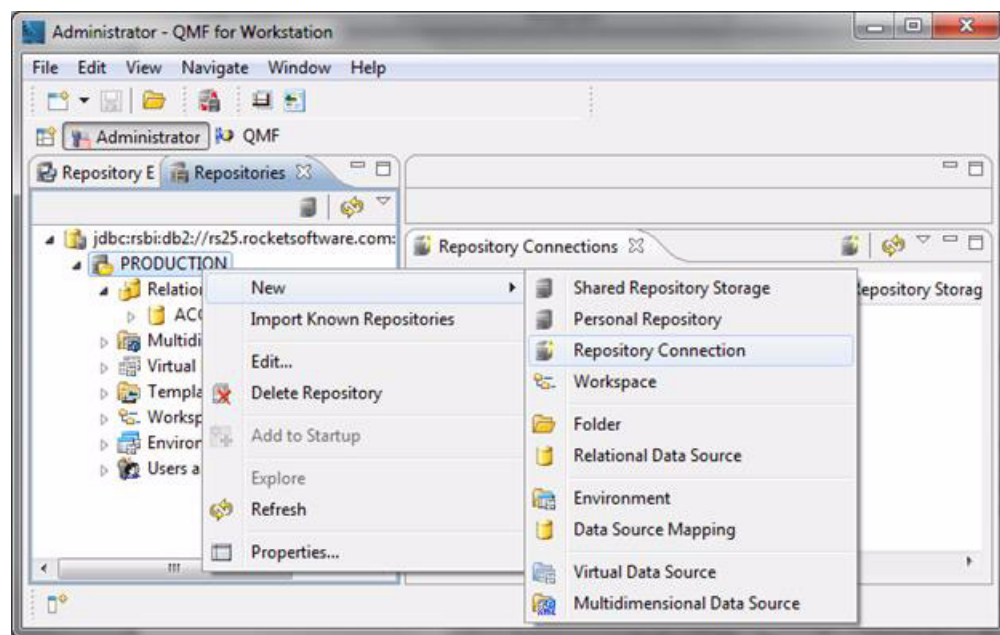


Figure 6-33 Creating a repository connection

2. Change the name of the repository connection to a user-friendly name, as shown in Figure 6-34. It is the name that end users will see. The rest of the information should already be filled in from the repository storage configuration done in 6.1, “Creating shared repository storage” on page 90.

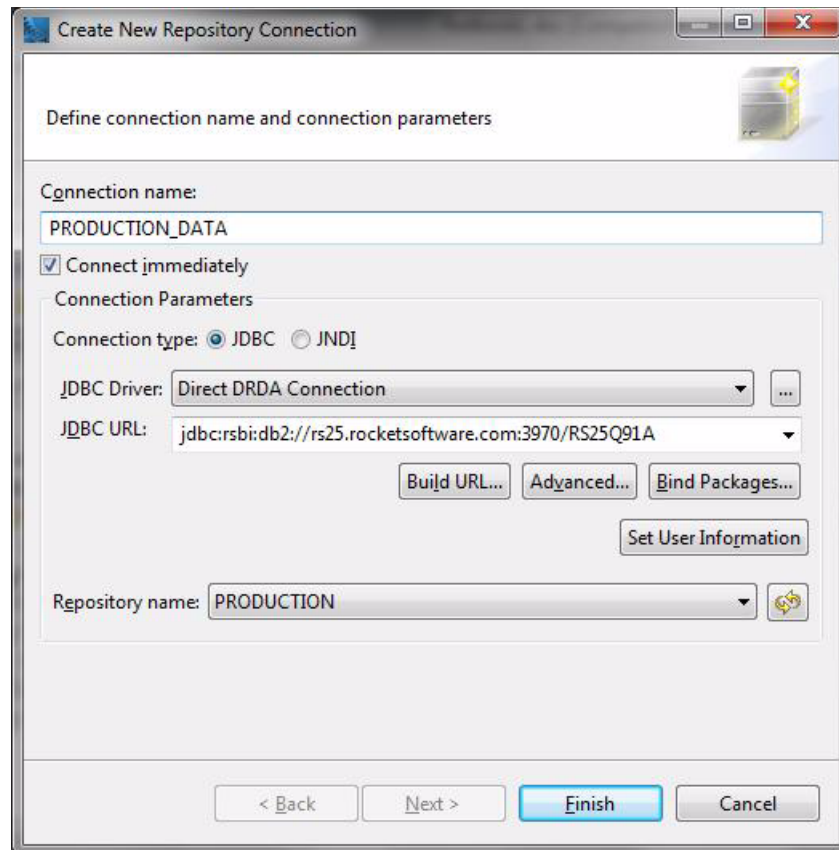


Figure 6-34 Naming the repository connection

Connection: By default, the **Connect immediately** check box is checked. This setting establishes a connection to the repository storage database and initiates the gathering of information tagged with the qualifier/repository name, followed by a tree display of this information in the Repository Explorer.

To make this connection to the repository storage database, a user ID and password is needed. In the examples shown here, the user ID and password from the original connection to the repository storage database is used. This user ID and password was put in place in the User Information page of the Create New Relational Data Source wizard, shown in Figure 6-20 on page 108.

3. Click **Finish** to see the implementation of the setup done so far, shown in Figure 6-35.

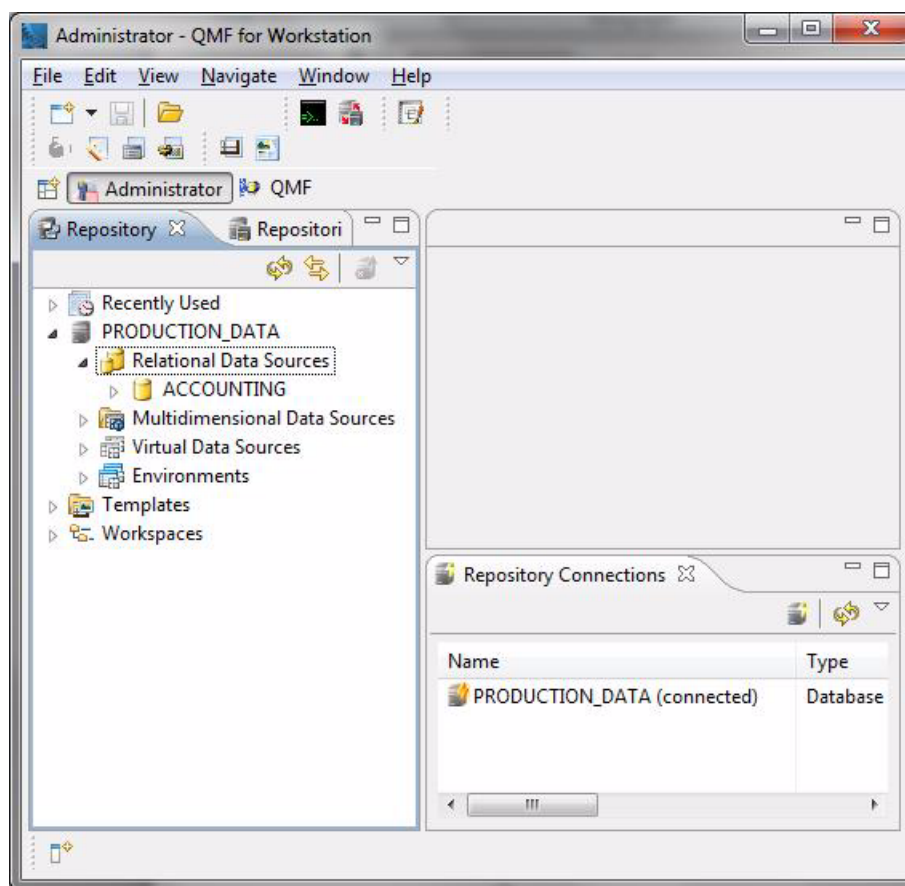


Figure 6-35 The Repository Explorer, showing the set up we have done so far

In the **Repository Connections** view in the lower left pane, the status of the entry for the PRODUCTION_DATA repository connection is “connected”. In the **Repository Explorer** view, the navigation tree can be expanded, showing the ACCOUNTING relational data source. If the product is properly shut down, the active connection to PRODUCTION_DATA is recorded and, the next time QMF for Workstation is started, QMF automatically attempts to connect to this data source.

When the Relational Data Sources branch is expanded, QMF prompts twice for credentials:

- ▶ One time to access repository storage to get connection information for the data source
- ▶ A second time for the connection to the data source itself, which, in this case, is our ACCOUNTING data source which holds production data the user needs to work with

Authentication credentials might not be the same in both cases.

Although, in this example, the repository storage database and the production database are both DB2 and are the same subsystem, there are still two logins that must take place at the beginning of each work session. We will show you in Chapter 8, “Configuring security” on page 145 how to minimize the number of logins required if you want to do so.

Figure 6-36 shows the prompt for repository storage located in DB2 subsystem RS25Q91A.

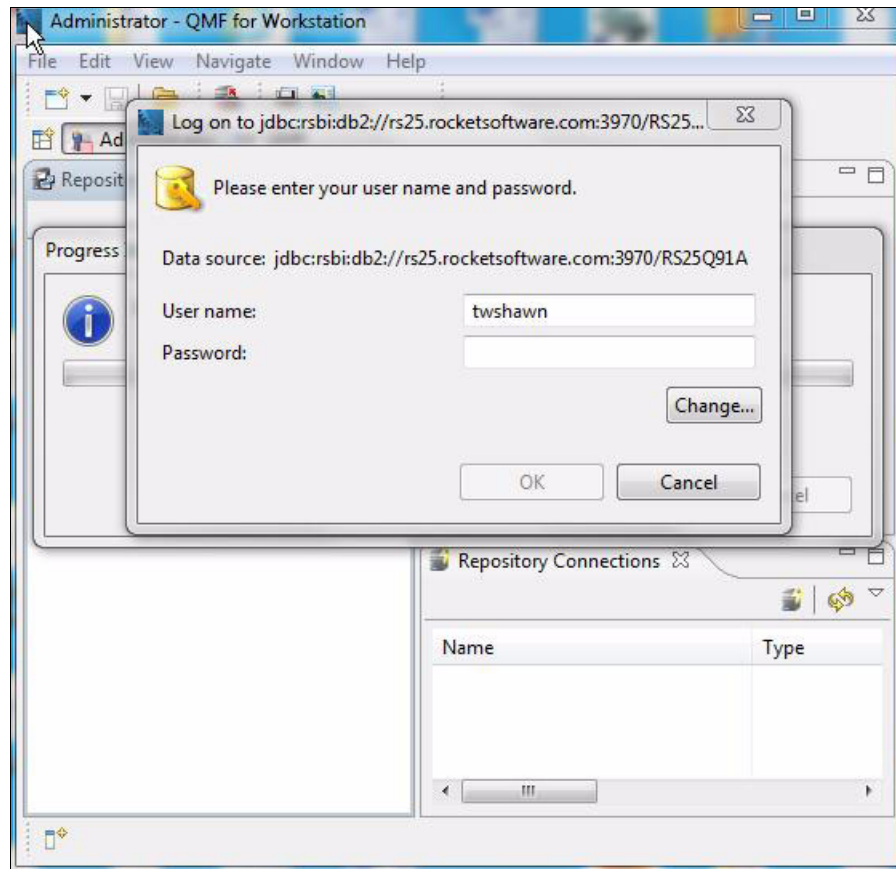


Figure 6-36 User ID and password prompt for repository storage on DB2 subsystem RS25Q91A

When the ACCOUNTING data source is expanded in the **Repository Explorer** tree, the user is again prompted to log into DB2 subsystem RS25Q91A to get to the ACCOUNTING data source on that subsystem, as shown in Figure 6-37.

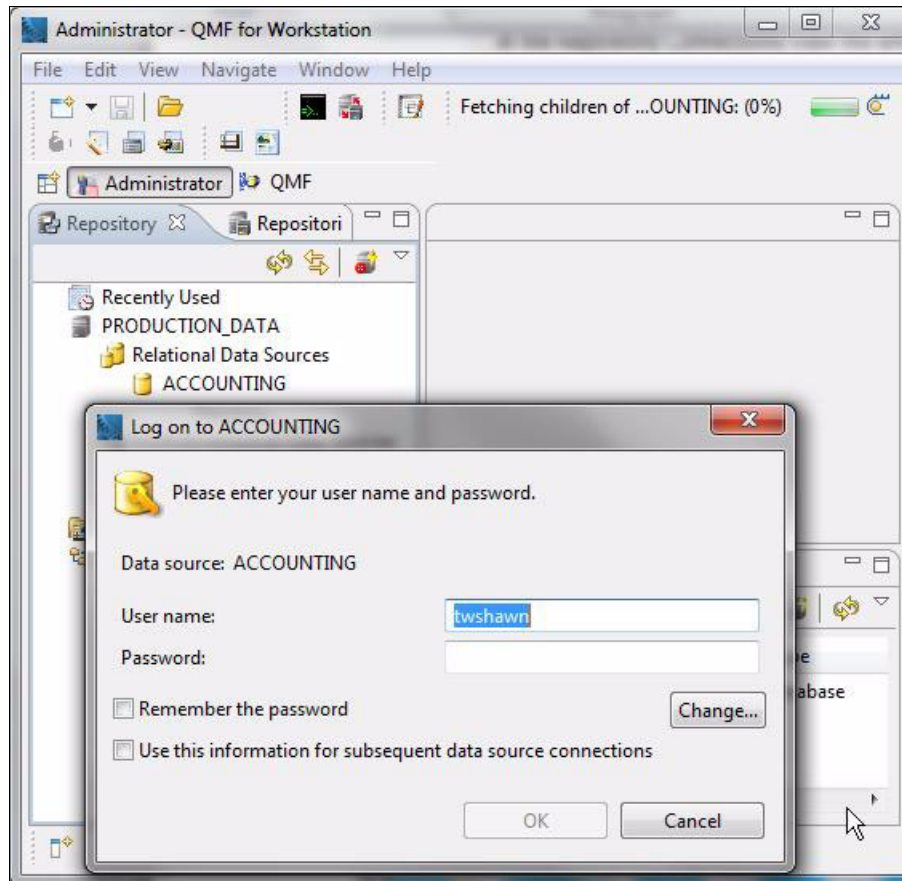


Figure 6-37 Logging into the ACCOUNTING data source on DB2 subsystem RS25Q91A

6.4.1 Populating user workspaces with content

As currently configured, QMF for Workstation is now set up for end user use in the QMF perspective, as shown in Figure 6-38. The **User** perspective is still empty (Figure 6-39).

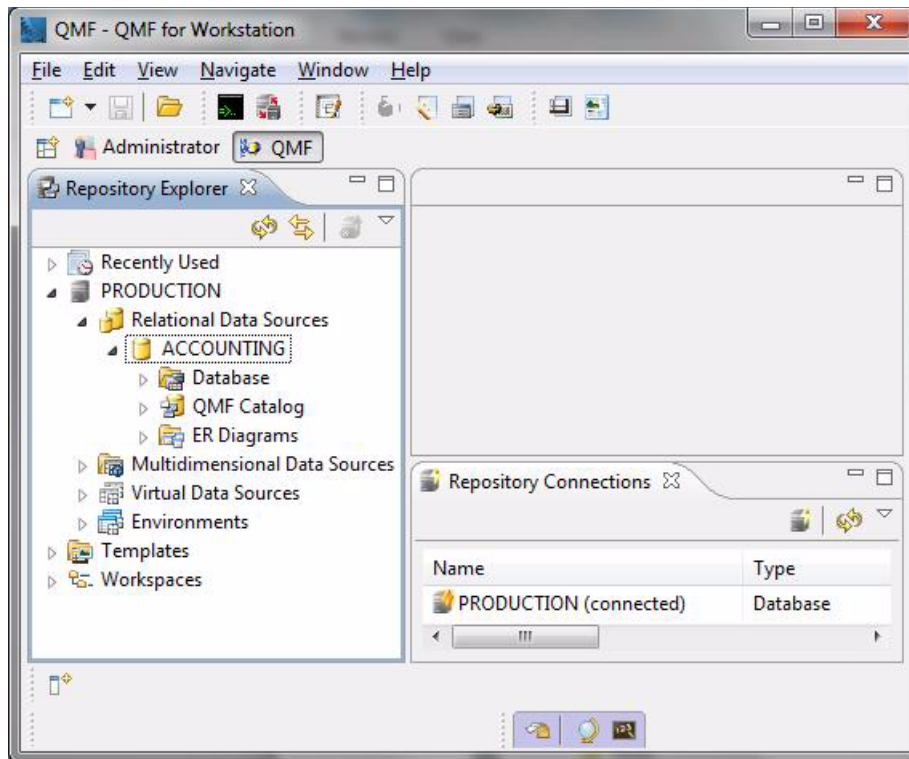


Figure 6-38 QMF perspective

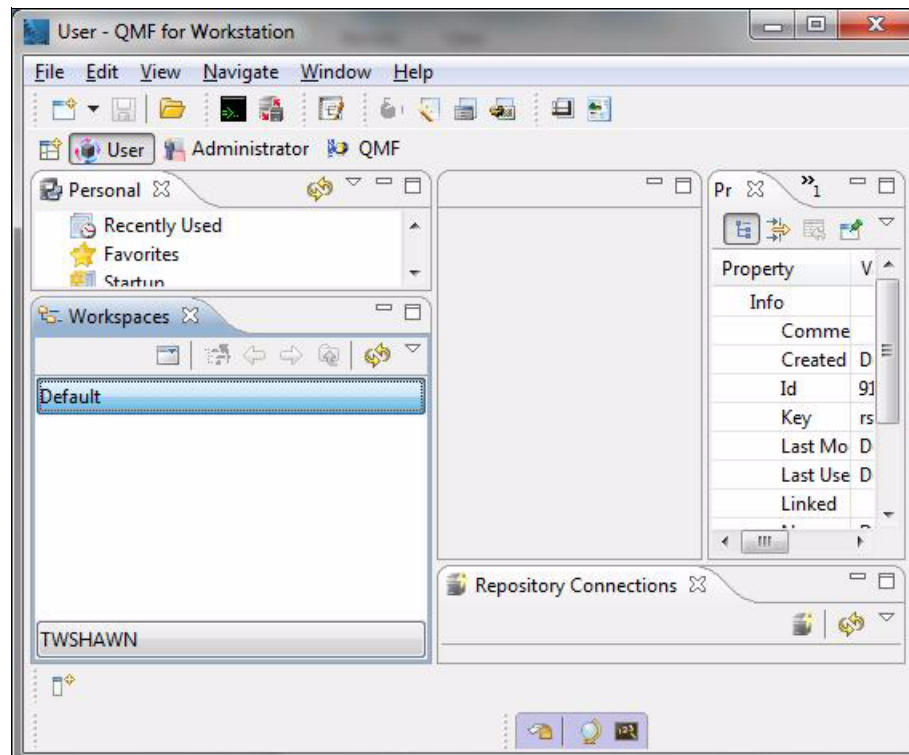


Figure 6-39 The initial (empty) User perspective

While the **QMF** perspective (which is the default perspective) has been set up to appear as close to QMF for Windows as the Eclipse interface allows, the **User** perspective is ideal for a user who is oriented more toward running a specific set of canned reports. The key to the **User** perspective is the workspace, which is a grouping of content that makes sense for the set of users that will access it. Until the administrator puts content into workspaces, there will be nothing of use in the **User** perspective. Think of a workspace as a Windows “folder”. The workspace can have objects such as queries, procedures, forms, tables, or any other tree branches or sub-branches from the Repository Explorer.

Because the **Enable Home Workspace Support** check box was checked when this repository was created (see Figure 6-16 on page 104), every time a new user logs in, a workspace is automatically created for that user. In the example shown in Figure 6-40, TWSHAWN has logged in and there is a workspace called TWSHAWN available for use.

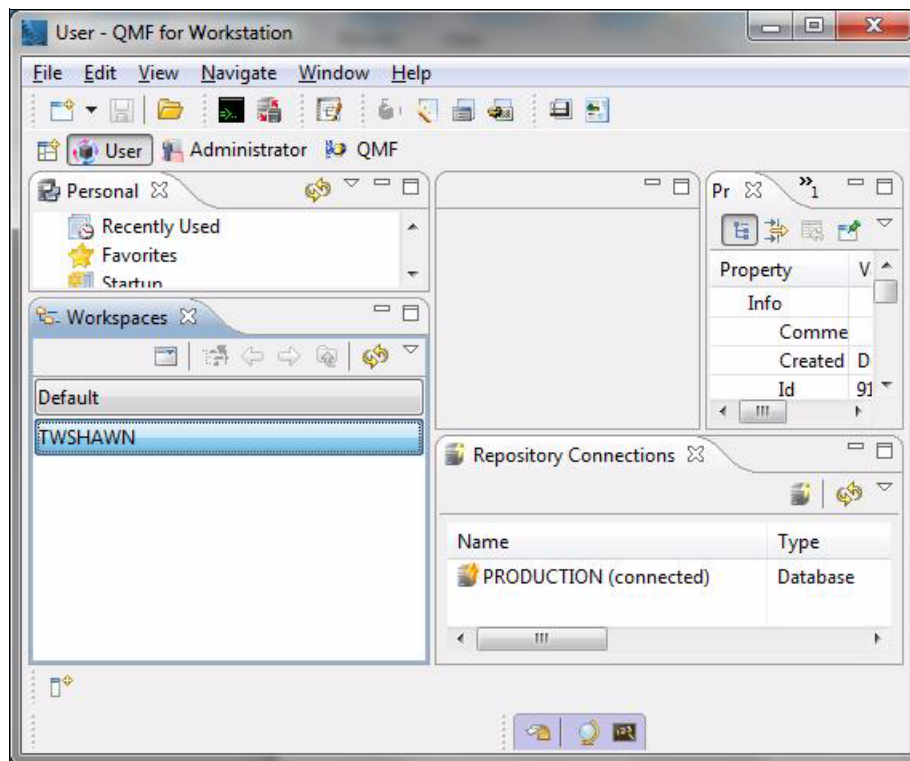


Figure 6-40 A workspace, defined and available

Now suppose that a second user, TWSHAW2, logs into the PRODUCTION_DATA repository connection (Figure 6-41 and Figure 6-42).

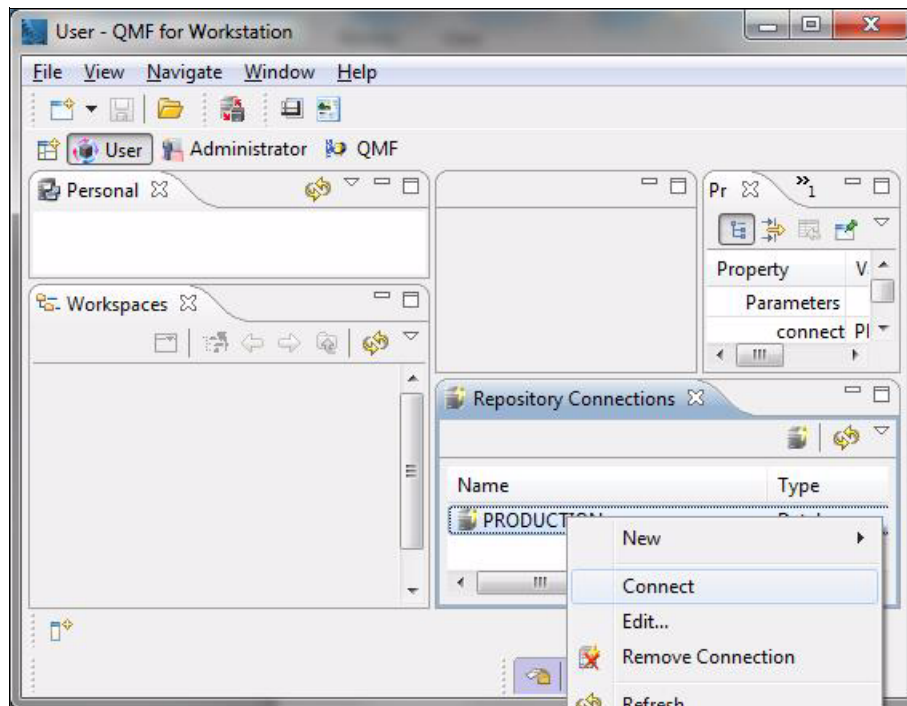


Figure 6-41 A second user, TWSHAW2, logs into the repository

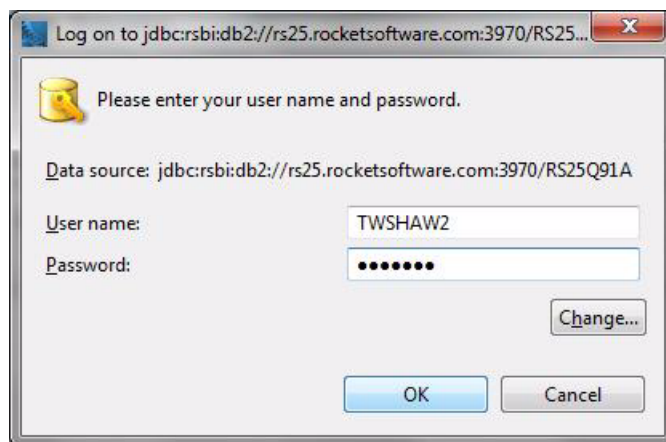


Figure 6-42 TWSHAW2 enters the user ID and password

Automatically, a workspace for TWSHAW2 is created (Figure 6-43).

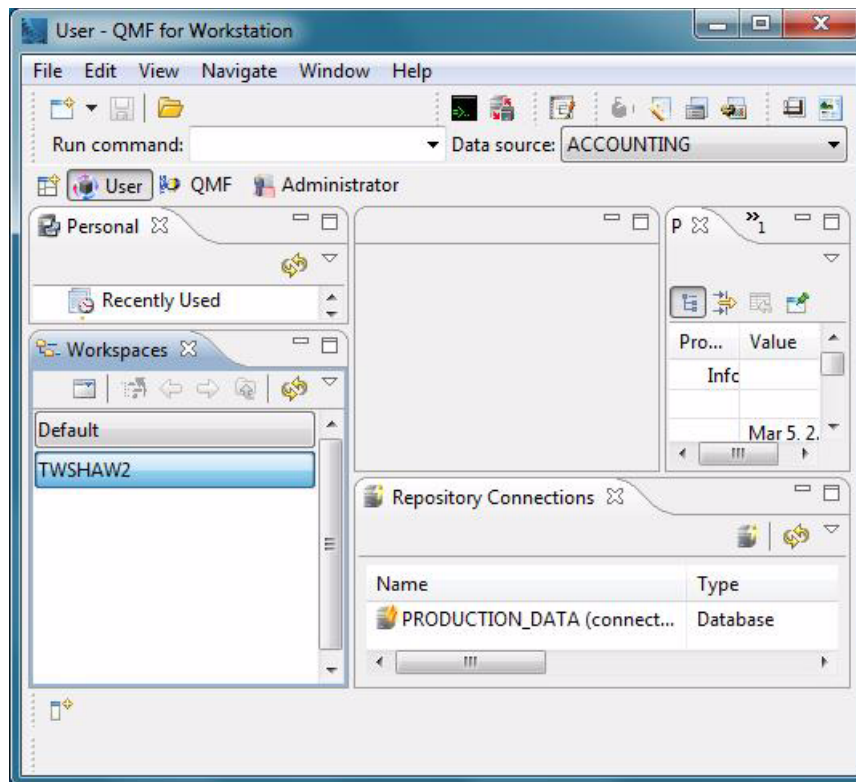


Figure 6-43 Automatic workspace generation

Tip: User TWSHAW2 no longer sees the home workspace for TWSHAWN because of the database-based security. By default, only the workspace owner and users that have SYSADM (or equivalent) authority can see a home workspace.

6.4.2 Creating workspace links to content in the Repository Explorer

Workspaces can be populated with links to content in the **Repository Explorer** tree. It can be done by drag-and-drop or copy and paste actions, as shown in Figure 6-44.

Links can be created to anything shown in the tree. For example, because the QMF catalog is shared between QMF for Workstation and QMF for TSO and CICS, you can create workspace links to any objects initially created in QMF for TSO and CICS. After the links are created, any further modifications to these objects that take place in QMF for Workstation are saved back to the catalog when the object is saved. Conversely, classic queries, procedures, and forms that are initially created in QMF for Workstation can be saved to the QMF catalog so that they are available for use in QMF for TSO and CICS. You could create a workspace link for these objects in this case as well.

With workspace links in place, objects in the QMF catalog can be accessed, edited, or run in either the QMF for Workstation/WebSphere or QMF for TSO and CICS environments.

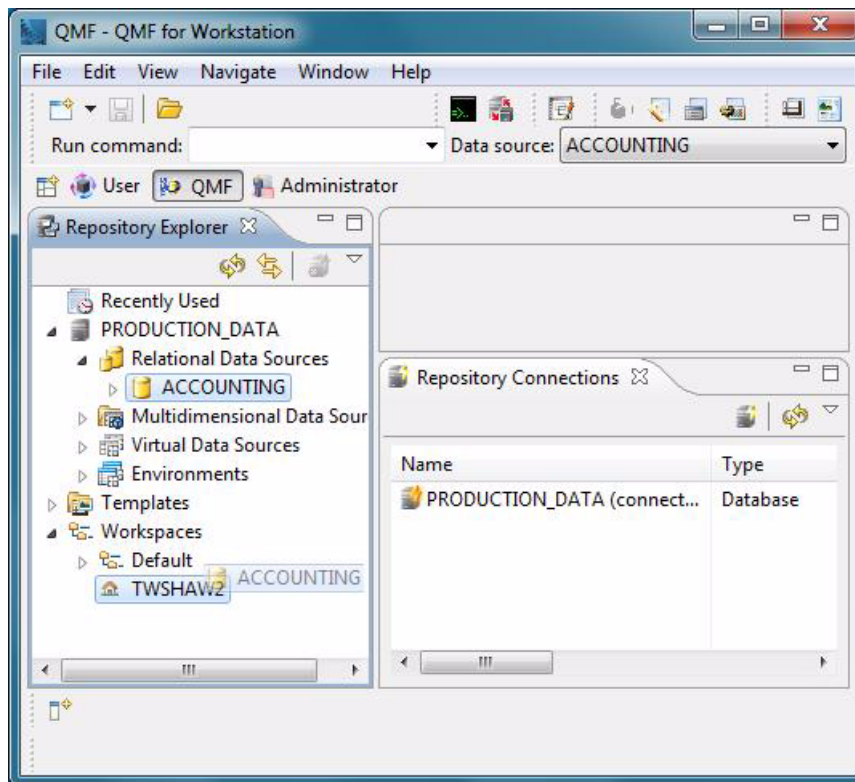


Figure 6-44 Populating the home workspace

New content can be created in the workspace by right-clicking and selecting the object type, as shown in Figure 6-45.

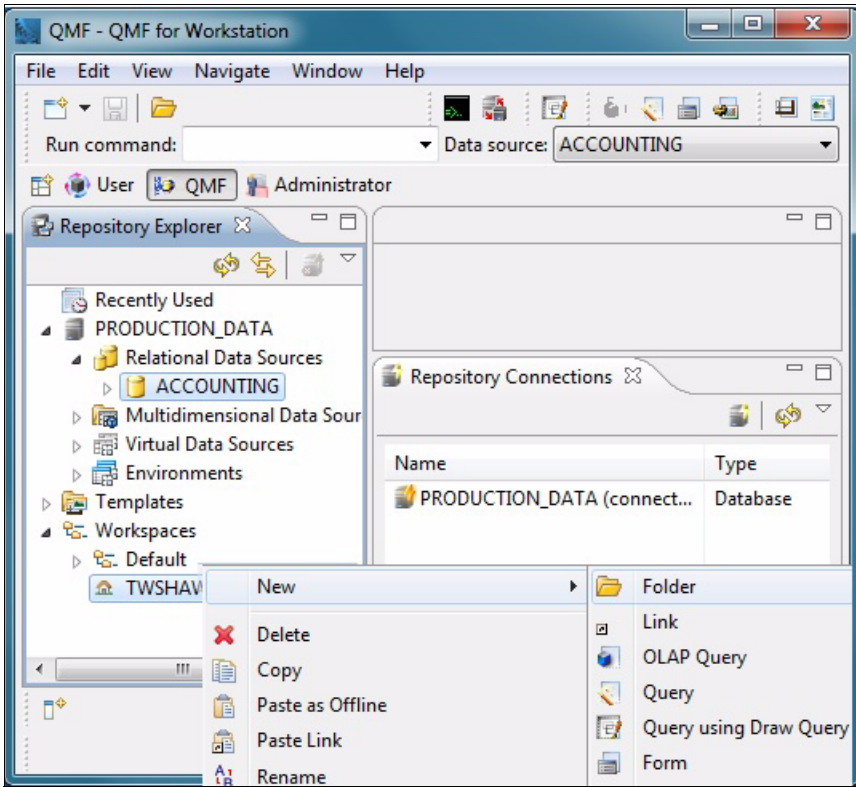


Figure 6-45 Creating new content in the workspace

Figure 6-46 shows the window that results when you select **New** → **Folder**.

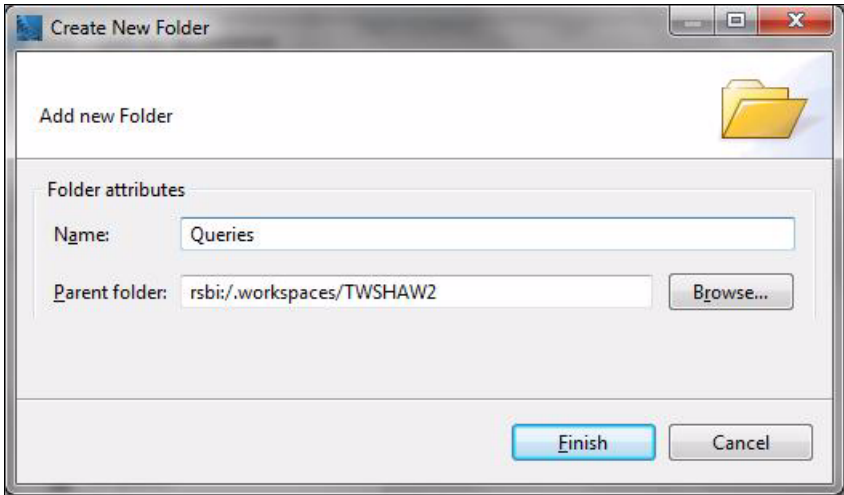


Figure 6-46 Creating a new folder in the workspace

Workspace folders can contain subfolders with a variety of content: queries, forms, procedures, reports, dashboards, links to tables, links to table schemas, links to relational data sources, or any other object or grouping of objects in the **Repository Explorer** tree. Organization is left up to the end user.

The **User** perspective shows a simplified view of the content (Figure 6-47).

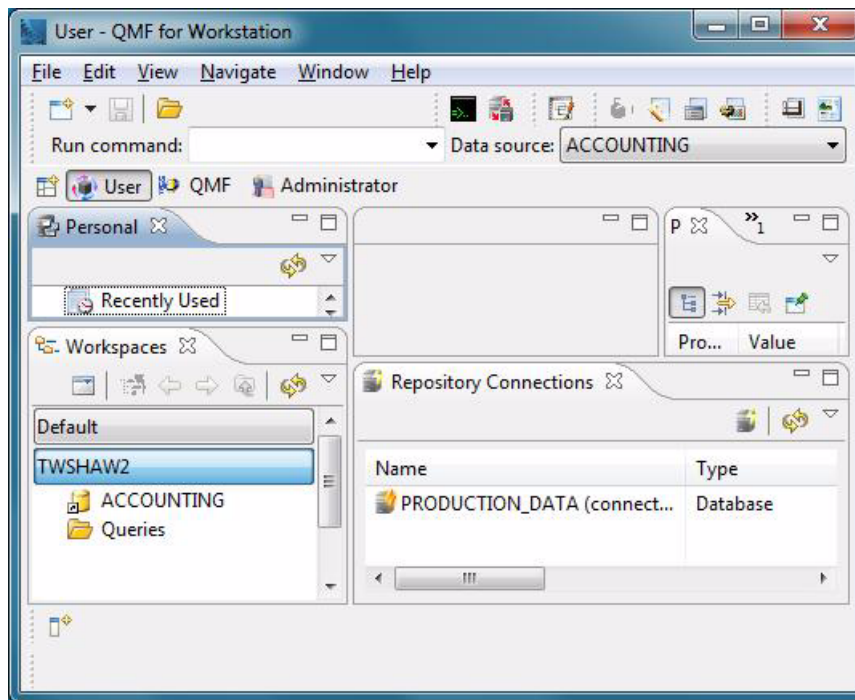


Figure 6-47 Populated User perspective view of content



Defining virtual data sources to reduce complexity for business users

This chapter provides an overview of the QMF metadata layer, a powerful set of capabilities that allows QMF users to readily access enterprise data without the need to understand the complexities of the associated database schemas. The following topics are covered:

- ▶ Benefits of the QMF metadata layer
- ▶ Reducing data complexity with virtual tables, including steps on how to create a virtual table from a QMF query as well as how to use a virtual table to simplify the underlying table schema for users
- ▶ Creating a virtual data source that federates data from multiple sources

7.1 Benefits of the QMF metadata layer

The QMF metadata facilities help to remove complexity for end users by giving you the following capabilities:

- ▶ Present a simplified data schema to end users and content developers by representing multiple physical database tables as a single virtual table.
- ▶ Rename physical database tables and columns, as well as remove columns that are not required by the target user base, thus enhancing the underlying database schemas to make them more user-friendly.
- ▶ Use a single, virtual database to provide a means of federating data stored across multiple data sources.
- ▶ Create ER diagrams and display these relationships to users when they work with the associated tables.

The first three capabilities listed here are described in detail in the following topics. For more information about working with ER diagrams, see *Installing and Managing QMF for Workstation and QMF for WebSphere* at the following website:

<http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp>

7.2 Reducing data complexity with virtual tables

Typical enterprise databases consist of a large and complex array of tables and columns. This underlying database schema is designed for efficiency, both in terms of data storage and retrieval. However, this efficiency can come at the expense of clarity for users who need to access and navigate the data. Most enterprise users who are tasked with developing queries, reports, and dashboards are not fully versed in the nuances of database schemas, nor should they have to be. Rather, users ought to be presented with a simplified data schema, tailored to the business view they use and the requirements they have for it.

The QMF metadata layer meets this need by allowing QMF administrators and power users to represent underlying database schemas using a simplified, virtual view of the data that is easier to understand and work with. It not only allows a broader base of users to access the data and to build content against it, it also makes the process much more efficient and productive because users are working against a simplified data model.

For example, suppose that Spiffy Insurance market analysts require access to data that totals insurance premiums across each of the company's business units, by date and also by premium type. This information resides across three tables in Spiffy's data warehouse as shown in Table 7-1.

Table 7-1 Tables containing insurance premium information

Table name	Content
IIWREPORT.INTERNAL_ORGANIZATION	A 28-column table providing business unit information
IIWREPORT.SOLVENCY_II_CP58_F1_LIFE_REVENUE_ANALYSIS_PREMIUMS_FACT	An 11-column table containing premium totals by type
IIWREPORT.CALENDAR_YEAR	A 6-column table containing reporting period information

To build queries, reports, and dashboards directly against this data, business users and content developers would need to understand the data warehouse schema in enough detail to know which three tables to access. They would then have to locate them among the hundreds of tables in the data warehouse, add them to the query, and join them across the relevant columns (if not already configured). The outcome of such a multistep operation would be a query diagram such as that shown in Figure 7-1. Notice that the columns of interest to the business user are represented in **bold**.

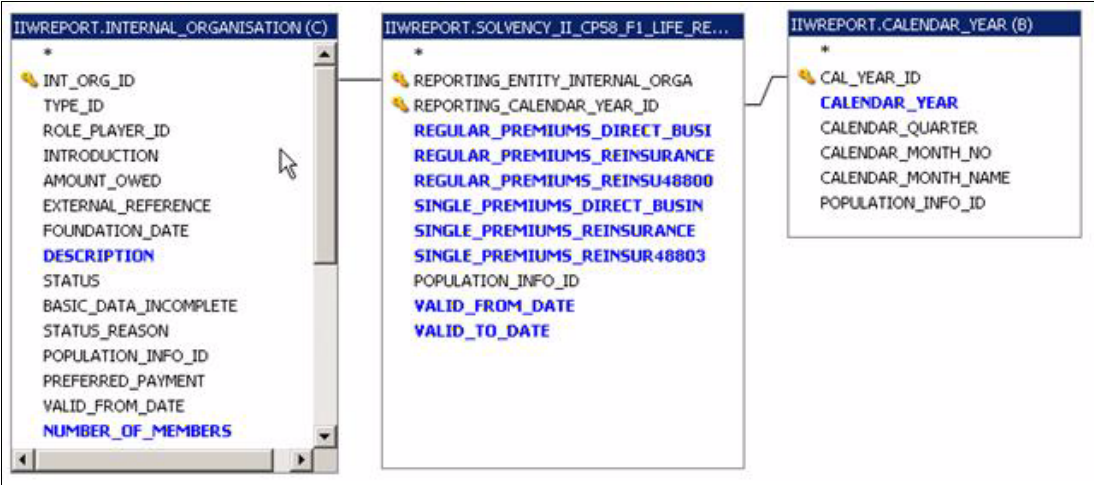


Figure 7-1 Underlying database schema for insurance premium data at Spiffy

In cases such as these, the process of developing a query, report, or dashboard is clearly hampered by the complexities of the underlying database schema. The QMF metadata layer is expressly designed to alleviate this problem. Using the metadata layer, administrators have the ability to represent schemas such as that in Figure 7-1 as one or more simplified, virtual tables.

For example, one way of simplifying the example schema shown with virtual tables would be to retain the three tables but remove all but the columns of interest, while also renaming the tables and columns so that they are more easily understood and remembered. Alternatively, two of the three tables could be represented as a single table, while the other remains “as is,” but with edits to both the table and column names. In our case, we represent the three tables as a single virtual table, as outlined in Figure 7-2.

Premiums By Year (A)
*
Description
Number Of Members
Regular Direct
Regular Reinsurance
Regular Reinsu48800
Single Direct
Single Reinsurance
Single Reinsur48803
Valid From Date
Valid To Date
Calendar Year

Figure 7-2 A virtual table providing a simplified view of Spiffy's insurance premium revenue

This virtual table contains the 11 columns of interest to business users and shields them from the underlying complexities of the real tables. In addition, the column names have been simplified and the table name is more descriptive. All the users know is that they are looking at insurance premiums by year. They are not concerned with where the data comes from.

Tip: The QMF metadata facilities also provide a means of shielding query, report, and dashboard content from the effects of database schema changes. Considering the foregoing example, the layout of the three underlying tables can be arbitrarily changed by database administrators and architects without impacting either the authored content or those who use it. As long as the format of the virtual table in Figure 7-2 on page 135 remains the same, end users and authored content remain unaffected.

7.2.1 Creating a virtual table from a QMF query

In the previous topic, we explained how you can create a single virtual table from one or more underlying tables. In this topic, we will show you how to use a QMF query as a starting point to create the virtual table.

Creating a virtual table from a QMF query involves these steps:

1. Create a QMF query that joins the physical database tables.
2. Create a virtual database within which the virtual table will reside.
3. Add the QMF query as a virtual table.
4. Rename the virtual table columns to suit business requirements.

We cover each step of the process in detail in the following topics.

Creating a QMF query that joins the physical database tables

The QMF query outlined in Figure 7-3 was authored using the methods covered in Chapter 9, “Getting to the data you need: Query methods” on page 191. Notice that this query can also include calculated columns that compute values derived from other column values. Any or all calculated columns in the query can be added to the virtual table that will be built from the query. Thus, a virtual table can contain content that is both real (from existing tables) and derived (based upon calculations from the data). The benefit is that end users do not need to know which content is real versus derived, because the virtual table masks this complexity for them.

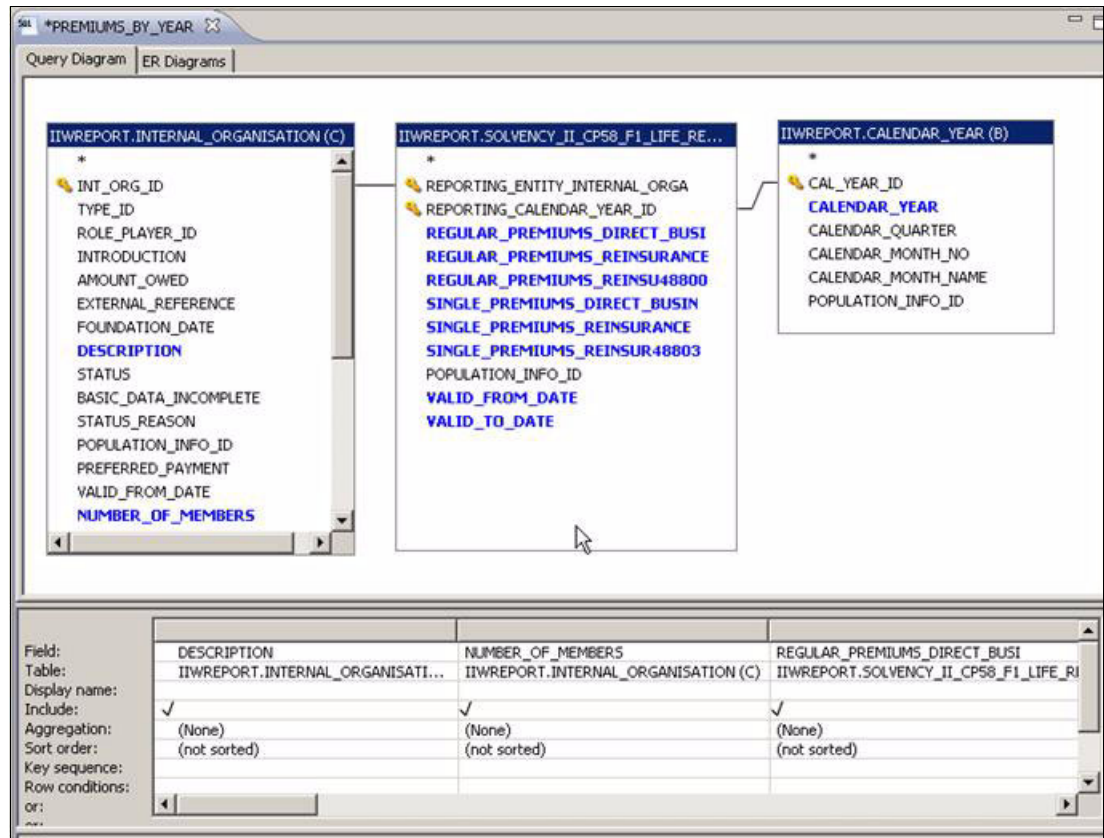


Figure 7-3 QMF query showing Spiffy Insurance premiums by year

When the query is finished, select **File** → **Save** to save the query to the QMF repository.

Creating a virtual database within which the virtual table will reside

After creating the query, follow these steps to create a new virtual data source to hold the virtual table:

1. Select **File** → **New** → **Other**, as shown in Figure 7-4.

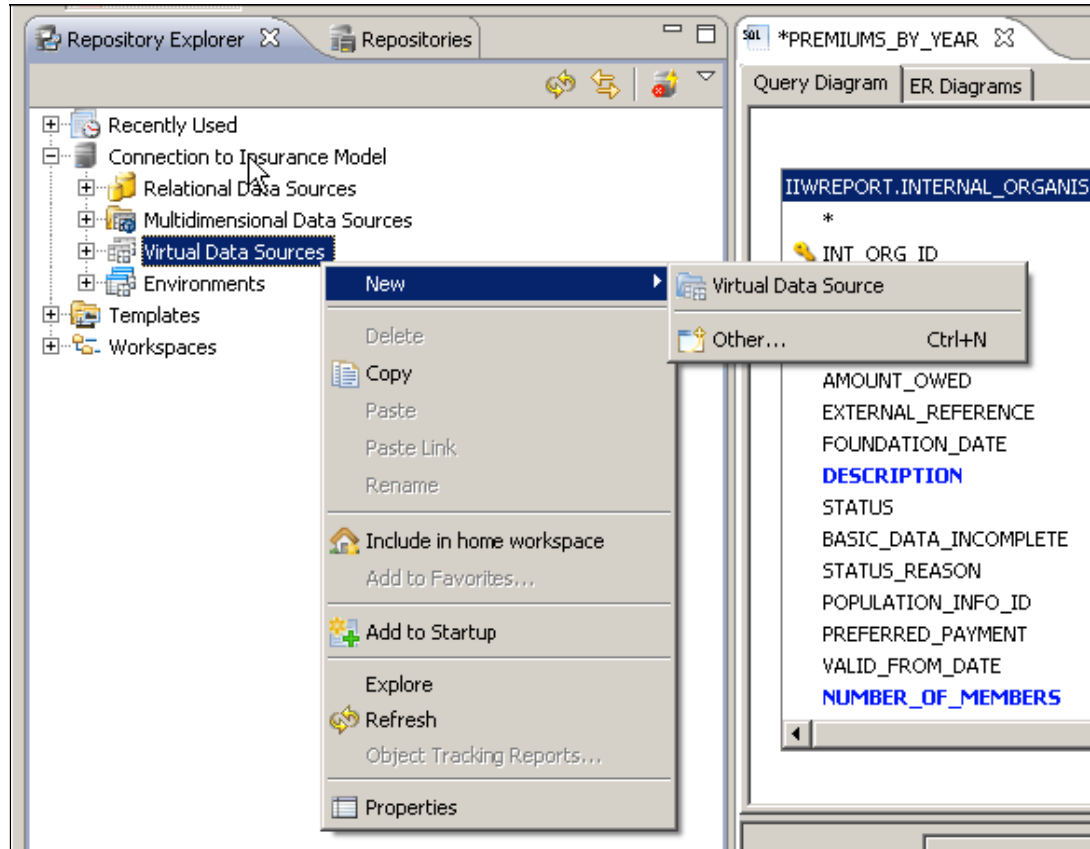


Figure 7-4 Creating a new virtual data source

2. Type the name of the virtual data source and click **Finish**. The virtual data source can be assigned any given name. Choose something that describes the content, as shown in the example in Figure 7-5.

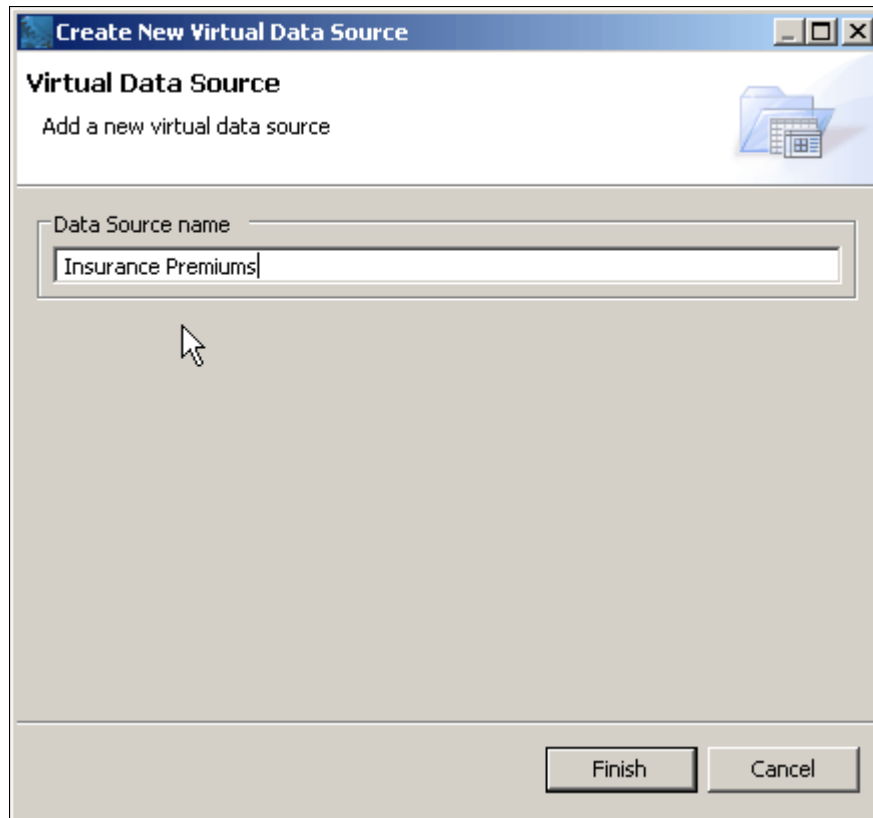


Figure 7-5 Naming a new virtual data source

Tip: The QMF SAVE DATA command (SAVE DATA AS name) has long been a common use of taking output from a QMF query and using it as input to a step in another QMF process. This function is still available, but now you can use a query as “virtual” input and save the overhead of storing data in a table before using it further.

Adding the QMF query as a virtual table

After you provide a name for the new virtual data source, the virtual data source editor is displayed, which allows you to populate the database with physical tables drawn from one or more databases or from QMF queries that have been saved to the QMF repository.

The virtual data source editor lists data sources and saved queries on the left and the contents of the virtual data source on the right. Resources can be added to the data source by clicking them in the left pane and clicking the **Copy Selected** icon, which is highlighted in Figure 7-6.

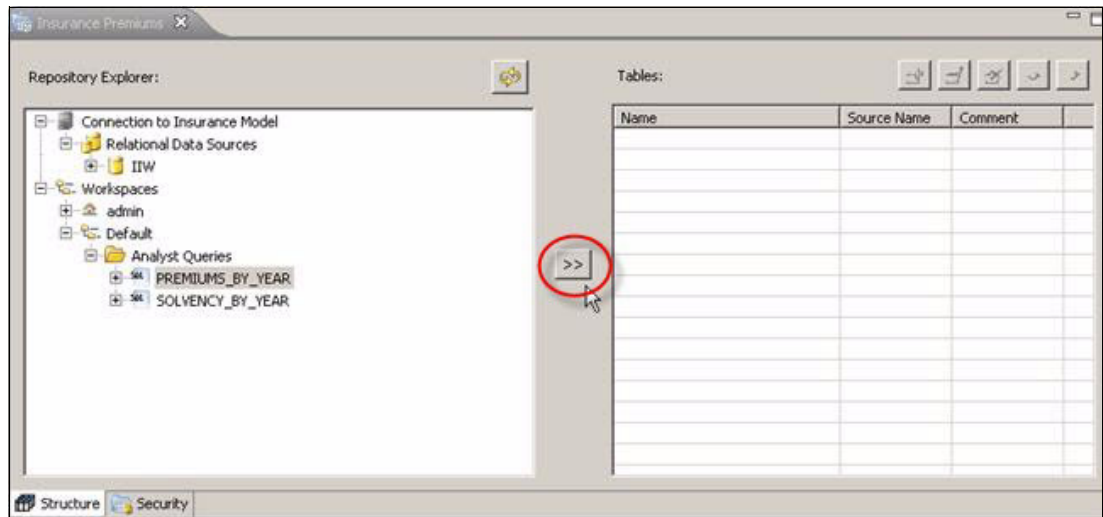


Figure 7-6 Adding a query to a virtual data source

In this case, the `PREMIUMS_BY_YEAR` query will be added to the virtual data source. When added, the table appears in the right pane of the editor, signifying that it is now represented as a virtual table in the virtual data source. Any number of queries or physical tables can be added using this same mechanism. Notice that this action does not actually move any data. The virtual data source is an abstract layer only; all data remains within the underlying data sources. When users access the virtual table, QMF parses the SQL and sends it to the respective underlying data sources.

Renaming the columns of the virtual table to suit business requirements

When a table has been added, QMF provides an opportunity to rename the table or columns. To do this, right-click the table or column name and select **Rename** from the context menu. The name can then be modified, as shown in Figure 7-7. Notice that the pane on the right retains the original name for reference. Columns can also be removed from the virtual table if they are deemed not relevant for the target audience.

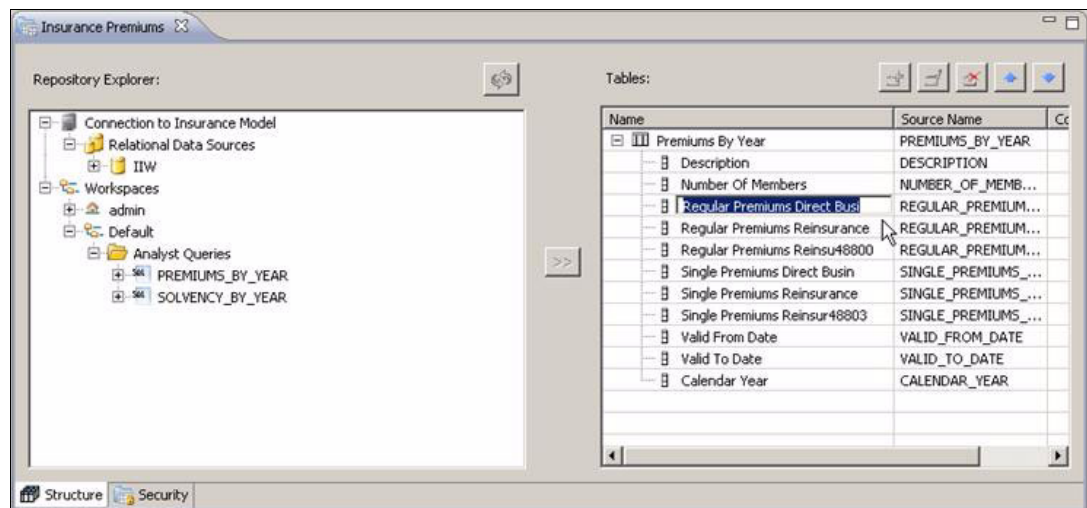


Figure 7-7 Renaming virtual table columns

7.2.2 Using a virtual table to simplify a table schema

In many cases, business data is organized into manageable collections of tables, but the barrier to productivity lies with the table and column names and the number of columns in the table that are completely irrelevant for enterprise users and content developers. In plain terms, there are simply too many items to choose from.

The QMF metadata layer can also be used to simplify the physical database schemas. Unlike the first example we used in 7.2, “Reducing data complexity with virtual tables” on page 134 and 7.3, “Creating a virtual data source that federates data from multiple sources” on page 143, where multiple tables are represented by a single virtual table, this approach uses a virtual table for each physical table, but the virtual table contains meaningful names and relevant columns.

For example, suppose that Spiffy Insurance wants to provide access to data stored in their “loss profile” table. This data is also associated with a reporting period table and an organization table. The latter two tables are generically useful in other queries, so for this reason it is better for Spiffy to make them available as independent entities rather than represent all three as a single virtual table. However, the downside to this approach is the fundamental complexity and number of the column names in the tables, shown in Figure 7-8.

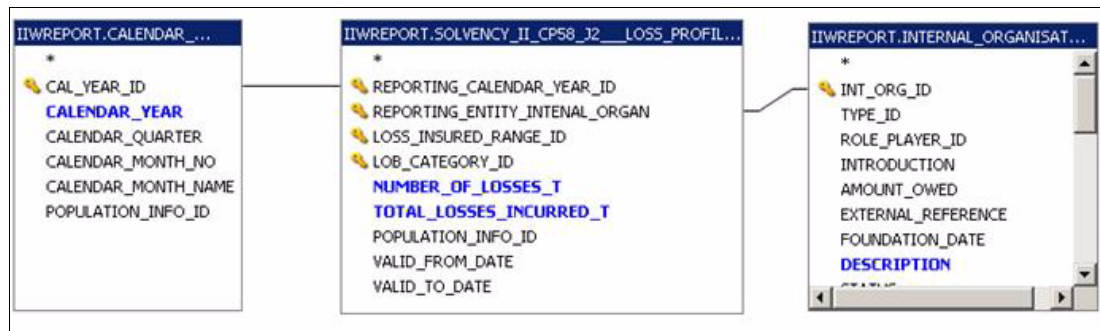


Figure 7-8 Underlying schema for insurance loss data at Spiffy

To simplify the underlying schema, the three tables can be added to the existing “Insurance Premiums” virtual data source. When added, superfluous columns can be removed and the remaining columns can be renamed, as outlined in the right pane in Figure 7-9.

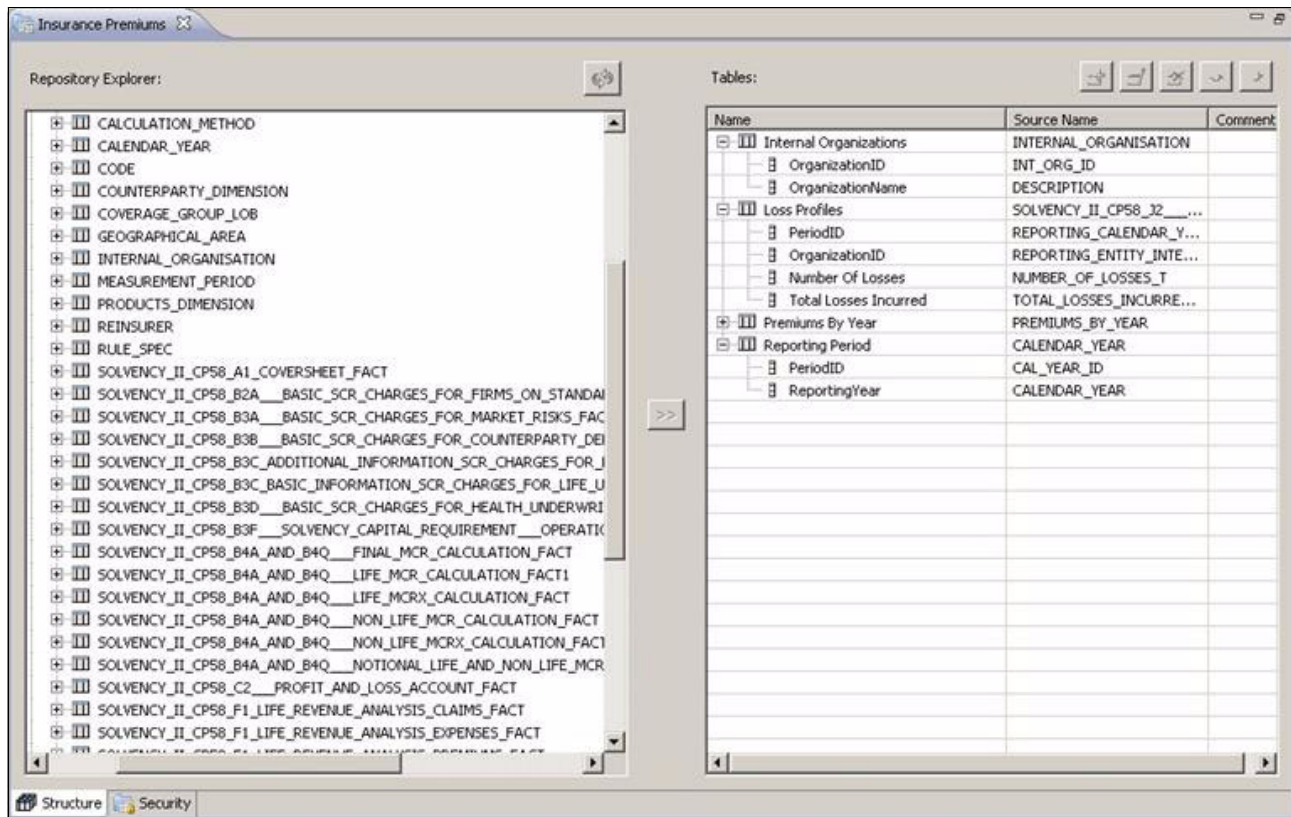


Figure 7-9 Transforming and simplifying database table schemas

With these changes in place, developing the query previously shown in Figure 7-8 on page 141 is now much simpler for end users, as shown here in Figure 7-10.

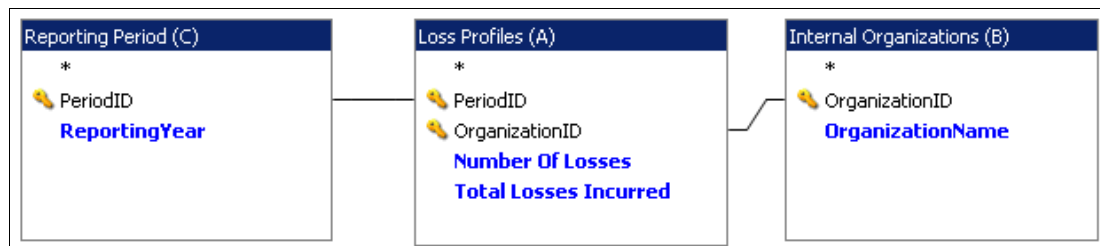


Figure 7-10 Simplified schema for insurance loss data at Spiffy

As before, when running this query against these virtual tables, QMF translates the user's SQL statements into a format that matches the actual database schema before sending it to the underlying data sources.

7.3 Creating a virtual data source that federates data from multiple sources

Enterprise data stores typically evolve over time. New systems are added alongside existing infrastructure to meet demand for new capabilities or to accommodate merges or corporate restructures. Inevitably, this results in data scattered across a number of distinct databases. For example, Spiffy Insurance stores its business and customer data primarily in DB2 for z/OS. This data includes sales by insurance product, region, date, and customer. However, as the result of an acquisition, Spiffy inherited a product administration system that is built within an Oracle database. Recognizing the value of the product administration system, Spiffy chose to retain it. The business database in DB2 refers to product IDs (PIDs) only. But product details by PID are recorded, maintained, and refined in Oracle.

Sales reports and dashboards at Spiffy must present business data on a PID basis. However, PID information is less readily understood by the broader base of enterprise users at Spiffy, and the company wants to report sales metrics against the insurance product names and categories instead. Spiffy also has a requirement to grant sales staff and analysts access to the last 30 days of transactions at the customer account level. This information is available in yet another system (IMS) and would form a natural part of a customer sales analysis dashboard. Now we are faced with a 3-way data dilemma.

Such an environment can make it challenging to draw information into a single query, report, or dashboard. However, because QMF virtual data sources act as a single, physical database, this feature can be used to create a single data source that federates data from DB2 for z/OS, Oracle, and IMS. Thus, all tables contained within the virtual data source, no matter where they physically reside, can be joined together.

To give end users a unified view of the tables and hide the distributed and complex nature of the underlying data schemas, Spiffy's IT staff could do the following procedure:

1. Create a virtual data source called "CustomerActivity"
2. Add tables from DB2 for z/OS, Oracle, and IMS as necessary

Figure 7-11 shows how the virtual data source editor would look in this case.

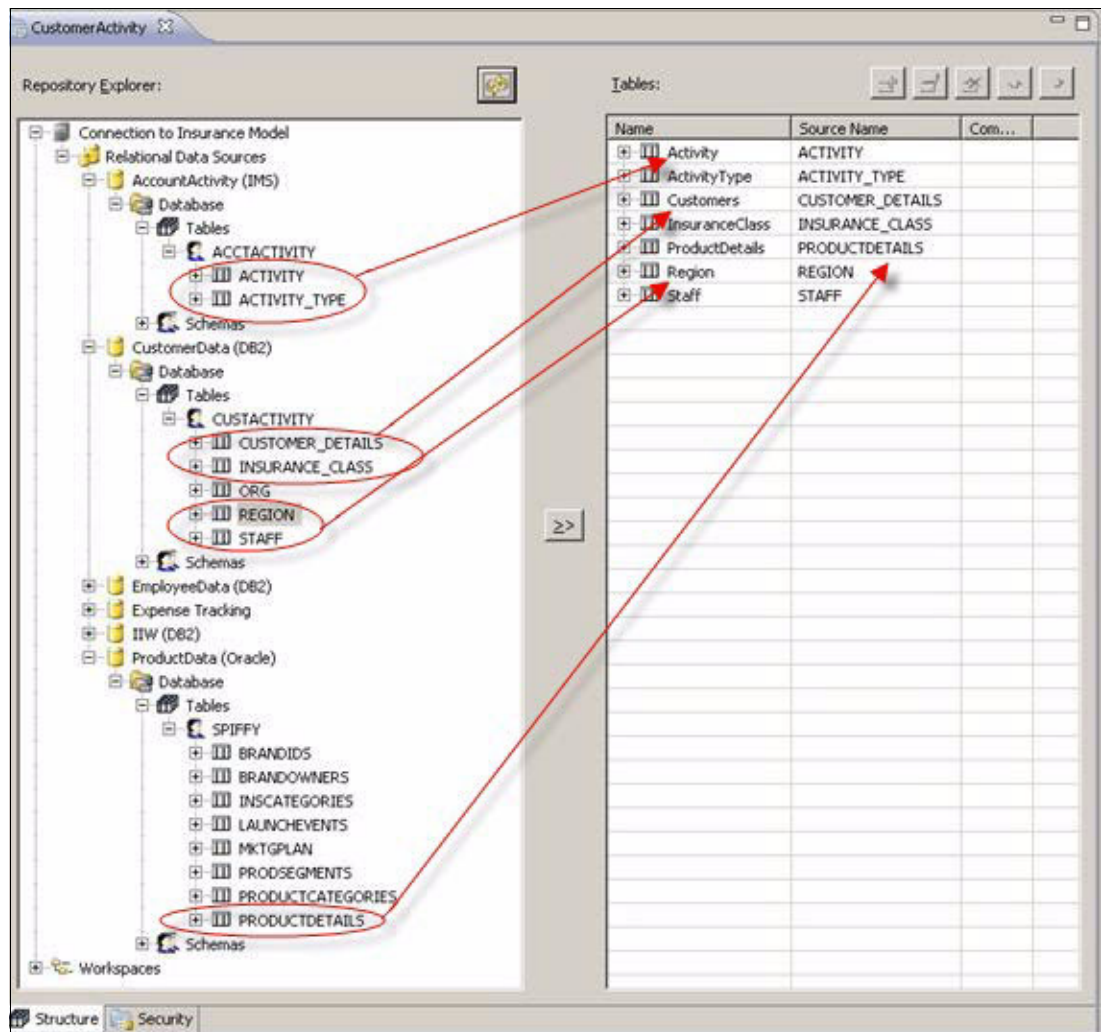


Figure 7-11 Representing multiple data sources as a single virtual data source

Spiffy's business end users and content developers are now free to use the "CustomerActivity" virtual data source to develop dashboards and reports that provide an overview of sales by customer and by product name, with real-time activity details on a per-account basis. Because all of this information appears to reside in a single database, it can be queried as such. Internally, QMF routes the component parts of the queries to DB2, Oracle, and IMS as necessary, joining the data as it is returned from each distinct server.



Configuring security

As the field of business analytics becomes more mission critical, the need increases to provide greater security for data assets. When a repository is created, as explained in Chapter 6, “Configuring access to data sources and populating user workspaces” on page 89, a decision must be made as to how might want to secure the repository content.

This chapter delves more deeply into the value and workings of the various security options in QMF. The following topics are covered:

- ▶ Security strategies
- ▶ Minimizing the number of logins required
- ▶ Use of security in dashboards

8.1 Security strategies

When you create a repository, you can specify database-based, internal, LDAP, or no security, as shown in Figure 8-1. Our recommended methodology is to use database-based security whenever possible.

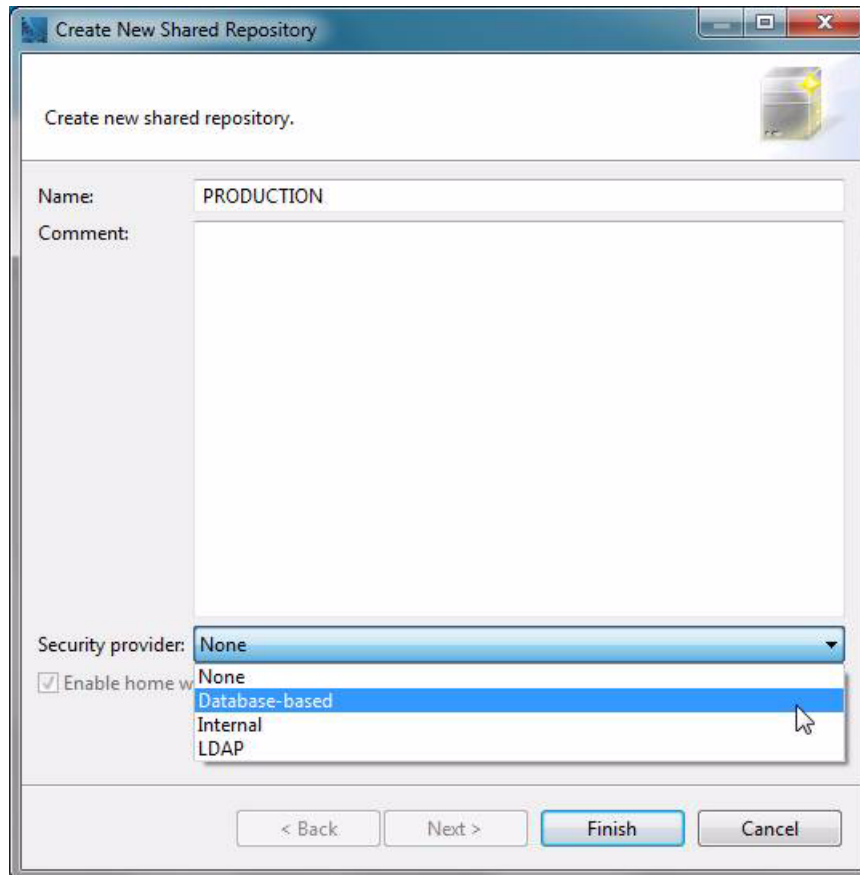


Figure 8-1 Security options available in QMF

The security provider setting gives you the following capabilities:

- ▶ Govern access to the repository.
- ▶ Assign permissions on objects, folders, and branches in the Repository Explorer tree.
- ▶ Put users into groups to streamline granting permissions.
- ▶ Create dashboards and reports that expose content based on the ID of the user running the report.

The security provider setting cannot be changed after the shared repository is created, so you need to carefully consider this choice. The following subtopics explain each option in detail.

8.1.1 Database-based security

As stated previously, the **Database-based** security provider setting is strongly recommended for QMF environments. This setting takes the user ID entered at the repository connection prompt when the initial connection to the repository storage database is made. This ID, which has already been validated by the repository storage database, is then used as the ID when evaluating permissions to access the repository content. This way, a single logon provides access to both the repository storage and the repository.

Shown in Figure 8-2, the **Enable home workspaces support** option causes a new workspace to automatically be created when each user first logs on. The name of the workspace is the same as the user ID and no permissions on the workspace are set to None. Only the user who creates (owns) the workspace and the administrator can have access.

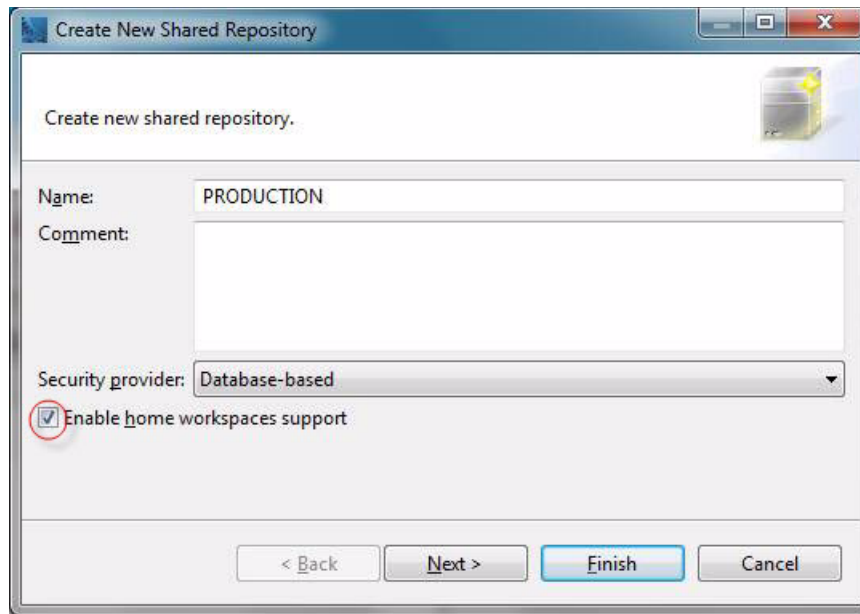


Figure 8-2 Creating a new shared repository

Adding users

After entering a name in the dialog shown in Figure 8-2 on page 147, click **Next**. The **Import user names** from QMF Catalog option (Figure 8-3 here) pulls all of the user ID values that have been defined in QMF resource limits groups and establishes them as users so that repository assets can be governed. Click **Finish** to close the dialog.

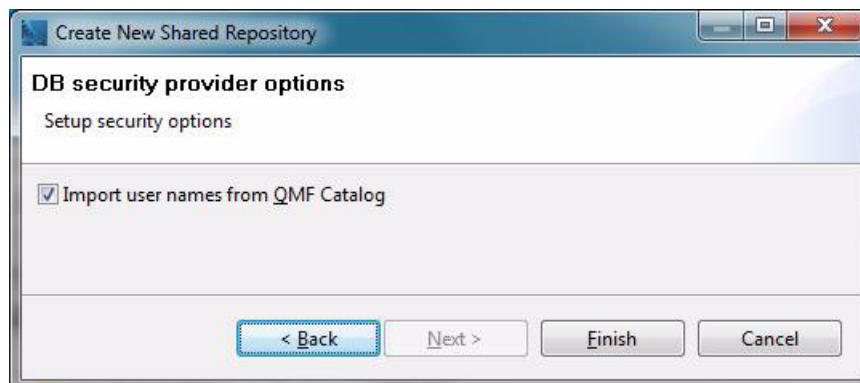


Figure 8-3 Importing user names from the QMF catalog

The user ID that you use to create the repository will be the administrator (Figure 8-4).

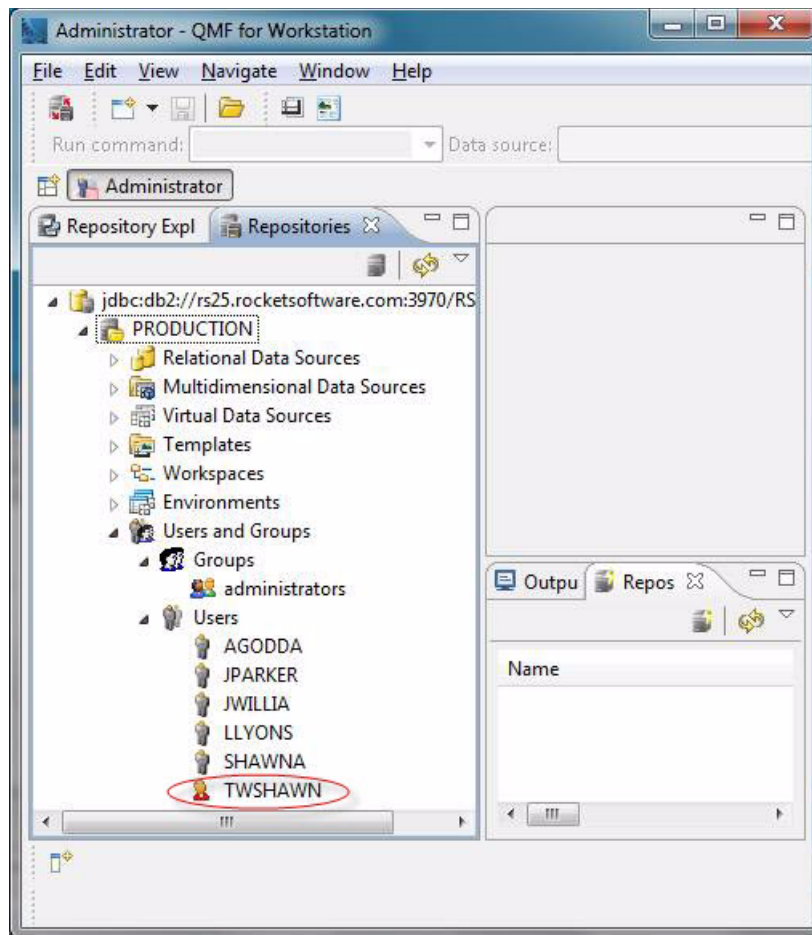


Figure 8-4 Establishing the administrator for the repository

Be aware that, when you import users, only one group is created, the “administrators” group. It is a one-time loading of user IDs. If new users are added to the QMF resource limits groups, they will not automatically become new repository users as a result. However, new users will automatically become repository users as a result of their first login to the repository. As shown in Figure 8-5, the user TWSHAW2 does not appear, as this user has not yet logged in.

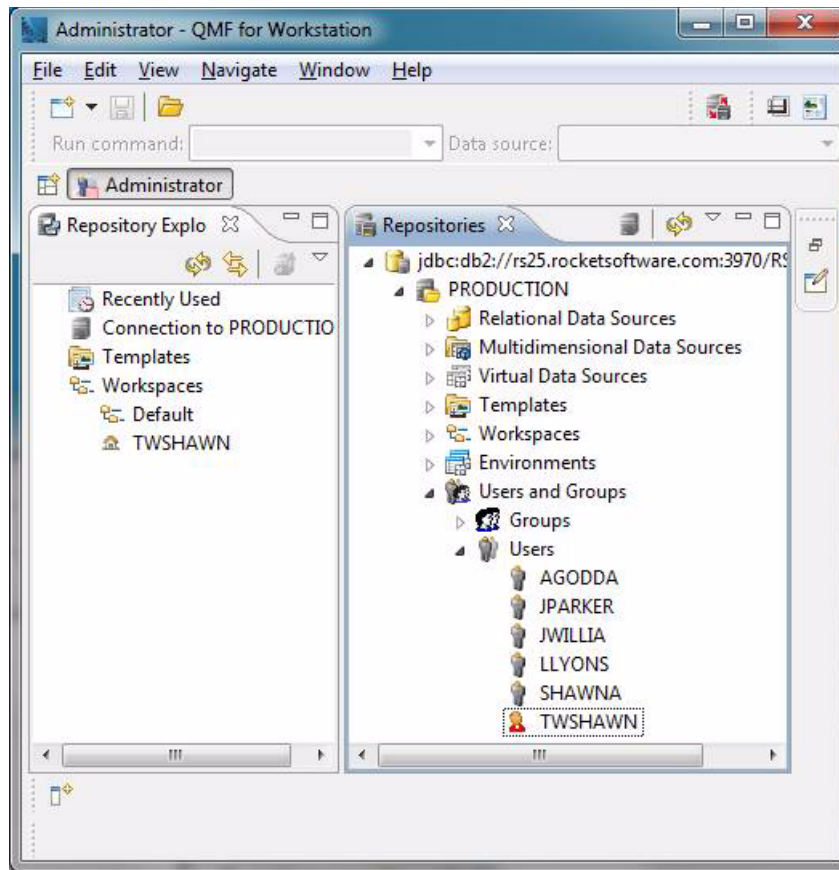


Figure 8-5 List of users before new user TWSHAW2 logs in

After the user TWSHAW2 logs in (Figure 8-6), there is a new user and a workspace.

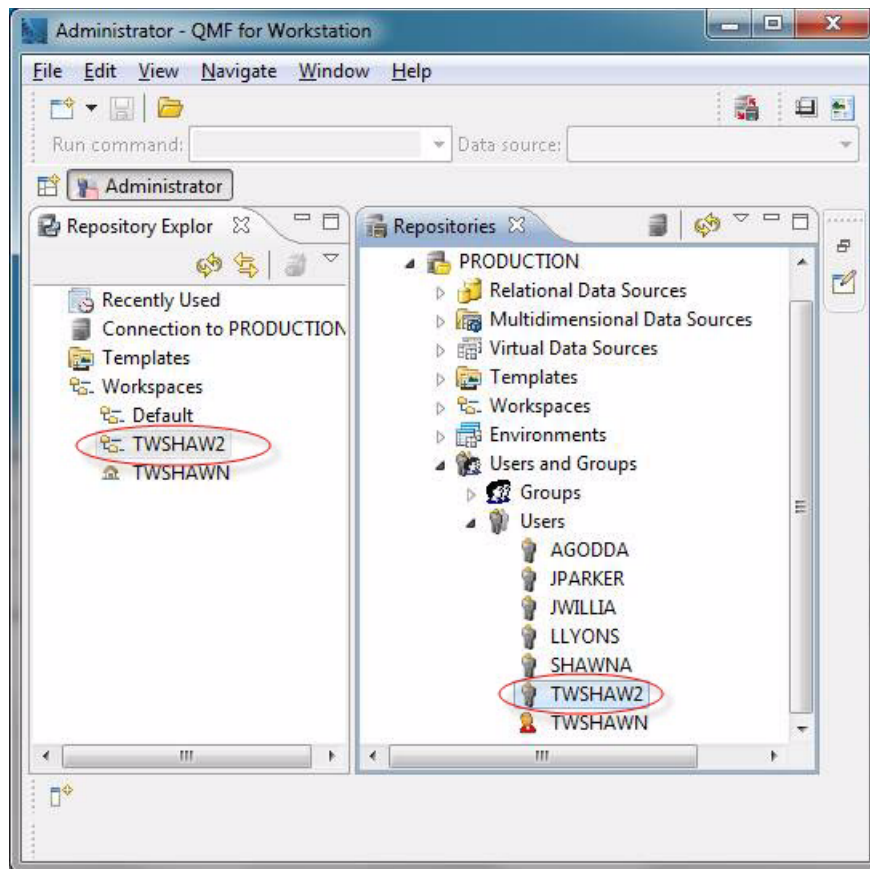


Figure 8-6 User ID and workspace for TWSHAW2, shown after TWSHAW2 logs in

The default permissions on these automatically generated home workspaces are set so that only the user and any administrators can see the workspace. Above TWSHAWN, the administrator is logged in (SHAWNA) and can see both TWSHAWN and TWSHAW2.

As shown in Figure 8-7, TWSHAW2 is logged in and can only see the TWSHAW2 workspace.

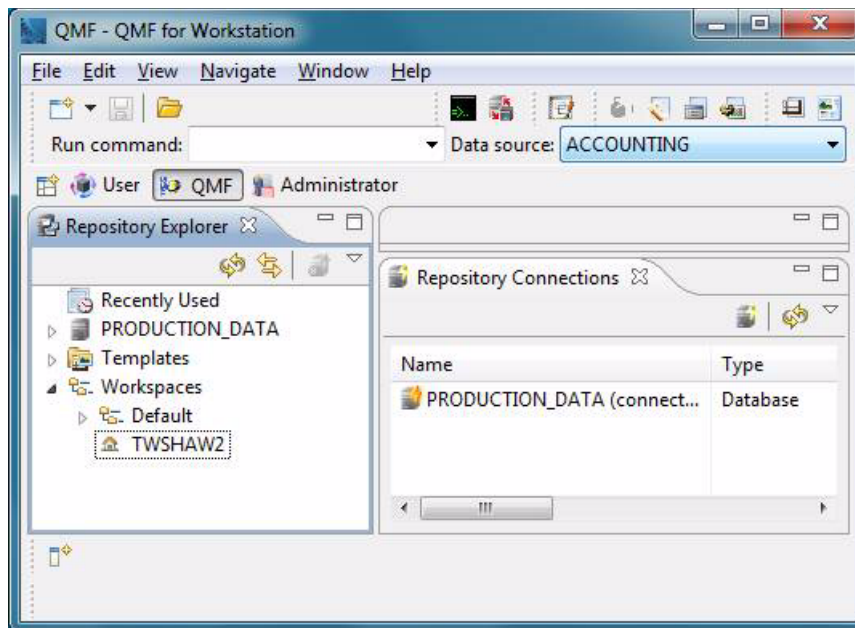


Figure 8-7 TWSHAW2 can only see his own workspace

By right-clicking the TWSHAW2 workspace and selecting **Properties**, we see that there are no permissions granted for other users (Figure 8-8).

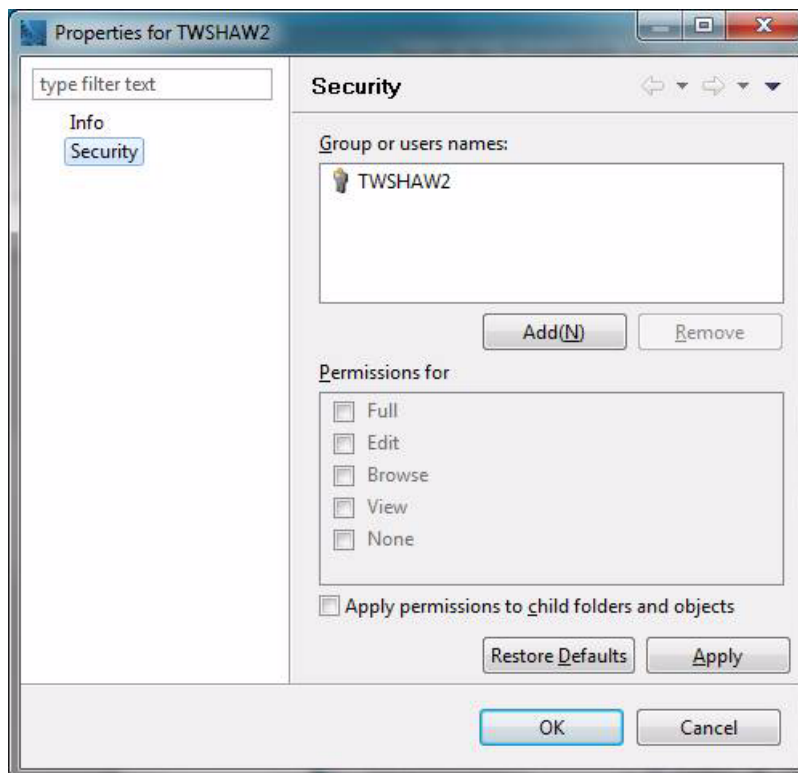


Figure 8-8 Checking the permissions on the TWSHAW2 workspace

To summarize, the configuration shown in Figure 8-8 on page 151 performs these functions:

- ▶ Imports existing users from the QMF catalog profiles table
- ▶ Creates private home workspaces for each user as that user logs in for the first time
- ▶ Creates a new repository user for each user on the first login

Configuring groups

As stated earlier, only one group, “administrators,” is created by default when a repository with a security provider of **Database-based** is selected, so the next step we want to take is to configure groups.

In QMF for Workstation, you create and manage groups within a table named RSBI.AUTHID_TABLE (be careful here, as there is also an RDBI.AUTHID_TABLE). The RSBI.AUTHID_TABLE has two columns: PRIMARY and SECONDARY. The PRIMARY column is used to hold the login ID of the user being put into the group. The SECONDARY column is used to hold the name of the group into which the user is being placed.

Tip: A user can be in multiple groups.

User-group relationships can be taken from a z/OS security access facility feature, such as RACF user-groups. However, the product does not provide a way of reading or importing these relationships from RACF. Therefore, you must set up the process for acquiring and loading this information.

User-group relationships can be taken from a QMF catalog profiles table (Q.PROFILES or RDBI.PROFILE_TABLE, depending on choices made during the configuration process). However, be careful here, as QMF allows the user to be in multiple groups in the profile table based on the values of the ENVIRONMENT and TRANSLATION columns. Database-based security does not recognize these nuances. (See *Installing and Managing QMF for TSO/CICS* for details on the Q.PROFILES table.)

In the example used in this topic, the QMF profile table does not use ENVIRONMENT and TRANSLATION values to assign a given user to different groups depending on access mode, and thus a query such as the following one will be fine:

```
INSERT INTO RSBI.AUTHID_TABLE  
SELECT DISTINCT CREATOR, RESOURCE_GROUP FROM Q.PROFILES
```

Alternatively, completely new QMF security groups can be created by putting any PRIMARY/SECONDARY pair into the RSBI.AUTHID_TABLE. These groups and this security mode apply only to repositories and are not in play during QMF for TSO and CICS sessions.

After we execute the query just shown, we see the current configuration (Figure 8-9):

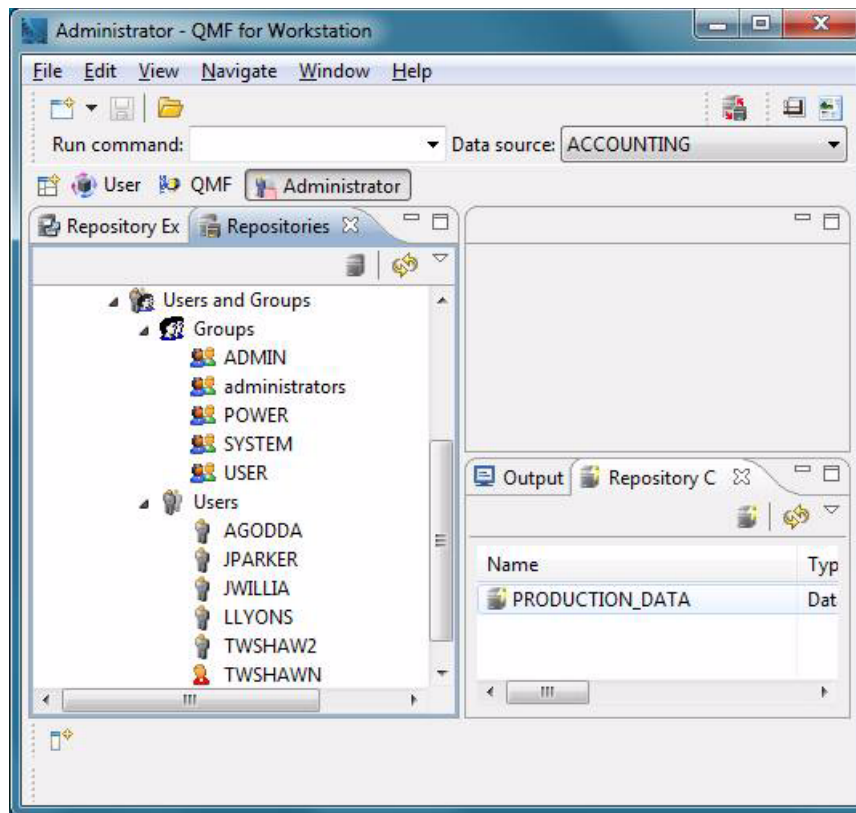


Figure 8-9 Inserting resource groups from the QMF profiles table

Tip: The only way to view the relationship between the users and the groups is to run a SELECT statement on the RSBI.AUTHID_TABLE. There is no user interface to view or administer these groups.

Setting security permissions for users and groups

When the users and groups are established, you can use them to set up Windows/Linux/UNIX-like permissions on the workspace and repository tree branches and the objects they hold. In general, a user ID that creates a tree branch or object (or otherwise has authority granted) can right-click, then select **Properties** → **Security** from the resulting context menu.

In Figure 8-10, we see the security properties for the TWSHAW2 home workspace. Only an administrator or TWSHAW2 can set the permissions on content owned by TWSHAW2.

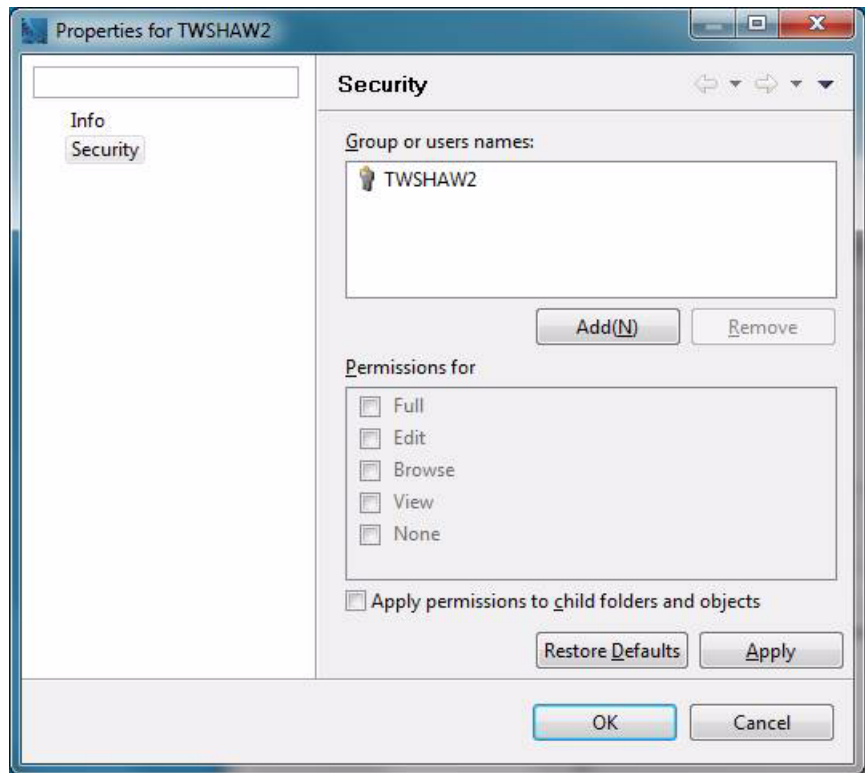


Figure 8-10 Properties and security on the TWSHAW2 workspace

By clicking the **Add(N)** button, TWSHAW2 can add a new user, who can be given permissions on this workspace. The only difference between a home workspace and other workspaces created by a user or an administrator is that, unlike a home workspace, the other workspaces are created with all defaults set to browse for the group “everyone”.

In Figure 8-11, we add the user JPARKER and all users in the IBM POWER® group will be added to the list eligible for access.

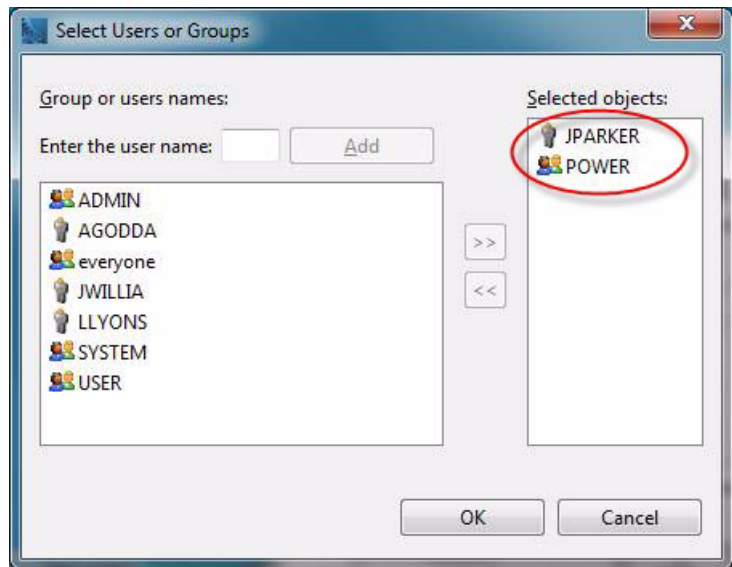


Figure 8-11 Adding a user and a group to a workspace

Notice that, while there are several groups that came from the RSBI.AUTHID_TABLE, the group “everyone” is special. It is predefined and all users are automatically in this group.

By default, the new list members are given View permission. In Figure 8-12, we elevate the POWER group to Edit permission.

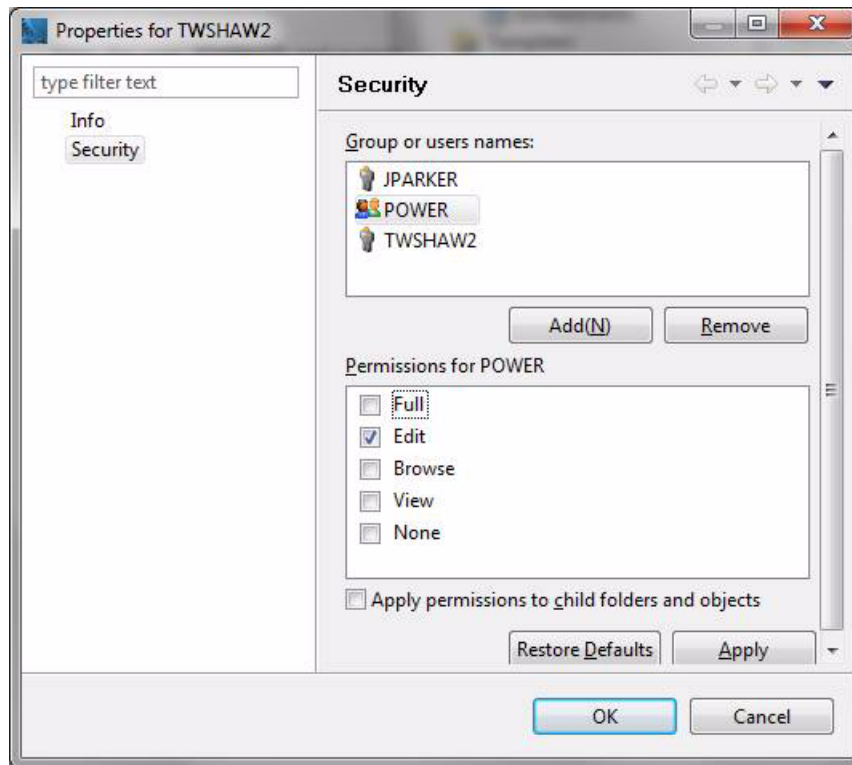


Figure 8-12 Changing the permission for the POWER group

The definitions of the permission levels are as follows:

- ▶ **Full:**
Grants permission to perform all functions (edit, view, browse, and delete) on the repository object. In addition, full permission allows users to modify permissions that have been assigned to the repository object.
- ▶ **Edit:**
Grants permission to make changes to, as well as view and browse, the repository object.
- ▶ **Browse:**
Grants permission to see the contents of the repository object. For example, the Browse permission on a directory in the repository allows a user to see what is inside that directory. Browse permission on a query object allows a user to open and run the query object.
- ▶ **View:**
Grants permission to see the repository object listed in the repository. For example, View permission on a directory in the repository allows the user only to see the directory. The user cannot open the directory or see the contents of the directory. View permission on a query object allows a user only to see the object; it cannot be opened or run.
- ▶ **None:**
Denies permission to see that the repository object is included in the repository.

A user such as TWSHAW2 cannot change permissions on repository content that is not owned by TWSHAW2 or for which Full permission has not been granted to TWSHAW2. For example, suppose that TWSHAW2 tries to access security properties for the multidimensional data sources shown in Figure 8-13.

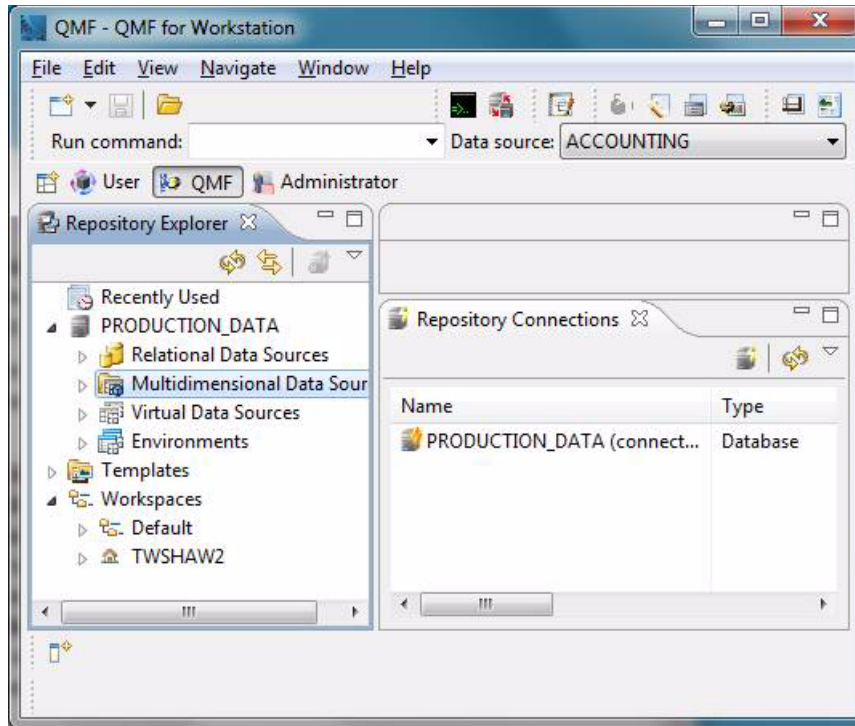


Figure 8-13 TWSHAW2 attempts to access multidimensional data sources

As shown in Figure 8-14, TWSHAW2 finds that he only has the ability to browse the data.

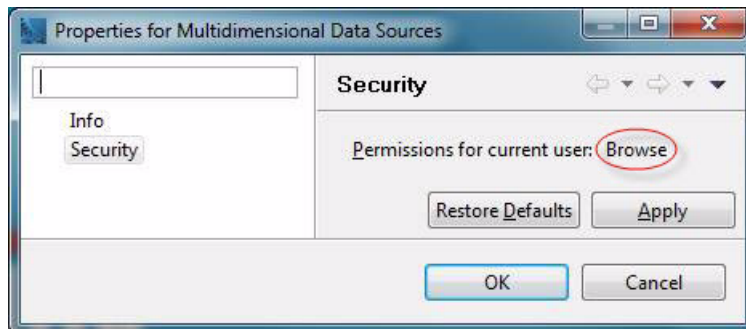


Figure 8-14 TWSHAW2's access to multidimensional data sources

By setting the permissions for the group “everyone” to **None**, the administrator can hide unwanted branches of the repository tree. For example, as shown in Figure 8-15, the administrator can set the permission to **None** for the group “everyone” on each of the multidimensional data sources, virtual data sources, templates, workspaces, and environments, one at a time.

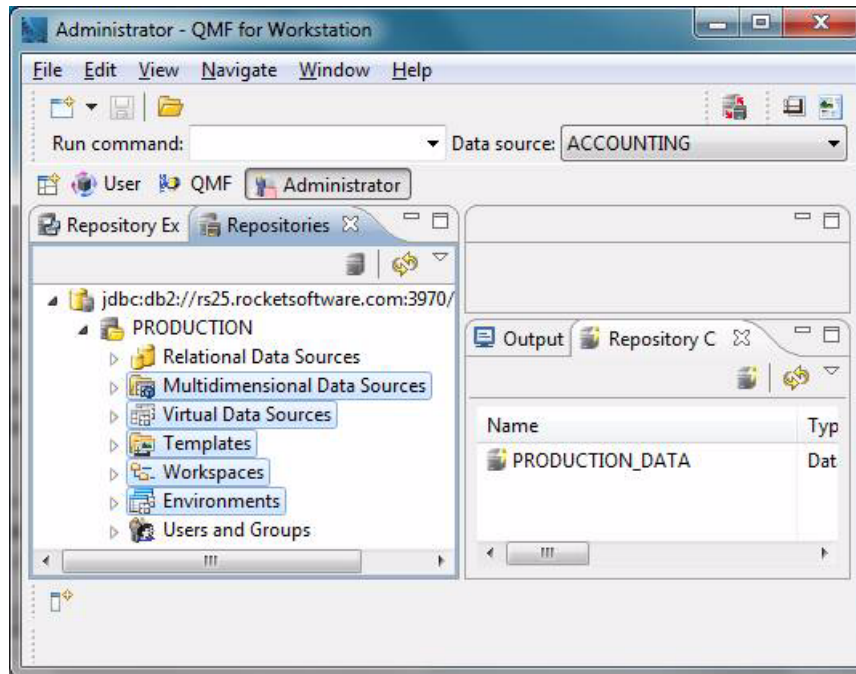


Figure 8-15 Setting permissions to None for the “everyone” group

It results in a very simplified explorer tree for end users, as shown in Figure 8-16.

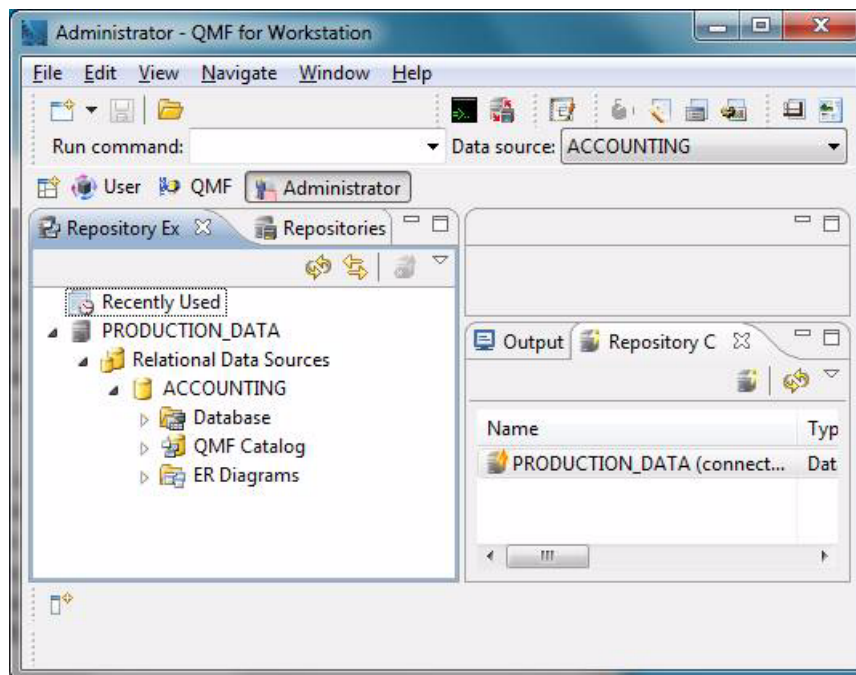


Figure 8-16 A simplified explorer tree presented to end users

After the users and groups are established, the setting of permissions is identical for the Internal and LDAP security provider settings, which we discuss next.

8.1.2 Internal security

The **Internal** security provider setting uses a set of QMF tables to store user IDs and encrypted passwords and groups, as well as the assigned permissions. QMF application logic enforces the users, groups, and permissions.

When the user connects to the repository, a login to the database holding the shared repository storage takes place. The internal security mechanism then requires another login to access the repository content (queries, procedures, forms, dashboards, and so on).

The first login is managed by the database. In the case of DB2 for z/OS, this login is typically handled by RACF (or whichever security manager is in place). The second login is managed by QMF and is handled by the application logic and the information stored in the QMF tables.

To set up internal security, you give the repository a name and select **Internal** for the security provider setting, as shown in Figure 8-17.

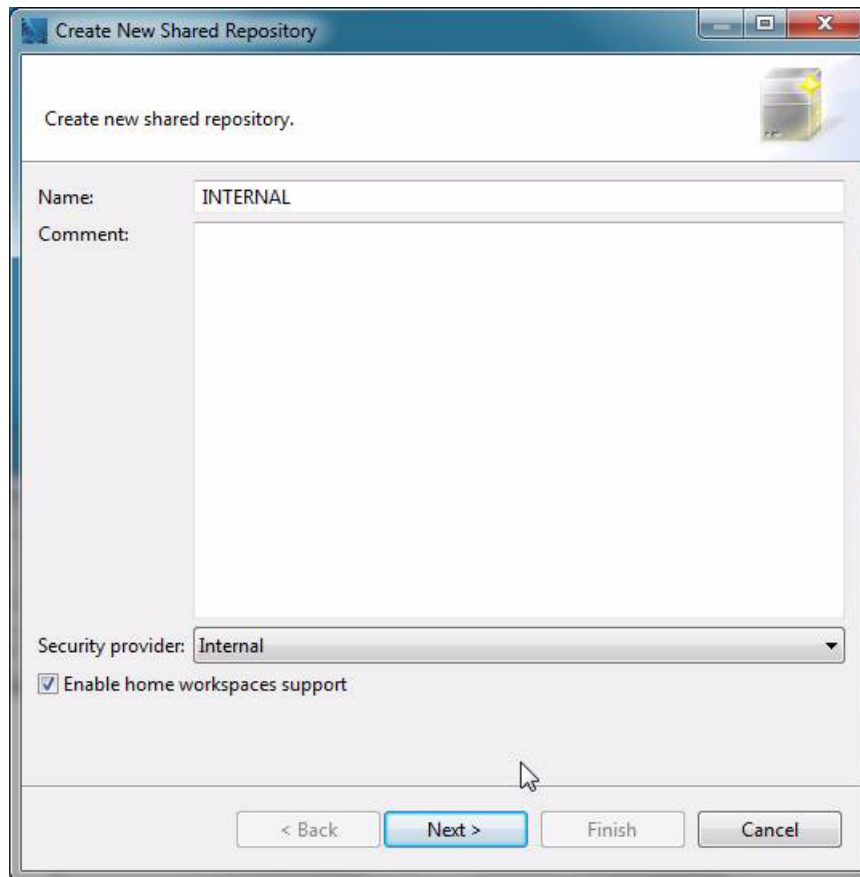
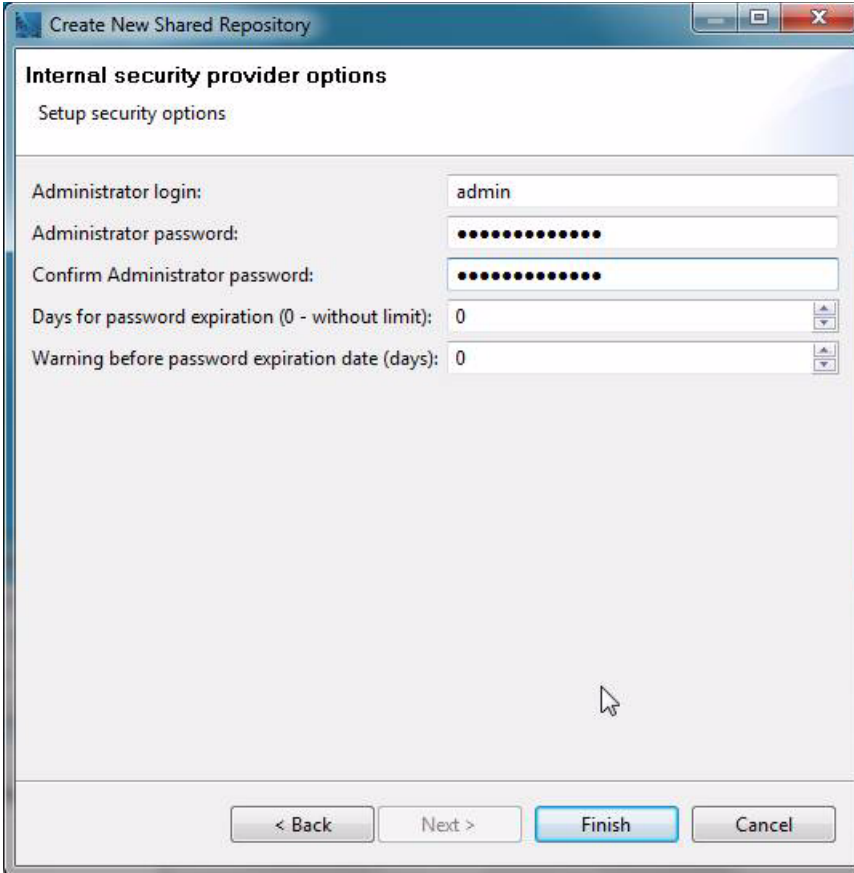


Figure 8-17 Establishing internal security

You must also establish administrator credentials, as shown in Figure 8-18. These credentials are stored in one of the QMF repository storage tables.



The screenshot shows a Windows-style dialog box titled "Create New Shared Repository". Inside, the "Internal security provider options" tab is selected, with the subtitle "Setup security options". The form contains the following fields and controls:

- Administrator login:** A text box containing the value "admin".
- Administrator password:** A password box filled with 12 dots.
- Confirm Administrator password:** A password box filled with 12 dots.
- Days for password expiration (0 - without limit):** A numeric spinner box set to "0".
- Warning before password expiration date (days):** A numeric spinner box set to "0".

At the bottom of the dialog, there are four buttons: "< Back", "Next >", "Finish" (which is highlighted in blue), and "Cancel".

Figure 8-18 Establishing administrator credentials for internal security

Click **Finish**. Notice that, in future sessions, when performing administration for this repository, these credentials must be used.

Now you can immediately connect using the administrator credentials, as shown in Figure 8-19.

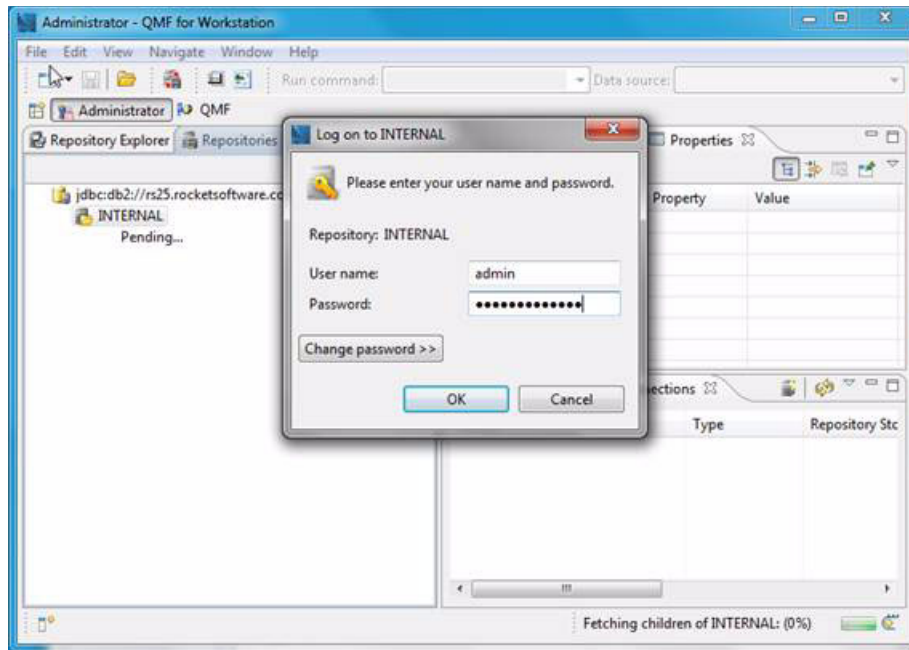


Figure 8-19 Connecting using the administrator credentials just established

Adding users

You must now configure data sources, as explained in Chapter 6, “Configuring access to data sources and populating user workspaces” on page 89. However, in addition to configuring the data sources, the users who will need access to this repository must also be configured. The login to a repository secured with internal has a **Change password** option. This option allows the administrator to create a set of users with a simple common temporary password that the users can change at the first login.

After the repository is established with internal security, there will be one user (the one defined for administration) and one group (“administrators”). Right-click **Users** to add a new user (Figure 8-20).

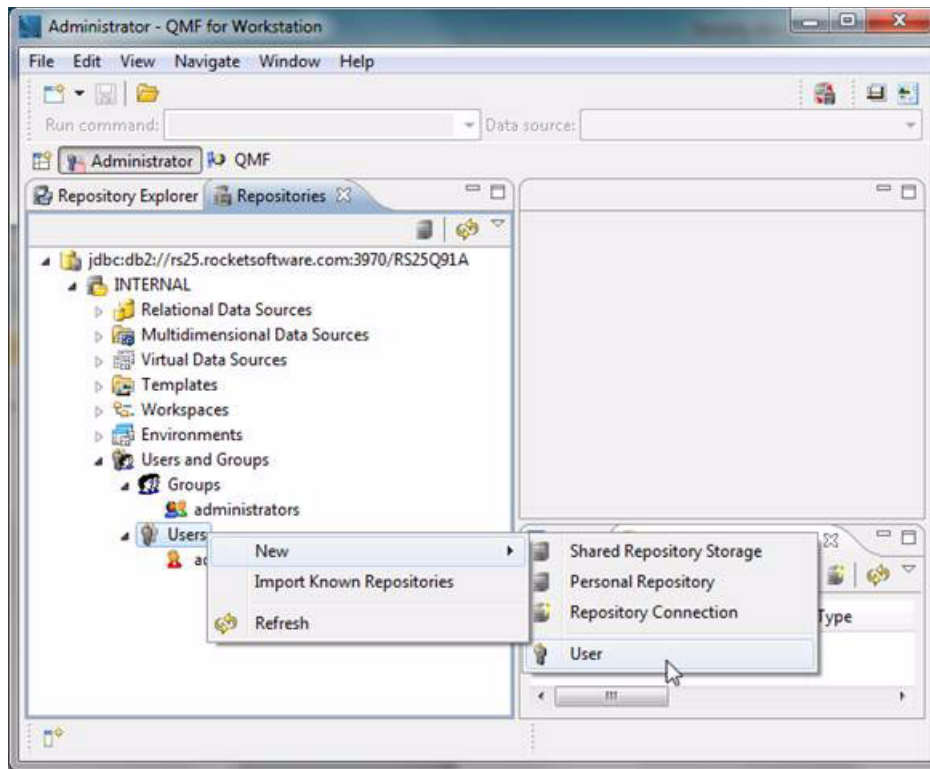


Figure 8-20 Adding a new user when the security provider setting is Internal

First, the new user must be defined as shown in Figure 8-21.

New Repository User

Add new repository user

User Information

User name: QAJPARKER

Full name: Joseph Parker

Password:

Confirm password:

Description: QA lead, administrator

☒ Has a home workspace

Group Membership

☐ everyone

☒ administrators

Finish Cancel

Figure 8-21 Defining a new user

This definition process must take place for each user of a repository that has a security provider setting of Internal. There is no bulk loading process for large numbers of users. Thus, for hundreds or thousands of users, a security provider setting of Internal might not be practical. If that is the case, one of the other security provider options might be more suitable.

Configuring groups

Groups are defined in a similar fashion, as shown in Figure 8-22.

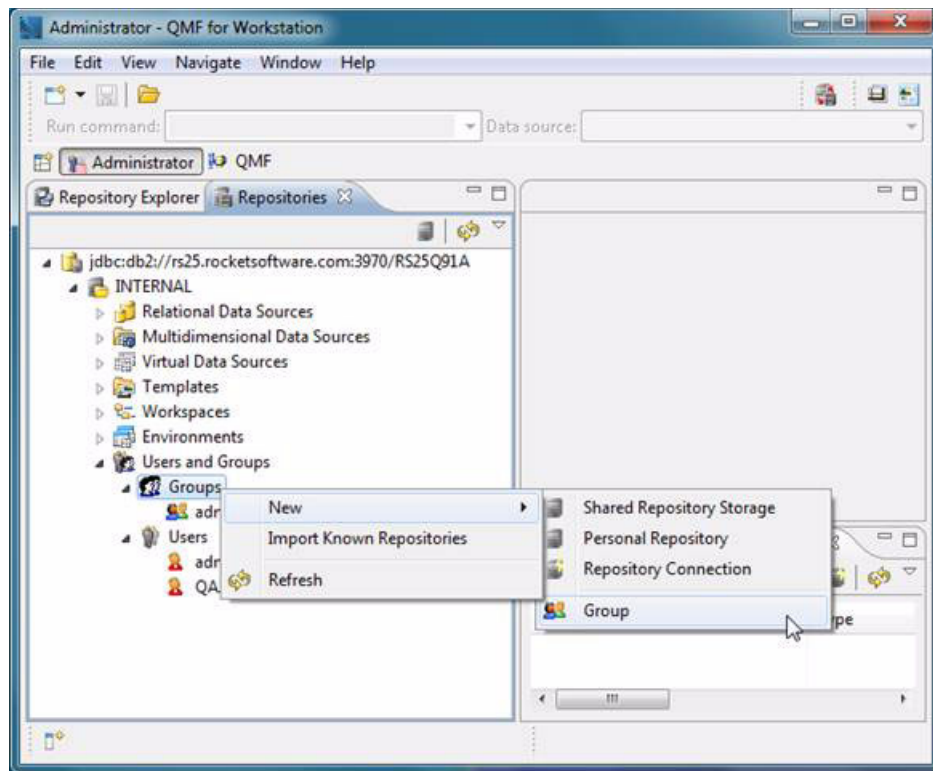


Figure 8-22 Defining groups when the security provider setting is Internal

Subgroups can be defined with a parent group and a set of users assigned, as shown in Figure 8-23 for the QA subgroup.

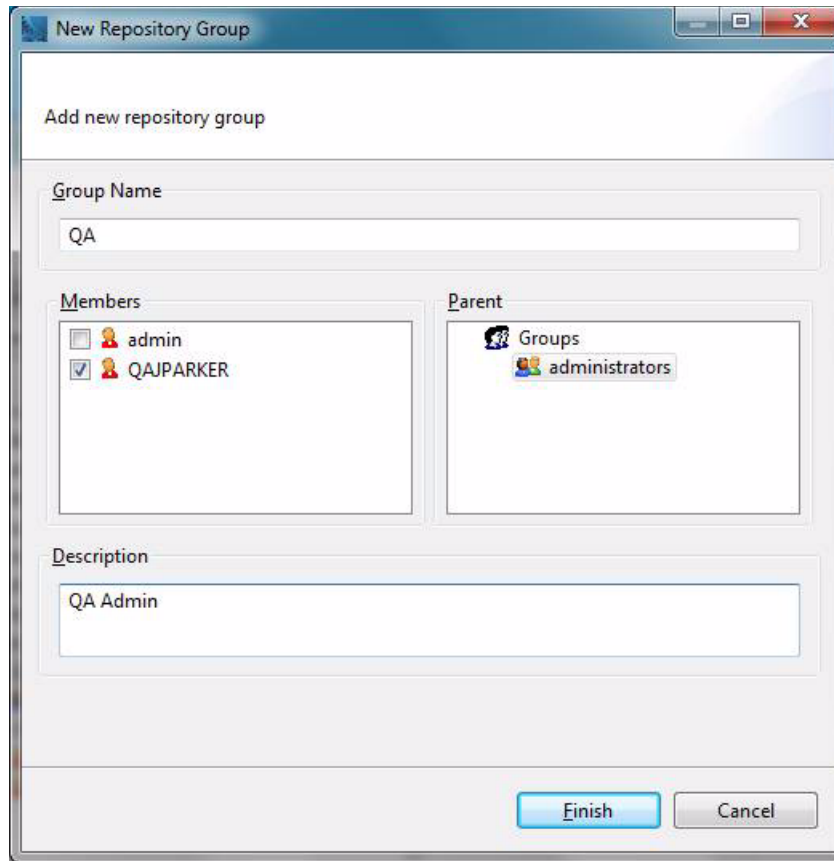


Figure 8-23 Parent group and assigned users

Figure 8-24 shows the newly defined QA subgroup as a subgroup of the administrators group.

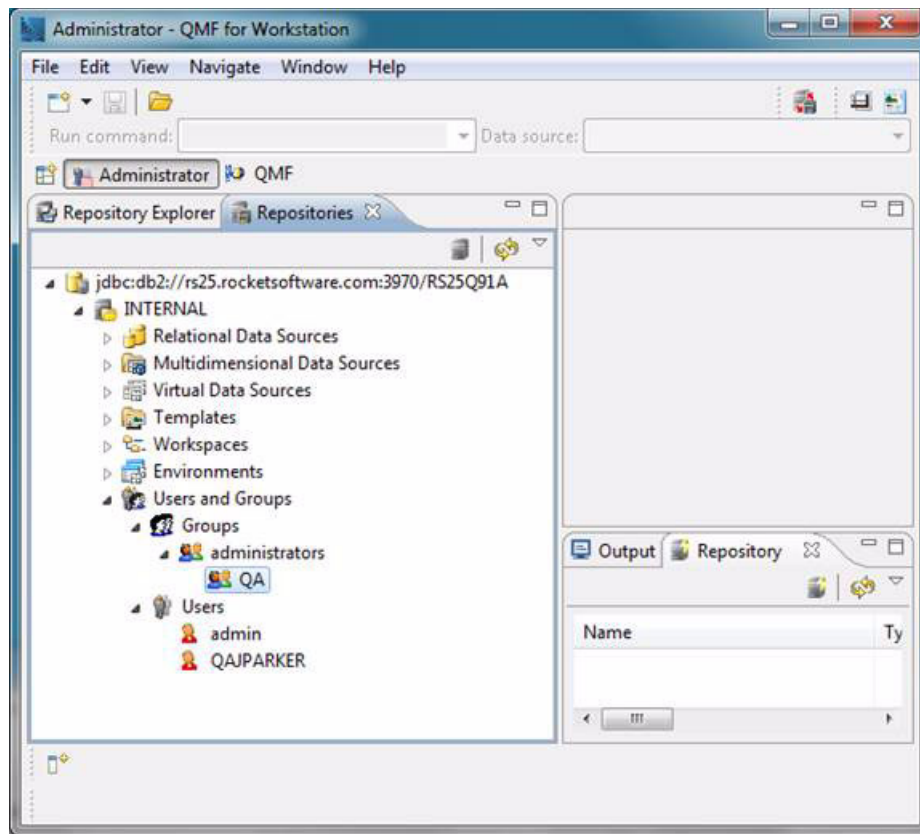


Figure 8-24 QA as a subgroup of the administrators group

Setting security permissions for users and groups

When the connection to the repository has been established, the users and groups can be used to apply permissions to the objects and folders in the workspaces exactly as in the previous example given for database-based security, explained in 8.1.1, “Database-based security” on page 147.

8.1.3 LDAP security

The LDAP security provider option uses a set of QMF tables to store LDAP connection information, groups, and assigned permissions. QMF uses the LDAP server to authenticate the user and manage the group memberships.

When a user connects to the repository, a login to the database holding the shared repository storage takes place. The LDAP security mechanism then requires another login to access the repository content (queries, procedures, forms, dashboards, and so on). The first login is managed by the database. In the case of DB2 for z/OS, this login is typically handled by RACF (or whichever security manager is in place). The second login is managed by the LDAP server.

To set up LDAP as the security provider, give the repository a name and select **LDAP** from the Security provider drop-down list (Figure 8-25).

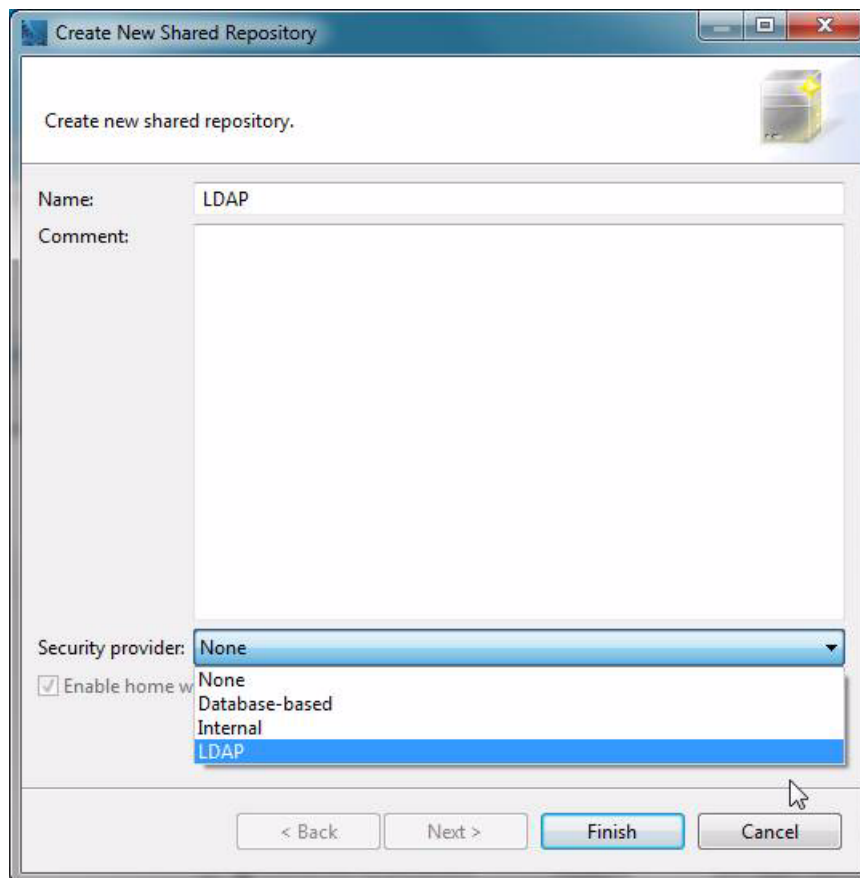


Figure 8-25 Specifying LDAP as the security provider

A popular choice for the LDAP server is the Windows Active Directory Server. However, as shown in Figure 8-26, there are many settings that are LDAP-server specific that need to be supplied. LDAP experts from within the organization should supply these settings.

An LDAP administrator must fill in the setting specific to your LDAP or Active Directory.

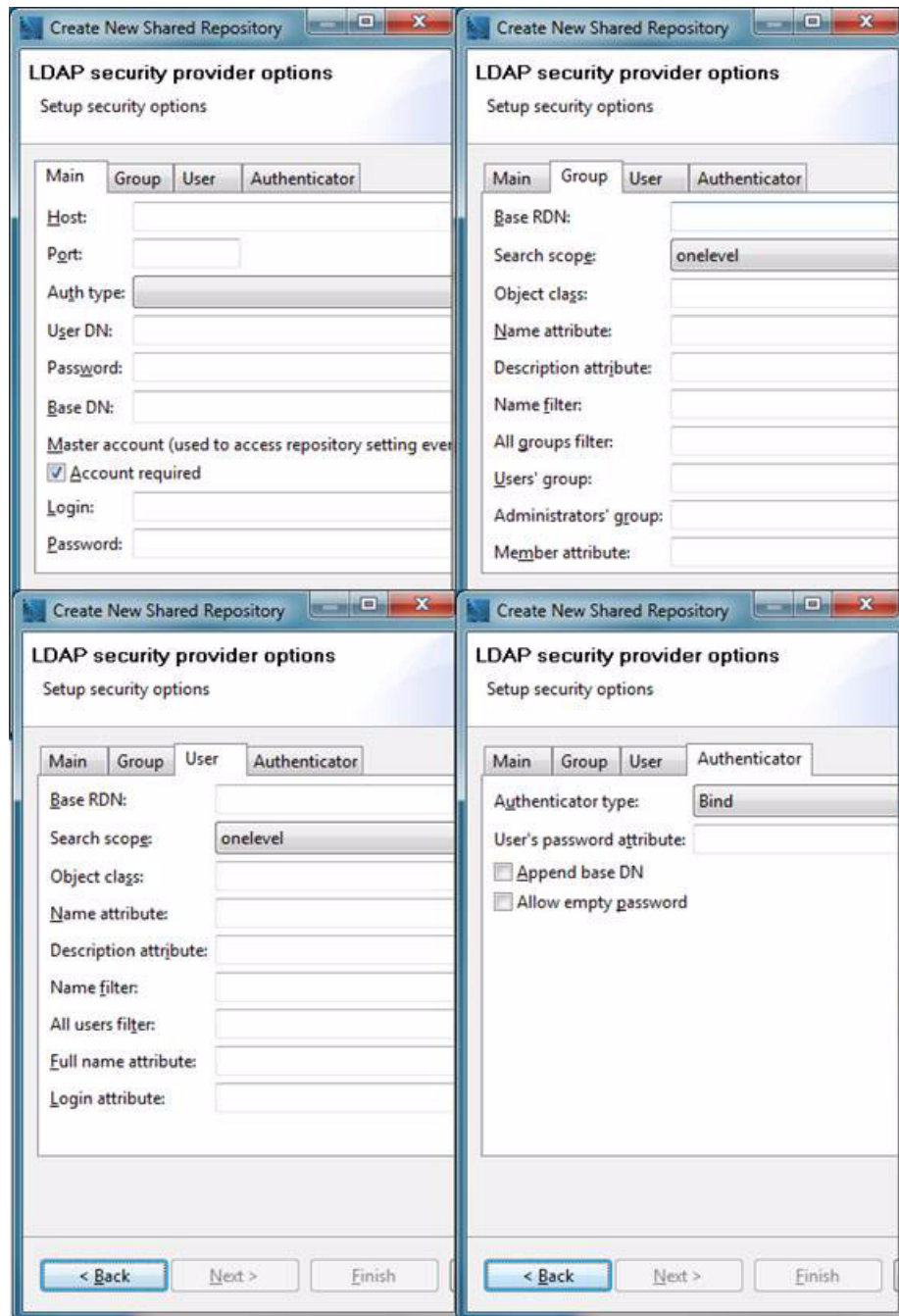


Figure 8-26 LDAP server settings for Windows Active Directory Server

After the connection to the LDAP-secured repository is established, the users and groups can be used to apply permissions to the objects and folders in the workspaces exactly as in the previous example given for database-based security in 8.1.1, "Database-based security" on page 147.

8.1.4 Using no security

A security provider setting of **None** is mostly suitable for an isolated user running a personal database with no other users, as this setting literally provides no security. If no security is used in a multi-user environment, then any user can edit, rename, or even delete anything in the repository, including the connection information to the data sources.

Personal repositories are automatically set up with no security in place, because no remote clients can connect to these types of repositories. For more on the Derby personal repository provided with QMF for Workstation, see 8.2.6, “Using the personal repository” on page 173.

8.2 Minimizing the number of logins required

Generally, there will be one repository storage database and many production data sources. So each work session will require one connection to the repository storage database and one connection to each of the production data sources of interest.

Furthermore, if the repository was secured with the Database-based, Internal, or LDAP options, there will be one more login required to get into the repository so that the user can be identified for permissions on the various objects. This topic looks at ways to cut down on the number of logins needed.

8.2.1 Automating the repository connection login

When QMF for Workstation starts, it requires the end user to log into a repository (behind the scenes it is logging into the repository storage database). However, if the administrator chooses, a permanent “automated” user ID and password for the repository storage can be recorded and used for this login, suppressing this dialog. In this case, every user makes this initial connection with the same ID.

Tip: If database-based security is used and the repository connection login is automated as described before, the security in place is equivalent to a setting of **None** for the security provider option. It is because each user will log into the repository storage with the same user ID, and therefore the database-based security will use that same user ID for everyone.

This “automated ID” should have only the privileges for the repository storage database to run the static SQL packages, as was automatically supplied by the **Grant** button in the Create New Shared Repository Storage wizard shown in Chapter 6, “Configuring access to data sources and populating user workspaces” on page 89. It is only used to get the connection information for the production data sources and build the end user’s **Repository Explorer** tree. The users cannot query any tables using this ID.

Tip: This “automated” user ID should be non-expiring.

To set this ID, right-click the repository connection and select **Edit** (Figure 8-27).

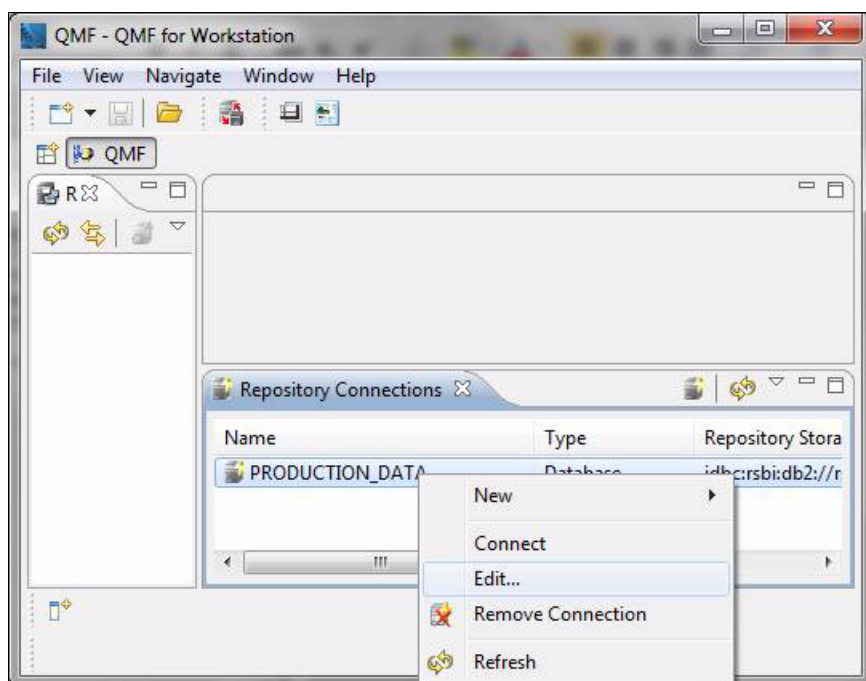


Figure 8-27 Setting up the automated ID

The “Remember password” option is available only in the **Edit** option for repository connections. It does not show up as a check box on the user login dialog, so it must be preconfigured. While the administrator can distribute an automated ID upon deployment, the user can change the ID on the local system, again by selecting **Edit**.

8.2.2 Using repository storage database credentials to log into the repository

The **Database-based** security provider setting automatically logs the user into the repository using the same credentials as those used to connect to the repository storage database, thus eliminating a prompted login.

When **Internal** or **LDAP** is chosen, there is a significant amount of setup and the user is forced to go through an additional login each time.

8.2.3 Storing authentication information for each data source

The next options to reduce the number of logins are performed at the data source level. One option to reduce the number of logins is to check the **Remember the password** check box (Figure 8-28). When this box is checked, the user will not be prompted when logging into the data source unless the password that has been remembered has expired or has been changed.

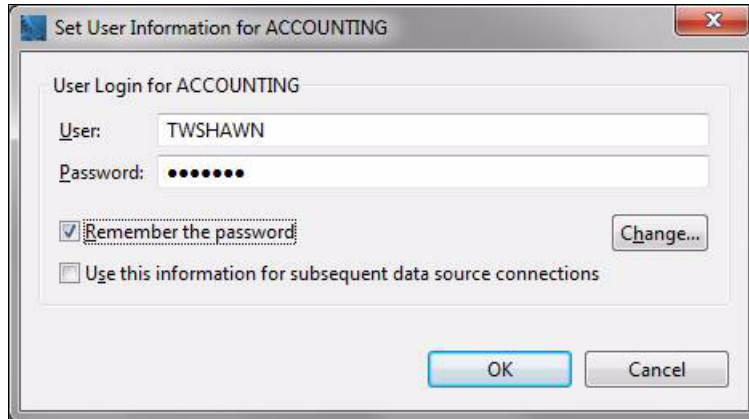


Figure 8-28 Configuring QMF to remember the data source login password

If saving passwords is not permitted at your installation, you can check the **Use this information for subsequent data source connections** check box. For example, where there are multiple LPARs within a sysplex, there could be many data sources and the user has the same user ID and password on *all* systems. This option would take the first login to a data source during a session and use those credentials for all of the other data sources connected to during the session.

In our setup shown thus far, this option would still require one login to the repository storage database and one login to the first data source of a session. However, it does significantly reduce logins if there are many data sources of interest in a session.

8.2.4 Using repository storage database credentials to log into each data source

Where the user ID and password are guaranteed to be the same at all or some of the data sources, including the repository storage database, the credentials for the repository storage database (as entered in the repository connection login) can be reused for each of the data sources configured within it.

This setting is made on a case-by-case basis for each database by the administrator. It is controlled by the **Try to use repository logins and passwords to connect this data source** check box (Figure 8-29).

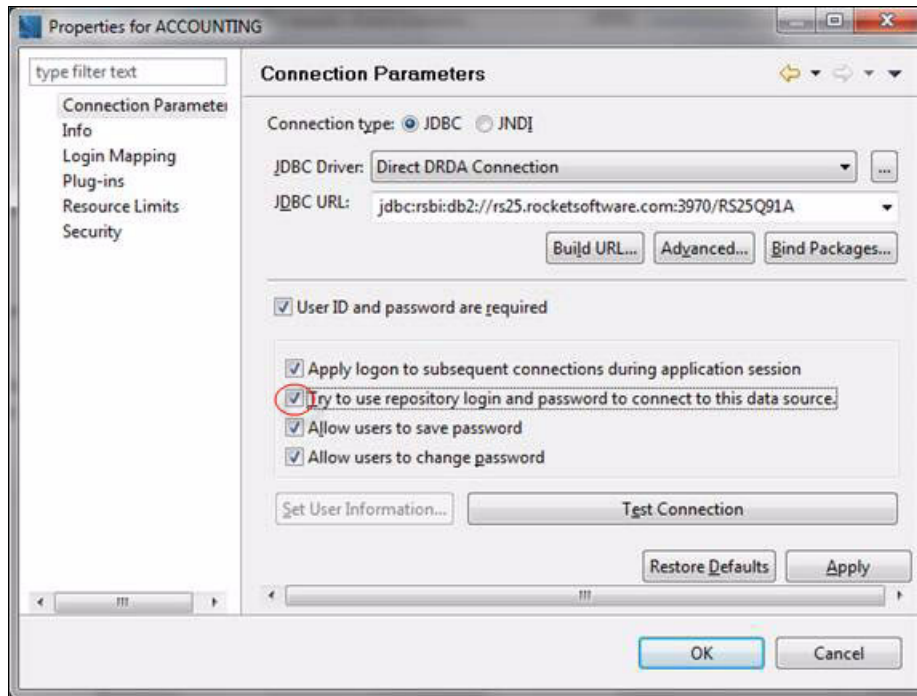


Figure 8-29 Reusing repository storage credentials for the data sources

In this case, “repository login” means the login to the repository, not the repository storage connection. If the security provider option is set to Internal, then the credentials provided for the internal engine are used at the particular database. If the security provider is LDAP, then the credentials provided for the LDAP authentication will be used at the particular database.

Because the security provider is **Database-based**, the repository connection login defaults to the login for the repository and is then reused a third time for the data source. Next we look at some examples.

Example 1

A user starts a session, then is prompted for a repository connection and supplies a user ID and password (no automated ID has been supplied, Remember password has not been checked). The authentication here is with the relational database holding the repository storage tables. The purpose of this login is to get a listing of the single repository (high-level qualifier) designated by the repository connection.

If this repository has security (as it absolutely should), then when the security provider option is set as follows, the login process is handled as described:

- ▶ **Internal:** A login prompt will occur. This login will be validated against the QMF internal credential store.
- ▶ **LDAP:** A login prompt will occur. This login will be validated against the local operating system’s LDAP information store, such as Active Directory for Windows.
- ▶ **Database-based:** No login is required. The database user ID used to connect to the repository storage database will be reused here for all privilege lookups.

A list of data sources is then provided. The user expands one and is prompted for a login. Total logins are as follows:

- ▶ Two if Database-based security was specified
- ▶ Three if LDAP or Internal security was specified

Example 2

A user starts a session and is not prompted for a repository connection because an automated ID has been supplied or Remember password has been checked. The authentication here is with the relational database holding the repository storage tables. The purpose of this login is to get a listing of the single repository (high-level qualifier) designated by the repository connection.

If this repository has security (as it absolutely should), then when the security provider option is set as follows, the login process is handled as described:

- ▶ **Internal:** A login prompt will occur. This login will be validated against the QMF internal credential store.
- ▶ **LDAP:** A login prompt will occur. This login will be validated against the local operating system's LDAP information store, such as Active Directory for Windows.
- ▶ **Database-based:** No login is required. The database user ID used to connect to the repository storage database will be reused here for all privilege lookups.

A list of data sources is then provided. The user expands one and is prompted for a login. Total logins are as follows:

- ▶ One if Database-based security was specified
- ▶ Two if LDAP or Internal security was specified

Example 3

A user starts a session and is not prompted for a repository connection because an automated ID has been supplied or Remember password has been checked. The authentication here is with the relational database holding the repository storage tables. The purpose of this login is to get a listing of the single repository (high-level qualifier) designated by the repository connection.

If this repository has security (as it absolutely should), then when the security provider option is set as follows, the login process is handled as described:

- ▶ **Internal:** A login prompt will occur. This login will be validated against the QMF internal credential store.
- ▶ **LDAP:** A login prompt will occur. This login will be validated against the local operating system's LDAP information store, such as Active Directory for Windows.
- ▶ **Database-based:** No login is required. The database user ID used to connect to the repository storage database will be reused here for all privilege lookups.

A list of data sources is then provided. The user expands one and is not prompted for a login because Remember password is checked or Try to use repository login is checked. Total logins are as follows:

- ▶ None if Database-based security was specified
- ▶ One if LDAP or Internal security was specified

8.2.5 Mapping logins to group IDs

Login mapping adds an additional dimension or degree of freedom for the administrator to reduce the number of logins. The methods previously described are, for the most part, dependent on the end-user to implement and maintain as passwords expire.

Login mapping allows the administrator to assign users or groups to use a particular ID and password based on the ID and password used when the initial repository connection is made. For example, six users can be placed in a group called SUPPORT. Then the group SUPPORT can be assigned the ID SUPPID with an accompanying password.

When any of the users in the SUPPORT group make the repository connection, the login mapping is read and the connection to the database is made. The user is directed to the data source, but will be logged in under the assigned ID SUPPID. It is most useful when applied to a group. Eventually, when the password expires, it only needs to be changed in one place.

See the QMF for Workstation help for a thorough, step-by-step set of instructions for login mapping.

8.2.6 Using the personal repository

A final option for reducing logins is the personal repository. QMF for Workstation installs with an internal Derby (open source) database engine. Derby is an embedded Java database for applications to store application data. QMF for Workstation end users can create repository storage databases on their local machines. These can be set up with the connection information for remote databases.

Important: The Derby database, as created by QMF for Workstation, can only support one connection at a time and so it is not suitable for a workgroup or an enterprise. However, because of this personal, one-connection limit, Derby does not require a login user ID or the creation of a password.

To set up the Derby database, first return to the Welcome window by selecting **Help** → **Welcome** from the menu, as shown in Figure 8-30.

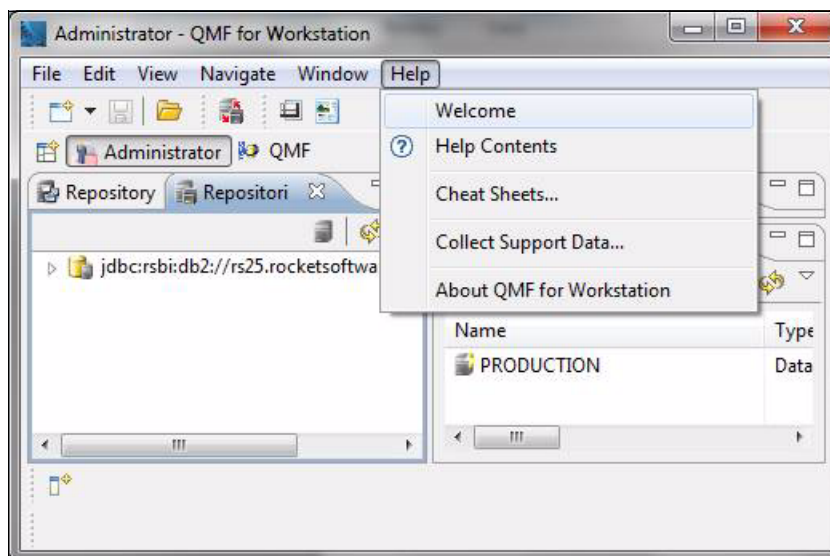


Figure 8-30 Navigating to the QMF Welcome window

The right-most icon will create a personal repository complete with a Derby database with sample tables and reports (Figure 8-31).

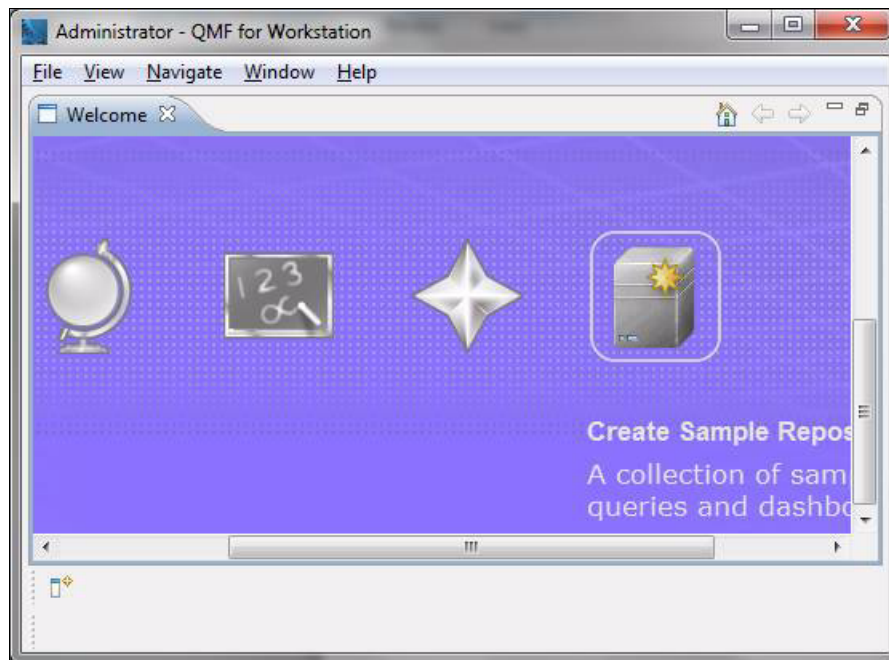


Figure 8-31 Icon to set up a personal repository from the Welcome window

The Derby repository connection is called Samples and the Derby database is Sample Data Source (Figure 8-32).

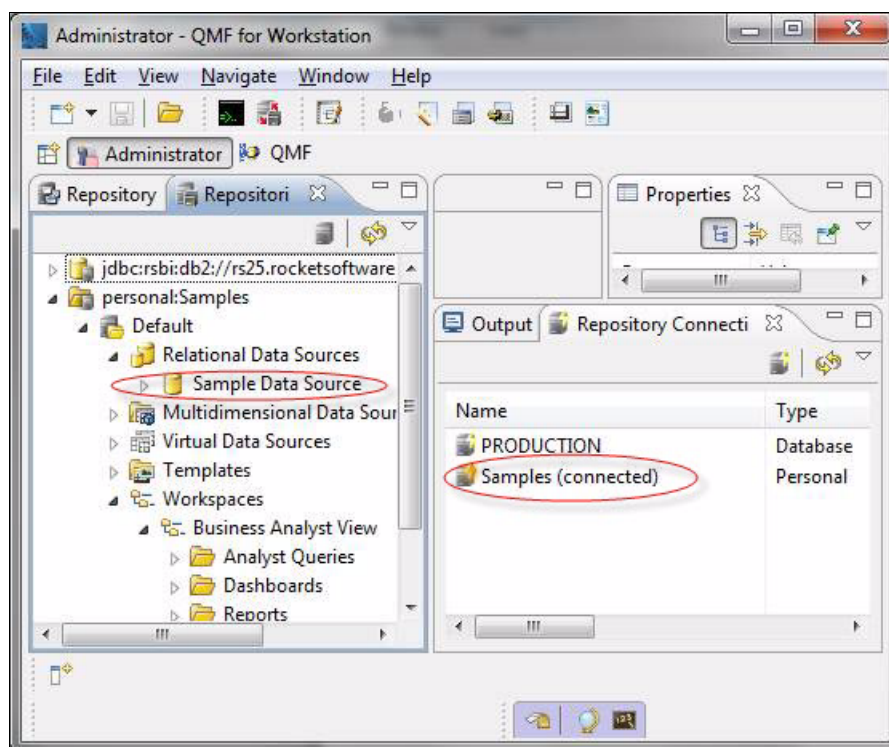


Figure 8-32 The Derby sample data source and repository

An empty personal repository can be created by selecting **File → New → Personal Repository**. Personal repositories are stored on the local machine in the current user's profile directory:

- ▶ On Windows XP, this directory is as follows:

C:\Documents and Settings\<user>\Application Data\IBM\QMF for Workstation\Personal Repositories

- ▶ On Windows 7, this directory is as follows:

C:\Users\<user>\AppData\Roaming\IBM\QMF for Workstation\ Personal Repositories

Tip: You can copy the subdirectories from the personal repositories directory on one machine to the personal repositories directory on another and, in doing so, transfer the personal repository, connection, data, reports, and all objects to the new system.

You can also drop and recreate the samples repository at any time by right-clicking it and selecting **Delete repository** from the context menu.

8.3 Use of security in dashboards

We discuss dashboards in depth Chapter 14, “Putting it all together: Developing dashboards” on page 289. However, to remain on point regarding security, we discuss dashboard security here. For dashboards in a repository with a security provider setting of Database-based, Internal, or LDAP, there is an interesting function in the expression builder that can bring the security into any aspect of the dashboard.

For example, two different users of the same dashboard might see different behavior based on their user IDs or security groups. One user might see only a subset of the rows in a table or only one series on a multiserie chart. This user might find certain buttons disabled, or switching behavior or entire layouts or scenes might be inaccessible, all based on the user ID. The function that provides this security feature is:

```
=isSecurityLevel(strText)
```

Here, `strText` is the name of a dashboard security list. The dashboard security list is a Globals element that lets you select which users or groups you want to bring into scope for the dashboard.

The `isSecurityLevel` function is commonly used in an “if” statement in the Visible property of the object to be governed.

For example, consider the following statement:

```
Label1.Visible = if(isSecurityLevel(strText),1,0)
```

This statement means that, if the current user is in the security list called `strText`, QMF makes this `Label1` visible; otherwise QMF hides it from view.

8.3.1 Integration methods for establishing security in dashboards

The following topics step through three common integration methods for establishing security in dashboards.

Here are the specifications for the sample dashboard, which is shown next in Figure 8-33.

- ▶ Dashboard layout:
 - Globals: Two security lists:
 - slAdmin contains Admin groups and users.
 - slSpecial contains Special groups and users.
 - Queries:
 - CompanyCashFlow shows total sales and profits by companyid.
 - MonthlyCashFlow shows total sales and profits by monthid.
 - Four scenes:
 - CashFlowSummary scene:
 - Label title showing current user ID
 - Column chart for Query CompanyCashFlow
 - Table for Query MonthlyCashFlow
 - Button called **Details** to navigate to the other scenes
 - DetailAdmin scene:
 - Content only for users in slAdmin
 - DetailSpecial scene:
 - Content only for users in slSpecial
 - DetailPublic scene:
 - Content for all users.

Tip: The three detail scenes are placeholder scenes and are not populated in this example. For more information about scenes, see Chapter 14, “Putting it all together: Developing dashboards” on page 289.

- ▶ Dashboard action:
 - The column chart in the dashboard shows the Sales and Profits series to members of slAdmin. Other users only see the Sales series.
 - The table omits the last quarter values unless the user is in slAdmin.
 - The **Details** button navigates to a different scene depending on which security list the user is in.

Tip: It is a “best practice” to create the dashboard without security and then, when it is running as desired, save a new version of it and implement the security features.

Figure 8-33 shows the dashboard with the queries and scenes.

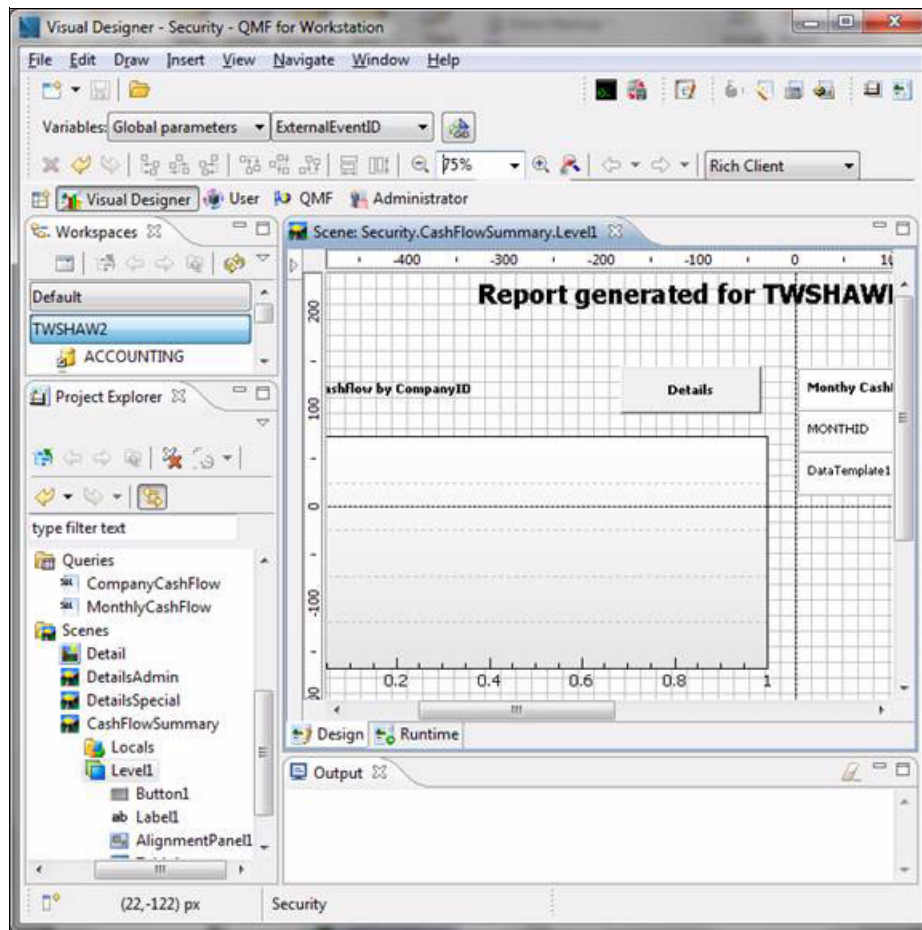


Figure 8-33 Sample dashboard with queries and scenes, ready to have security implemented

The queries that drive this dashboard are shown in Figure 8-34.

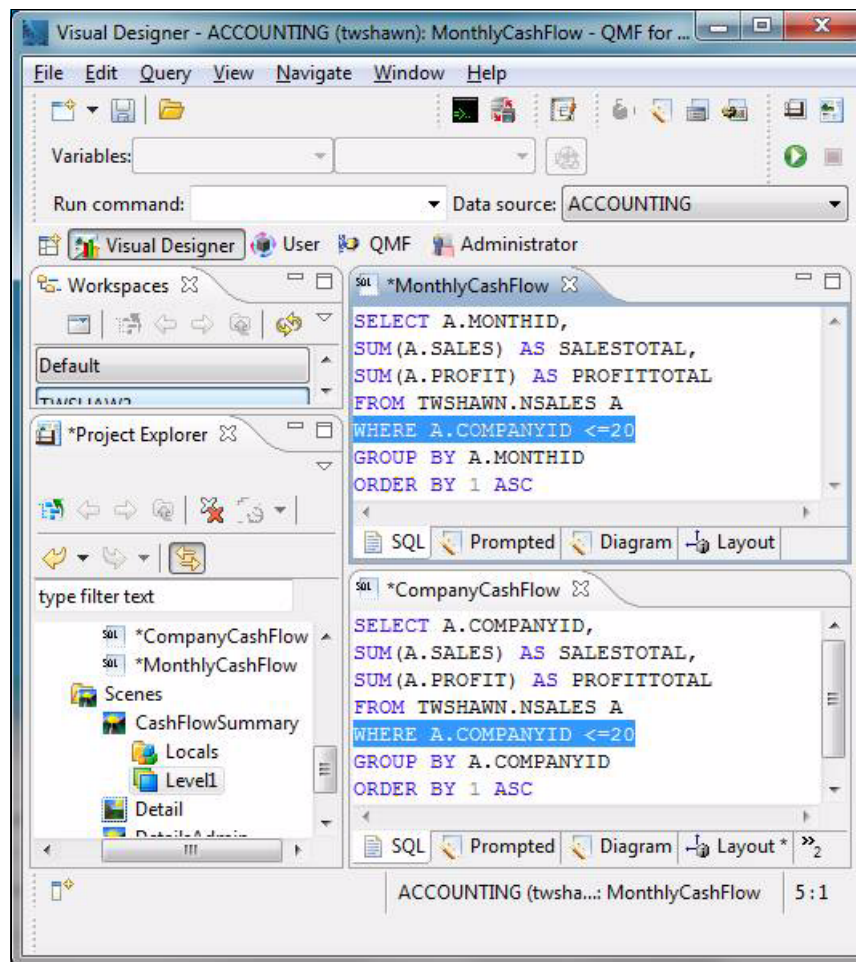


Figure 8-34 The queries for the sample dashboard

Notice that the data returned is being restricted by a traditional WHERE clause. This type of security applies to all users. The security lists will allow the same data to be viewed with different restrictions for different users.

When the dashboard runs with no security lists, it shows everyone all of the data (Figure 8-35).

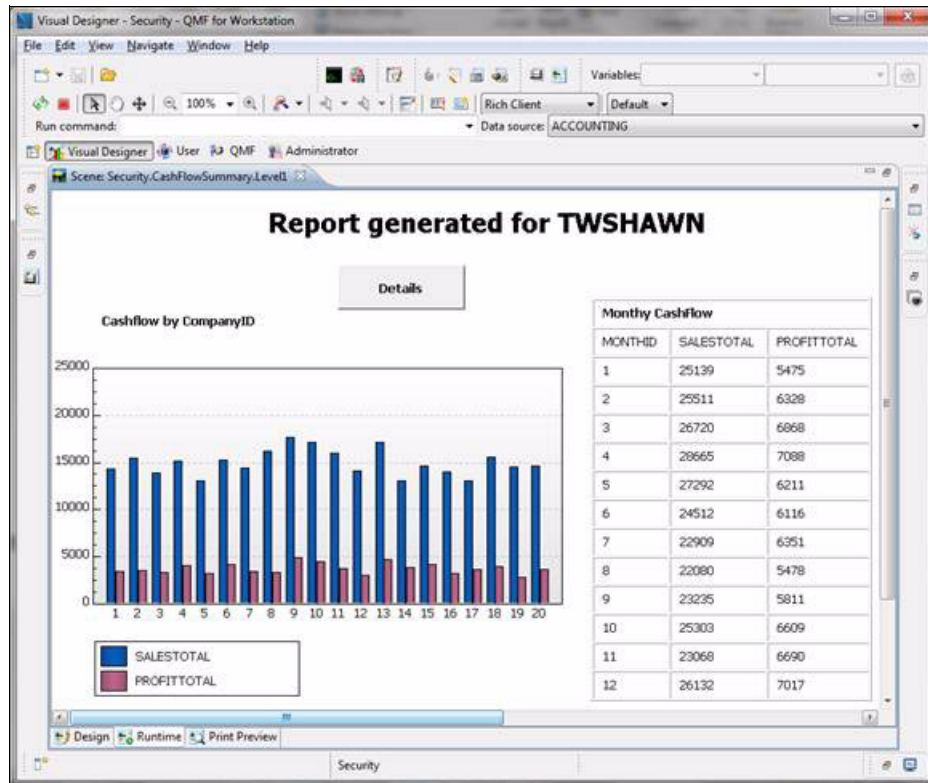


Figure 8-35 Sample dashboard with no security list

Notice that the text "Report generated for TWSHAWN" is in a label with a formula for the Text property. The formula is as follows:

```
= 'Report generated for ' + upper(DSQAO_CONNECT_ID)
```

DSQAO_CONNECT_ID is a global variable. The list of the available global variables is in the menu **View** → **Preferences** → **Global Variables**.

Double-click **Globals** → **Security Lists**. The New Security List wizard opens (Figure 8-36).

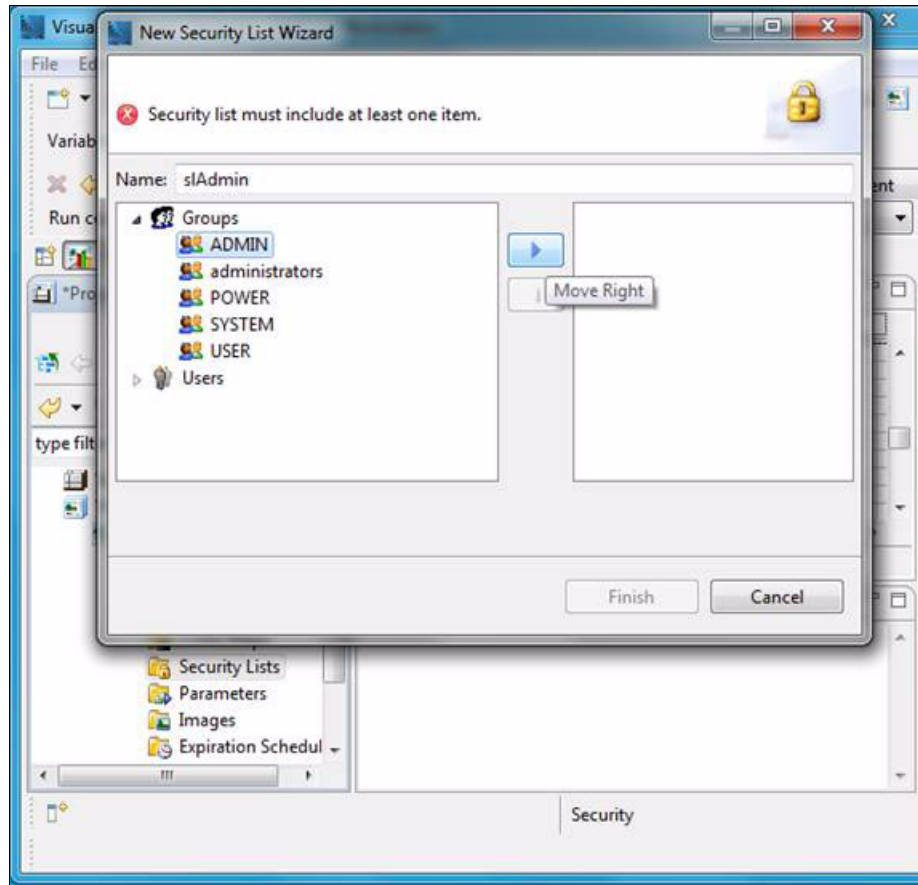


Figure 8-36 The global security list

We name the list in our example **slAdmin**. We then click **Move Right** and click **Finish**. By selecting the group **ADMIN**, all users in that group will be included in any functions or expressions using the **slAdmin** security list.

We double-click **Globals** → **Security Lists** again to create the slSpecial security list, as shown in Figure 8-37.

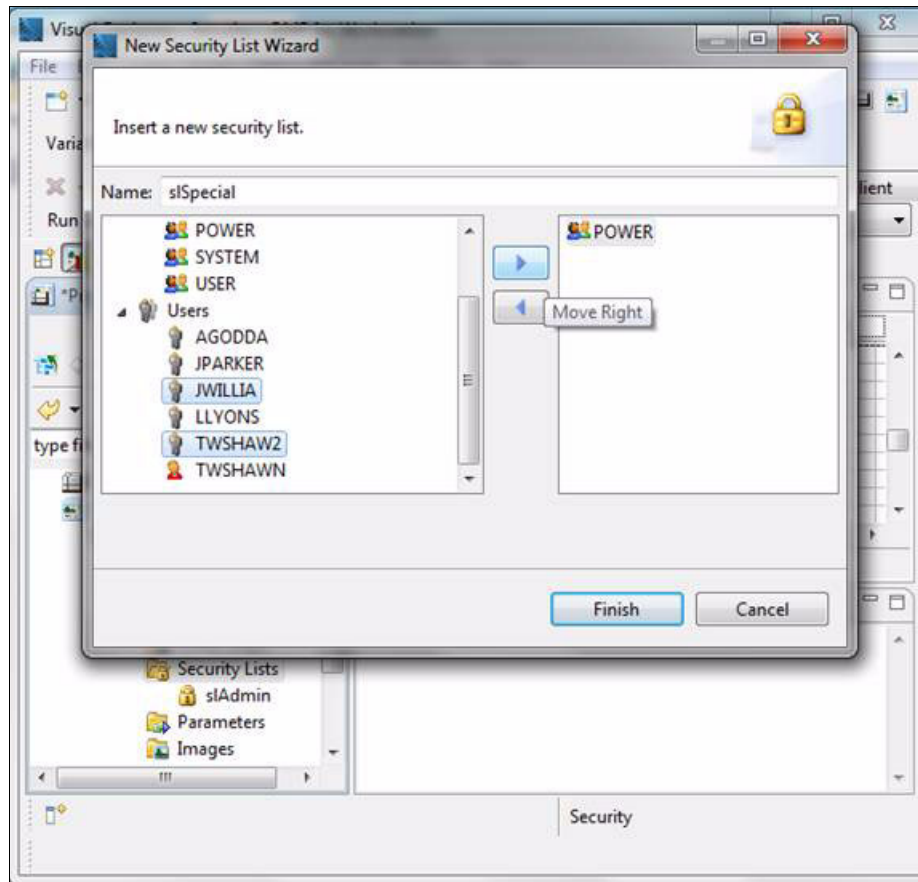


Figure 8-37 Naming the slSpecial security list

We then select the group POWER and the users JWILLIA and TWSHAW2. With this selection, all users in the group POWER, as well as the two additional users JWILLIA and TWSHAW2, are included in any functions or expressions using the slSpecial security list.

Now we implement security for the dashboard. We start with field-level security by restricting PROFITTOTAL from the column chart (Figure 8-38 on page 182). The entry point to implement the desired security is at the column chart data template on the vertical value bar for PROFITTOTAL.

Select the value **true** for the **Visible** property and then double-click the value to bring up the Expression Designer.

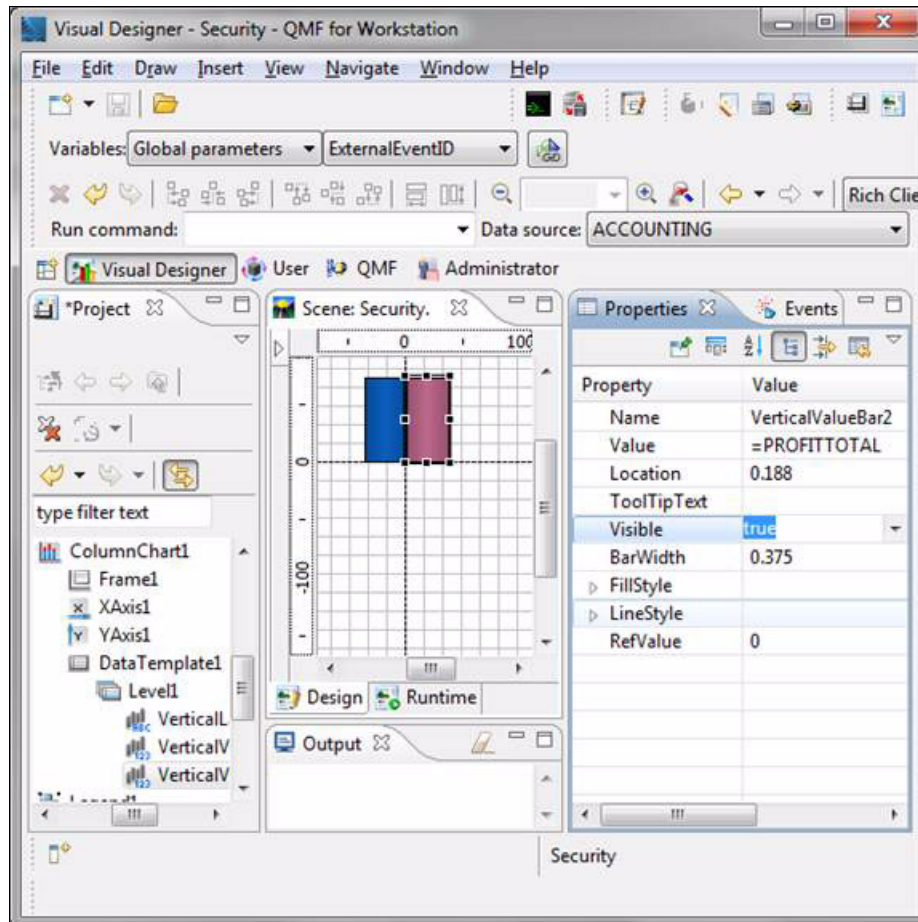


Figure 8-38 Specifying field-level security for PROFITTOTAL

We can now enter the following formula, which is shown in Figure 8-39:

```
=if(isSecurityLevel("slAdmin"),1,0)
```

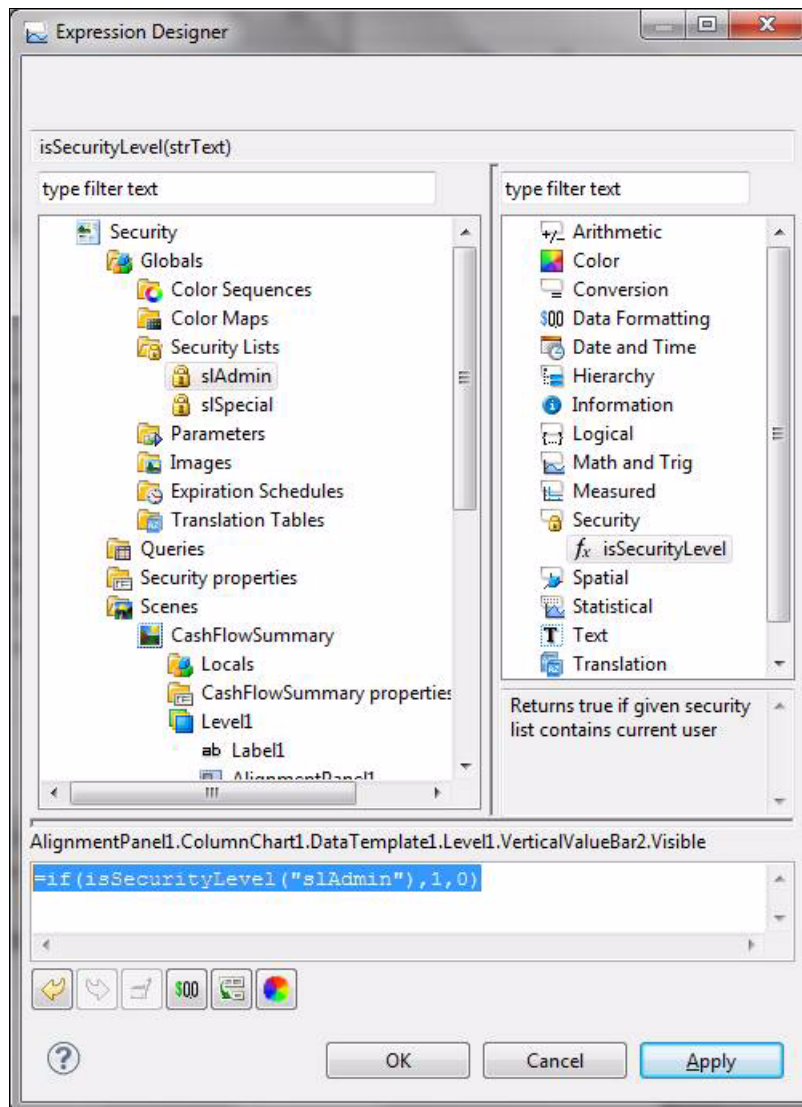


Figure 8-39 The formula for the Visible property

This formula means that, if the current user is in the slAdmin security list, then set Visible to true (1); otherwise set it to false (0) and hide PROFITTOTAL.

Click **OK** to close the Expression Designer.

The dashboard, run by TWSHAWN (who is in the slAdmin list because he is in the ADMIN group), still shows the PROFITTOTAL (Figure 8-40).

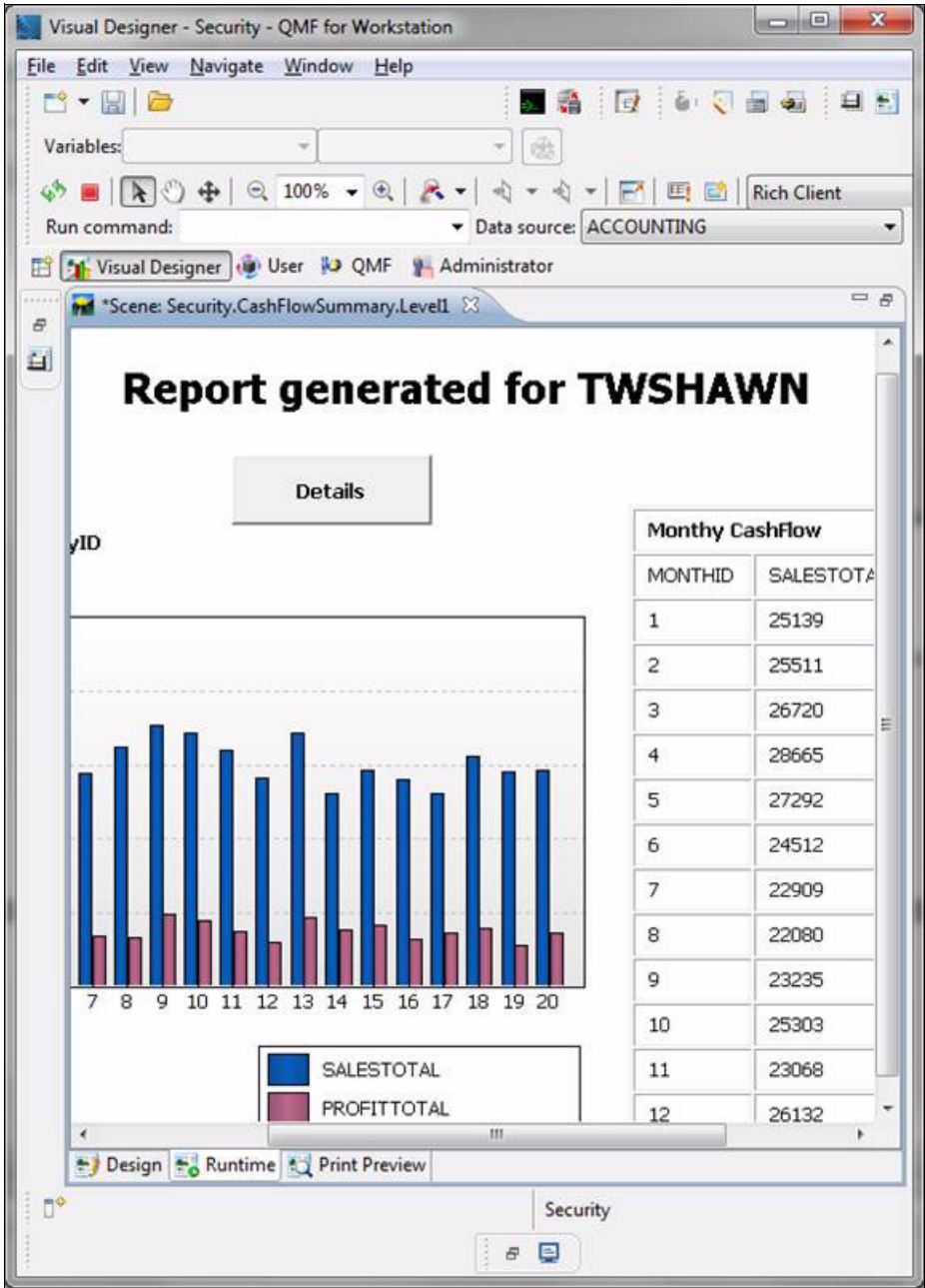


Figure 8-40 The dashboard as seen by TWSHAWN

However, when TWSHAW2 runs the dashboard, the PROFITTOTAL bars are gone (Figure 8-41).

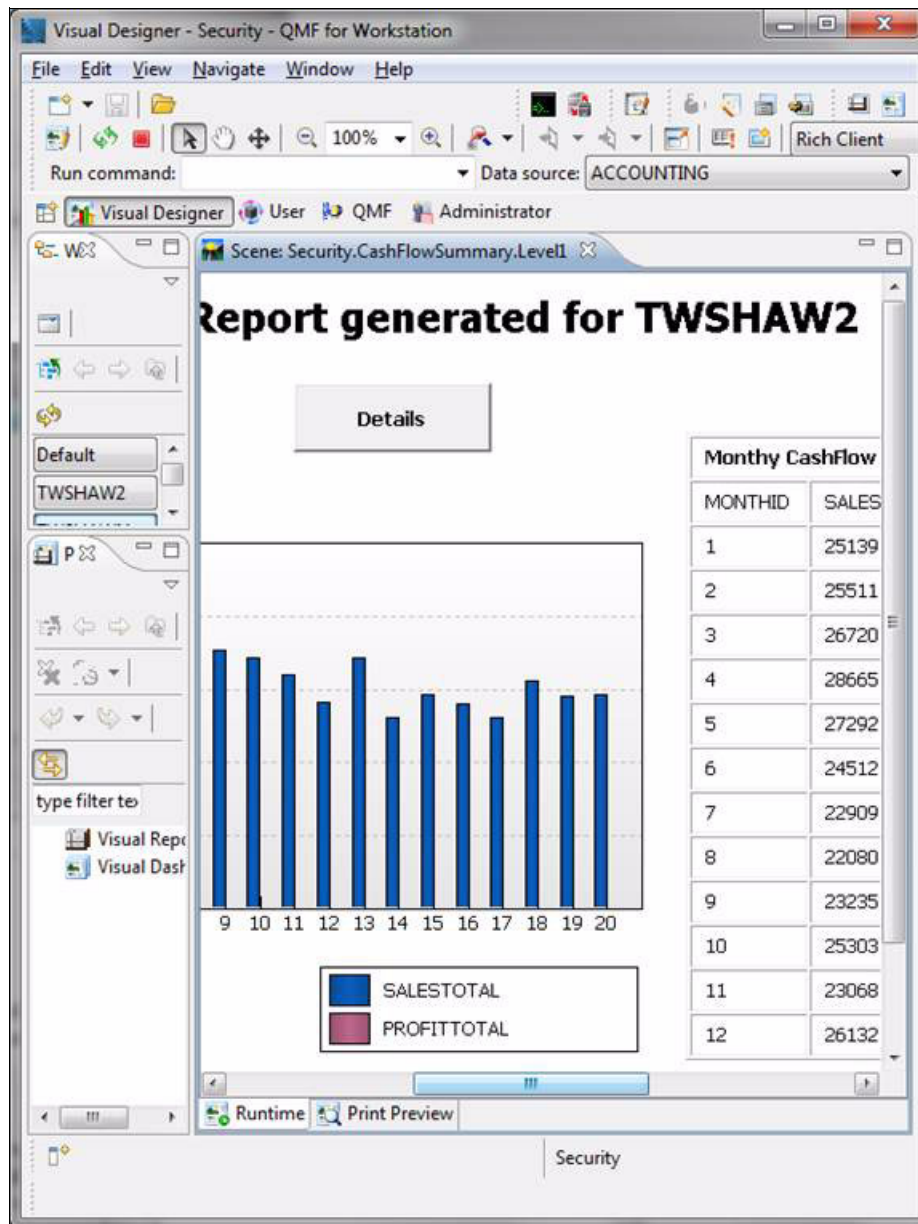


Figure 8-41 PROFITTOTAL now not shown to TWSHAW2

However, we see that, in the legend, PROFITTOTAL is still present. Every component of the chart that needs to be hidden must have the security function applied.

In the legend, we need to apply the same formula, shown again next, to the visibility of the label and the rectangle for PROFITTOTAL (Figure 8-42).

```
=if(isSecurityLevel("slAdmin"),1,0)
```

If you Ctrl-click to multiselect these two, you can paste the formula into the common **Visible** property and both will now be controlled by the expression.

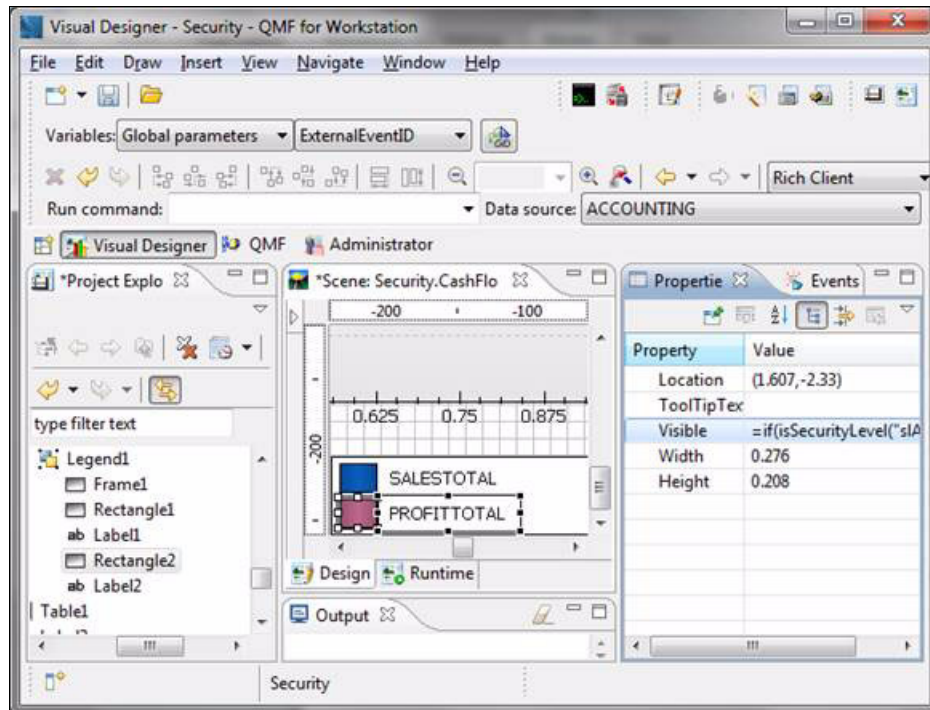


Figure 8-42 Using isSecurityLevel to hide additional components of the sample dashboard

Now the dashboard runs for TWSHAW2 without any trace of PROFITTOTAL in the chart (Figure 8-43).

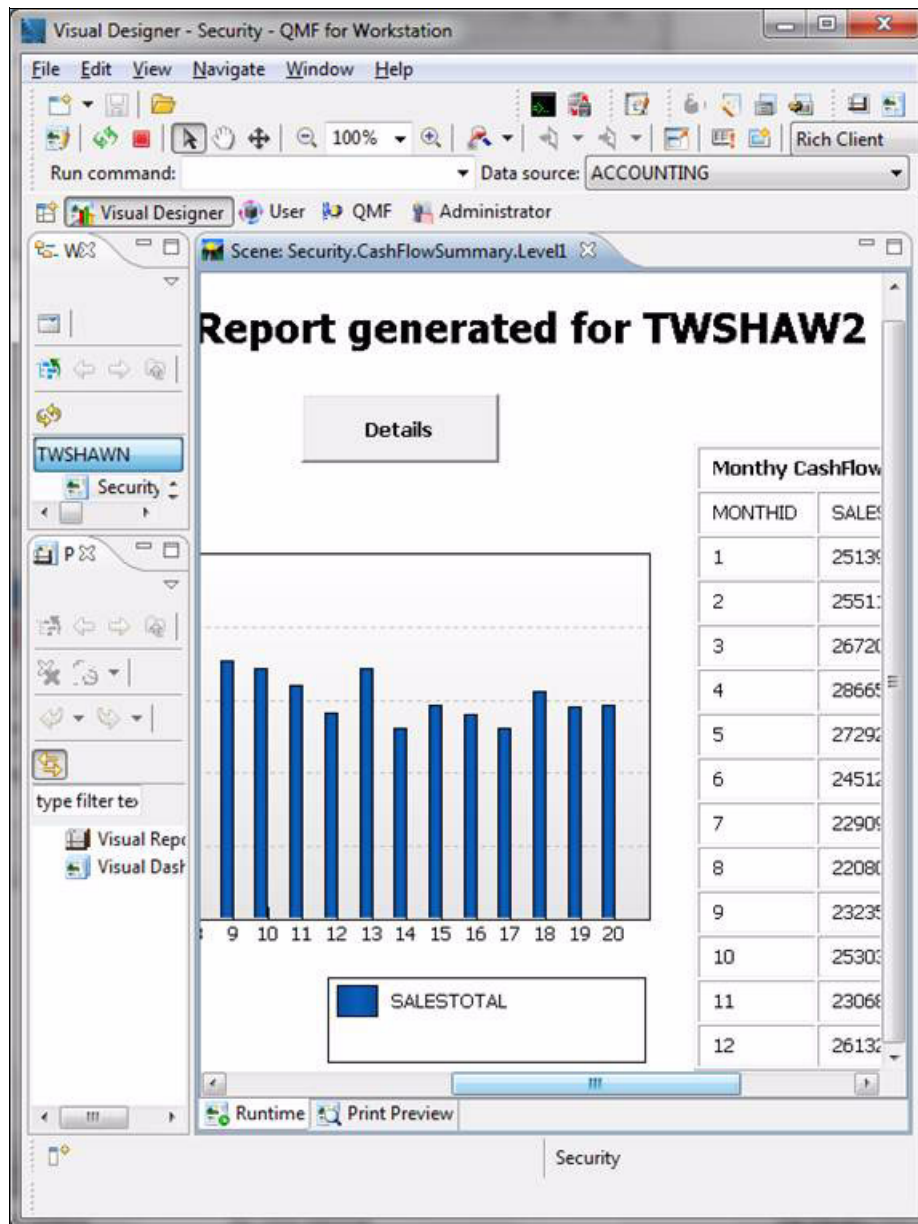


Figure 8-43 The dashboard with all security options set for user TWSHAW2

Restricting a field from a table by security would be more difficult because the table column has more individual elements that would need to be hidden. Instead, we apply the security at the row level.

8.3.2 Restricting fourth quarter rows from the table (row-level security)

The entry point to implement the desired security is at the table data template on the table row. If the same formula is used again, it would wipe out the entire table content, leaving only the header and title for users not in the slAdmin security list.

Therefore, the expression has to be modified to also use the values of the content of the row. To hide the last quarter, MONTHID = 10 or 11 or 12, the condition should be as follows:

```
=if(isSecurityLevel("slAdmin") | MONTHID < 10,1,0)
```

This means that, if the user is in the slAdmin security list or if the MONTHID is less than 10, then show the row. And if the user is not in slAdmin, only show the rows for MONTHIDs up to 10 and hide 10, 11, and 12 (Figure 8-44).

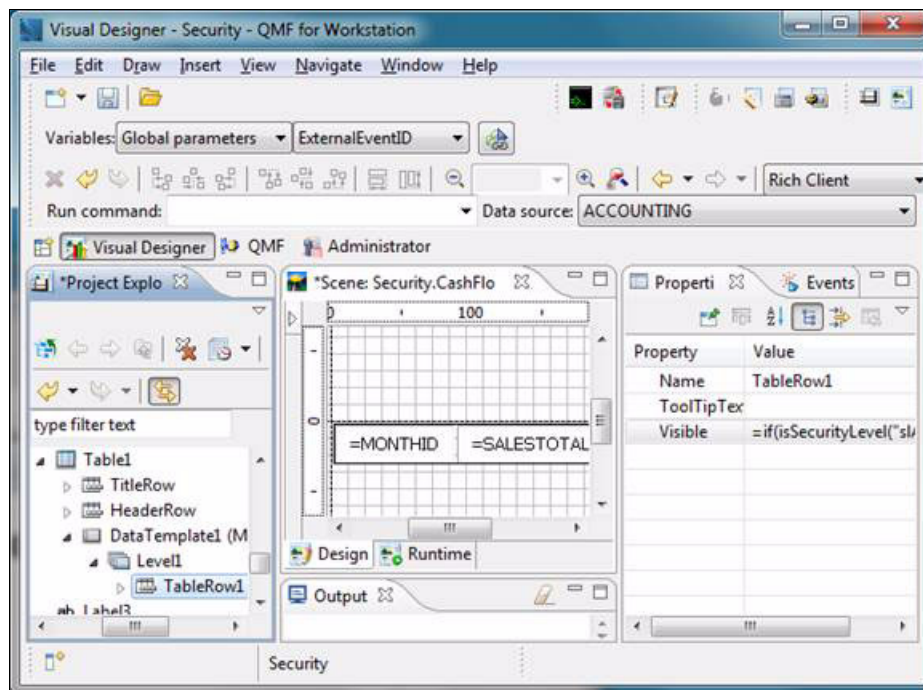


Figure 8-44 Specifying row-level security in the dashboard

Here is the dashboard for TWSHAW2 (Figure 8-45).

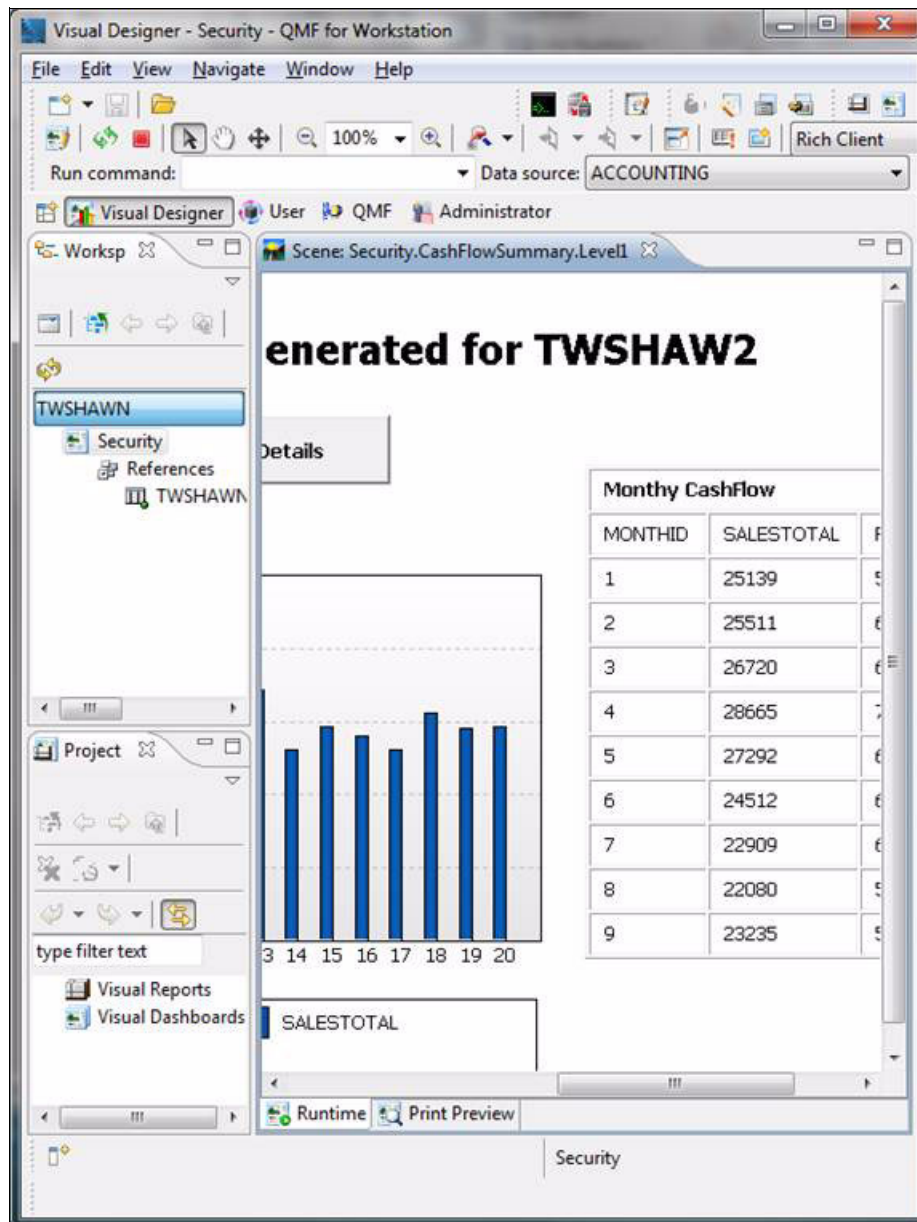


Figure 8-45 TWSHAW2 sees this dashboard

The final example for this function is the **Details** button, which handles navigation to the other scenes. The click event for this button has three “Jump to new location” actions (Figure 8-46).

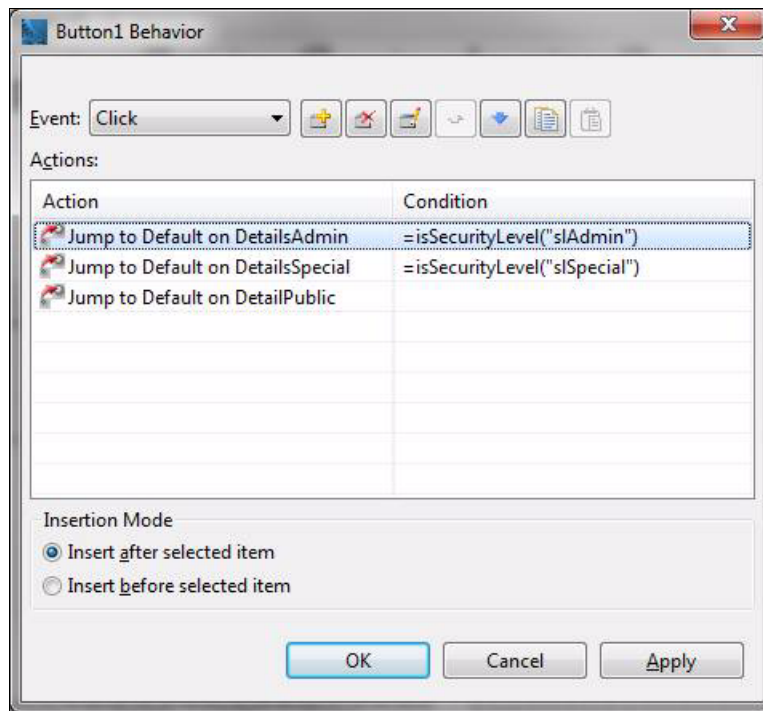


Figure 8-46 Click events for the Details button

In the Condition field, `isSecurityLevel` checks for membership in the given Security List in order to allow the navigation. Because this field is a Condition field and not a general field, such as Visible, you do not need to wrap the function in an “if” statement. Only users in the `slAdmin` list will go to the `DetailAdmin` scene, and so on.

Other uses of the `isSecurityLevel` function are driven by the need. While Visible is the most common use, it could also be used to set colors, fonts, or any other settable property that a particular dashboard calls for.



Getting to the data you need: Query methods

QMF provides a wide variety of options for disseminating business data. Features provided span the range of analytics output required today, from tabular and graphical reports to fully interactive dashboards. All of this content is underpinned by the QMF query facilities, which provide many different ways for both novice and experienced users alike to build queries.

This chapter covers how to use the various QMF query interfaces to access relational, hierarchical, and multidimensional data. We will also explain the many ways in which QMF queries can be used to transform the data further, drill down at deeper levels, and drive content in QMF reports, charts, graphs, dashboards, and third-party applications such as Microsoft Excel.

This chapter covers the following topics:

- ▶ Queries against relational data
- ▶ Queries against hierarchical data
- ▶ Transforming data using analytical queries
- ▶ What can I do with my query results?
- ▶ Queries against multidimensional data

9.1 Queries against relational data

There are a number of ways to create a new query, including:

- ▶ Browsing through your database structure and double-clicking a particular table
- ▶ Selecting **File** → **New** → **Query** from the menu or clicking the **New Query** icon in the toolbar
- ▶ Clicking the **Draw Query** icon in the toolbar
- ▶ Clicking tables that have been arranged in an arbitrary folder structure in your QMF workspace
- ▶ Using the QMF command bar to directly display a given table with a default query

In this case, we will start by using the QMF command bar. To display the command bar if it is not already shown, click the **Show Command Bar** icon in the toolbar, as shown in Figure 9-1.

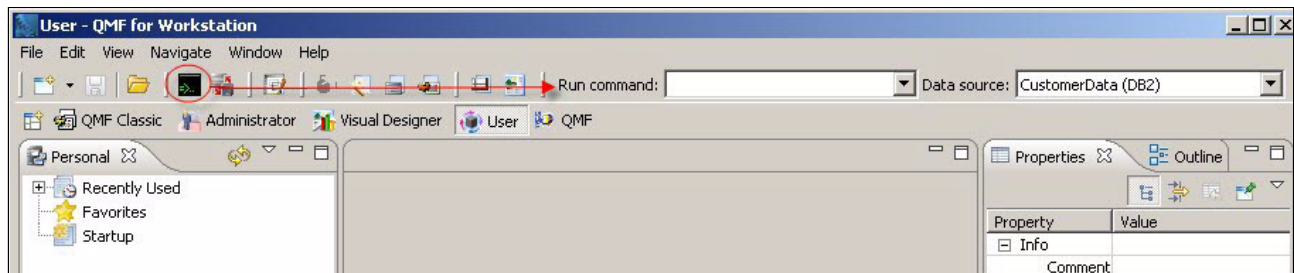


Figure 9-1 The Show Command Bar icon and the QMF command bar

The command bar accepts QMF procedure commands. For experienced QMF users, it is perhaps the most efficient means of accessing a QMF object (query, report, or procedure) or a database table. The command bar includes an area for the command itself, along with a means of selecting the server against which the command will run.

You only need to type enough letters of the name of the command to allow QMF for Workstation to distinguish it from other procedure commands. We will use the **DISPLAY** command, but we only need to enter “DI” because **DISPLAY** is the only command that starts with those letters. If there were, for example, a command called **DIRECT**, the user would have to enter “DIS” to differentiate the **DISPLAY** from the **DIRECT** command.

In this example, we will develop a query that uses the **Q.STAFF** and **Q.ORG** tables, located in Spiffy Insurance's **CustomerData** database in **DB2**. We'll start by displaying the first table, **Q.STAFF**. To access the table, enter the following into the **Run command** field:

```
di Q.STAFF
```

QMF for Workstation creates a default query from the **DISPLAY** command and runs it, as shown in Figure 9-2 on page 193.

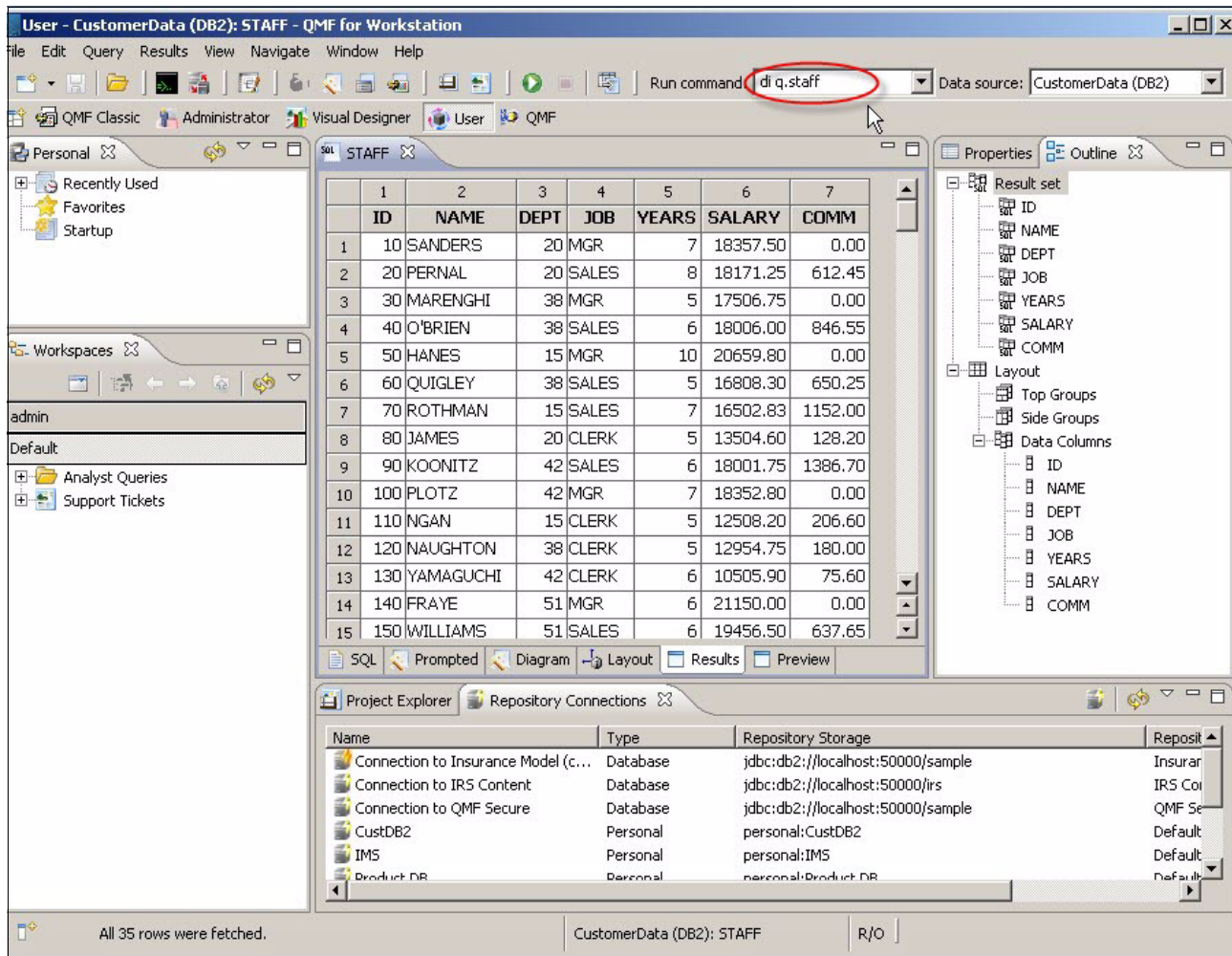


Figure 9-2 Results of the `di q.staff` command

Now you see six tabs beneath the results:

- ▶ SQL
- ▶ Prompted
- ▶ Diagram
- ▶ Layout
- ▶ Results
- ▶ Preview

The Diagram, Prompted, and SQL tabs provide three different query editing interfaces to assist in developing queries.

Tip: Actions taken in one tab automatically appear in the query views shown in the other tabs, allowing users to develop the query using a blend of the three query editors if necessary. However, certain changes made to the query text in the SQL editor (such as unions and subselects) might render the query unsuitable for the prompted or diagram editors. In these cases, the query will remain editable in the SQL view only.

The next three topics explain the diagram, prompted, and SQL editors in more detail.

9.1.1 Developing queries using the query diagram view

The **Diagram** tab provides access to the *query diagram view*. It is perhaps the most familiar query development style for end users. The view consists of an upper area that contains the table diagram, a center area that reflects the selected columns, and a lower area that summarizes the query's conditional statements.

Let's apply a number of edits to the default query, starting with the addition of our second table, Q.ORG. Like Q.STAFF, Q.ORG is a QMF sample table shipped with the product. Add the Q.ORG table to the query by locating it in the workspace and dragging and dropping it into the diagram's table area, as shown in Figure 9-3.

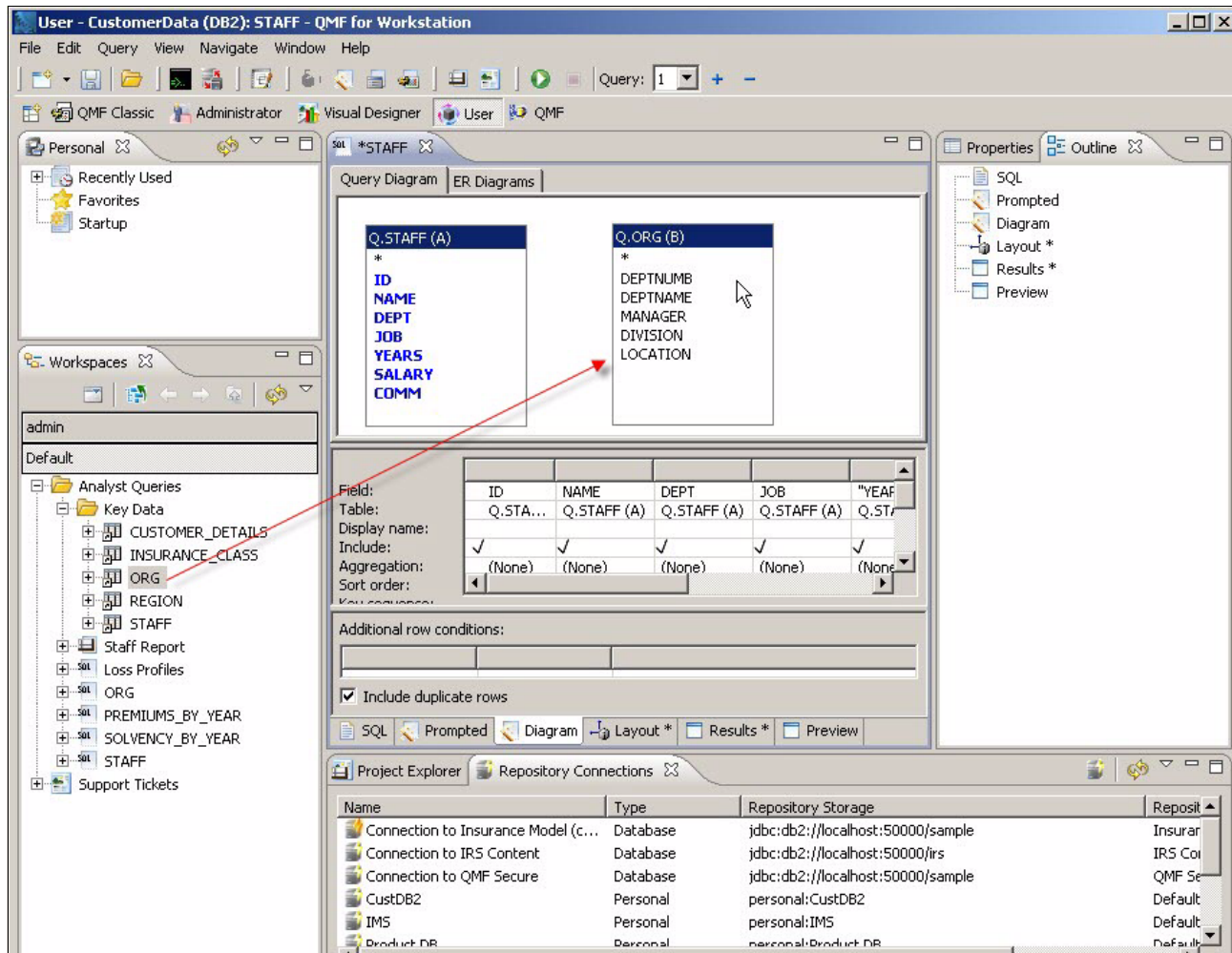


Figure 9-3 Adding a table to a query

After the Q.ORG table has been added, a connection (a join) between it and the Q.STAFF table can be defined by dragging the respective column of Q.STAFF onto the corresponding column in Q.ORG (or vice-versa). This action automatically defines a join between the two tables, as shown in Figure 9-4 on page 195.

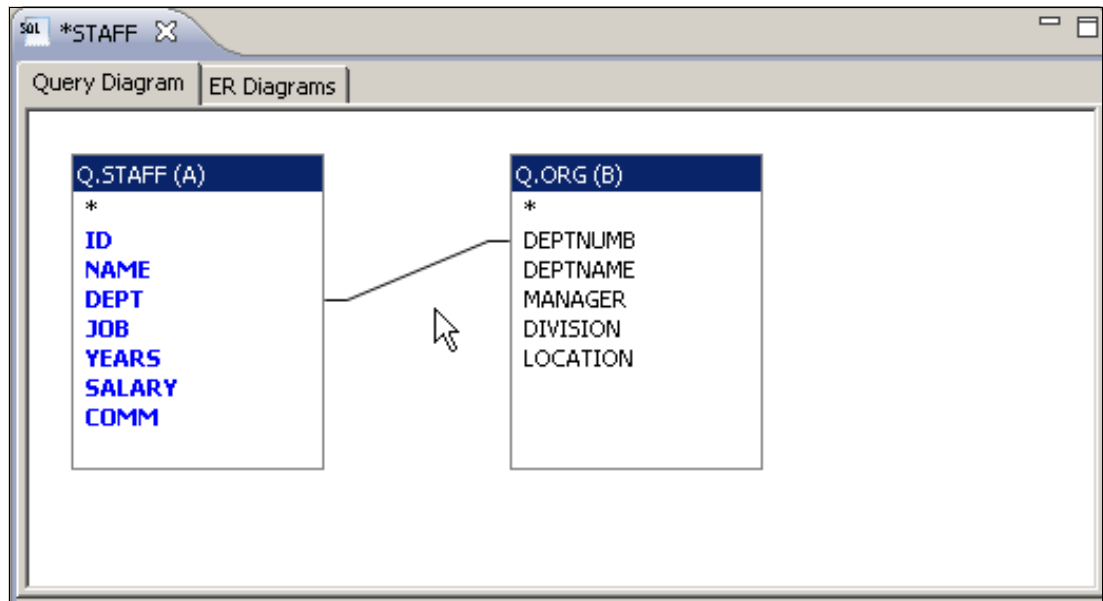


Figure 9-4 Joining the two tables

By default, the join type is set to an inner join. However, this can be changed by right-clicking the join connection and selecting **Change Join** from the context menu. (This menu also provides a means of removing the join.) The Join Tables window is displayed, as shown in Figure 9-5.

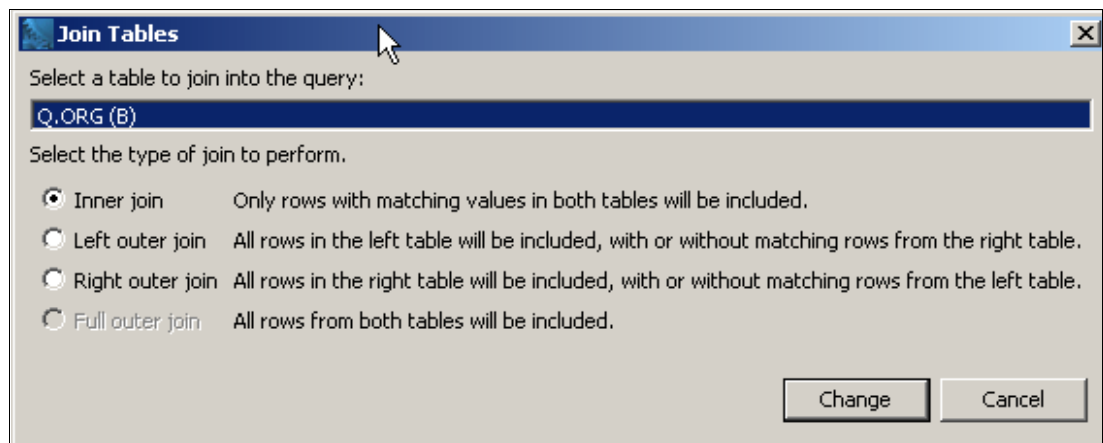


Figure 9-5 Join Tables window

With the Q.ORG table in place, its columns can be added to the query by double-clicking each desired column name. In this case, we double-click the DEPTNAME column to add it to the query. Double-clicking a column toggles between adding and removing the column. Thus, double-clicking a column that is already included in the query will remove it from the query. Using this approach, we add DEPTNAME and remove the ID and DEPT columns, as shown in Figure 9-6 on page 196.

The diagram view's column area allows conditions, ordering, and aggregates to be applied to specific query columns. In this case, we will use the **Sort** order row to sort the results first by DEPTNAME, then by JOB. It can be done by clicking in the cells, as shown in Figure 9-6 on page 196.

The key sequence indicates the sorting order. Sorting follows the column order in which the ordering is applied. It can be changed by clicking the key sequence field and selecting an alternate order for a given column.

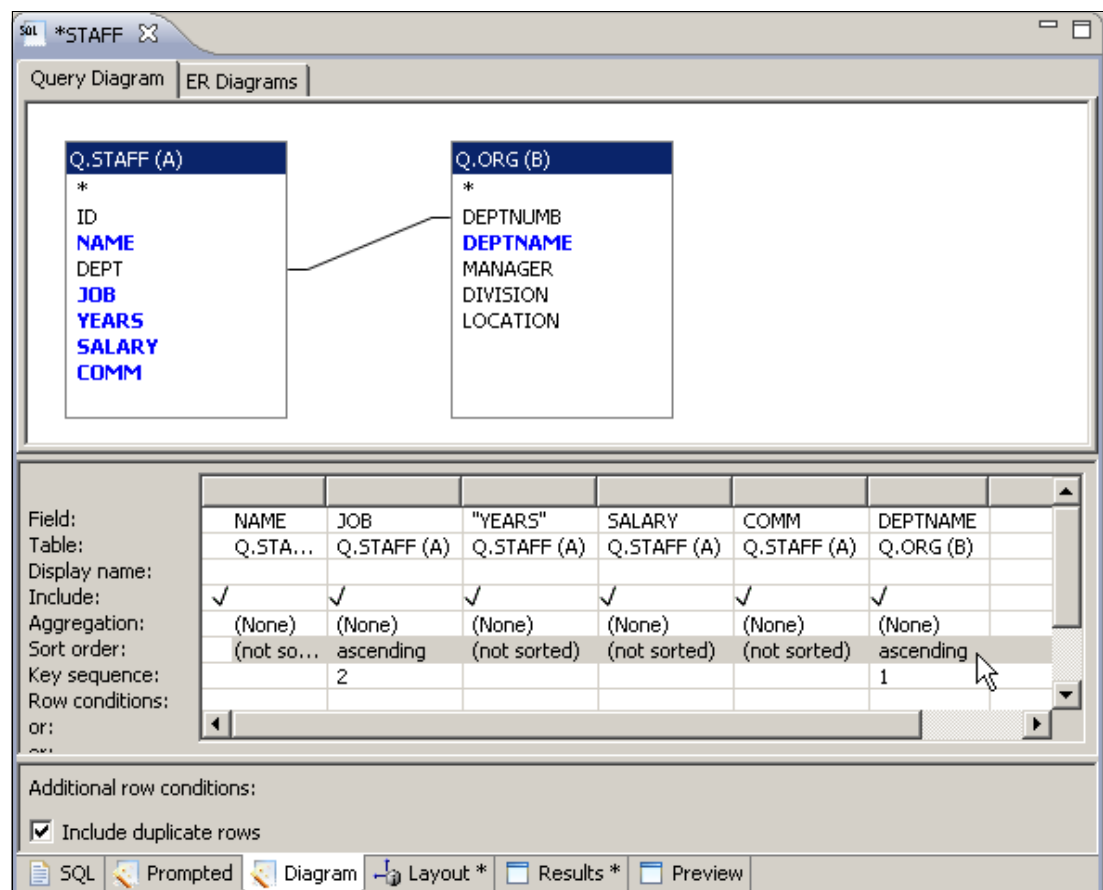


Figure 9-6 Query column selections and sort order

You can select aggregate functions for a given column (such as sum, max, min, average, and so on) as well as apply a row condition. For example, we could constrain the results to employees with more than five years of service by entering the following into the Row conditions cell for the YEARS column:

>5

If we want an exact match, we would drop the > operator and simply enter 5 into the field, which implies an "equal to 5" statement. In our example, we will leave the results as-is, with no conditions applied.

The query can be run by clicking the Run Query icon in the toolbar, as shown in Figure 9-7 on page 197, or by selecting **Query** → **Run** from the menu.



Figure 9-7 Run Query toolbar icon

After the query runs, its output is displayed in the Results tab. Users can return to the **SQL**, **Prompted**, or **Diagram** tabs as needed to further refine the query. Changes to the query are only applied when the query is re-executed.

9.1.2 Developing queries using the prompted query view

The Prompted tab, shown in Figure 9-8, provides a prompted query view that allows users to methodically build a query by focusing on each of the five aspects of the query, one at a time:

- ▶ The tables from which data will be drawn
- ▶ The relationships (joins) between the tables
- ▶ The columns to be returned from the query
- ▶ Sort conditions to be applied to one or more result set columns
- ▶ Query conditions to be applied to one or more table columns

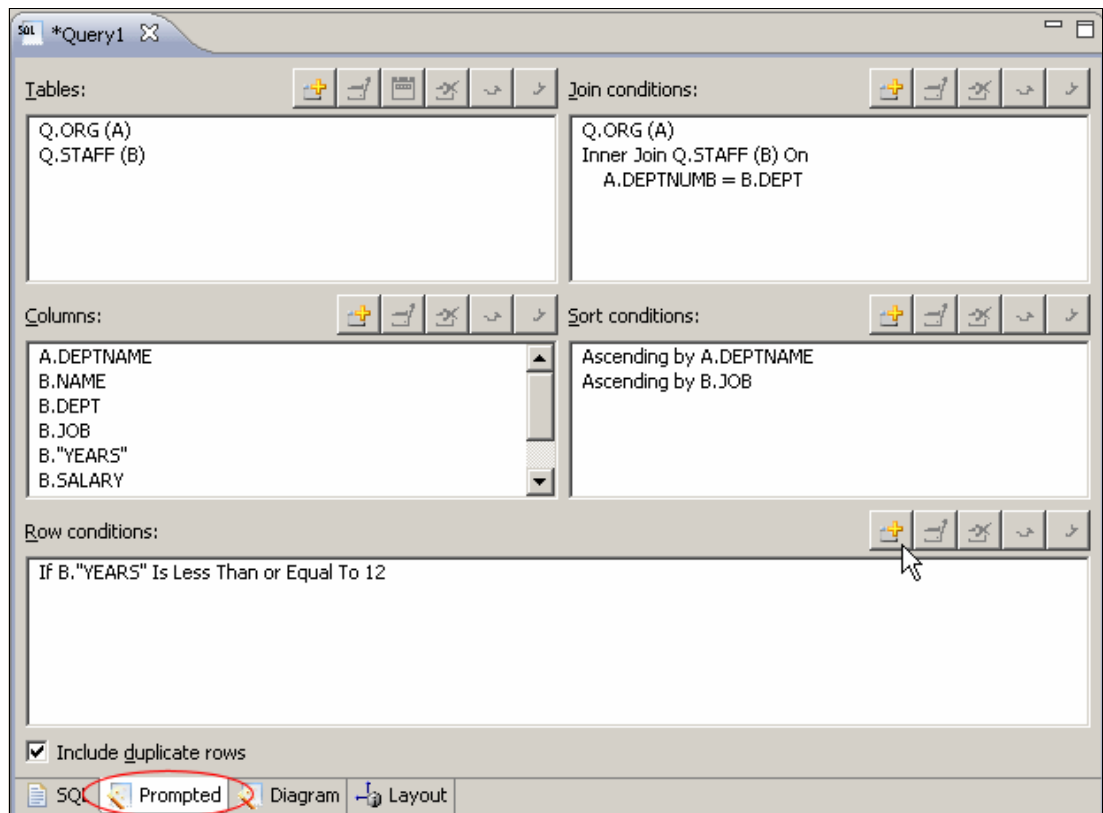


Figure 9-8 Developing a query in the prompted query view

The five sections to develop a query are laid out across the prompted query view, starting on the top left and working across the view. Each of the five sections includes a mini toolbar with the ability to add, edit, or delete items, as well as move them up or down in the listing. When adding elements to each of the five sections, the user is presented with a supporting dialog box with clear and simple options related to the section of the query.

For example, when defining the relationship between the tables added to the query (something that is often difficult for non-technical users to express), the prompted query view presents the window shown in Figure 9-9.

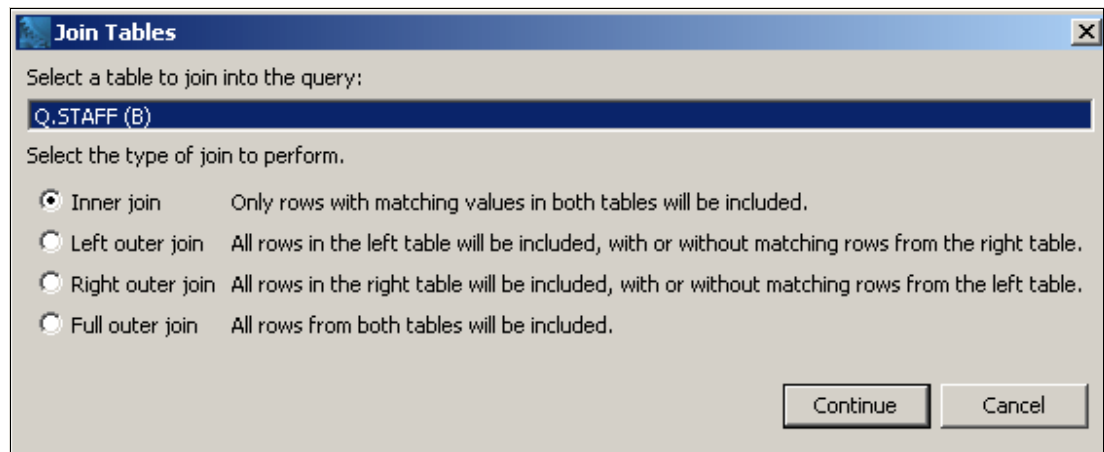


Figure 9-9 Defining the join condition between tables in the query

Similarly, when defining the sort conditions to be applied to one or more columns in the query result set, the prompted query view presents the options shown in Figure 9-10 on page 199.

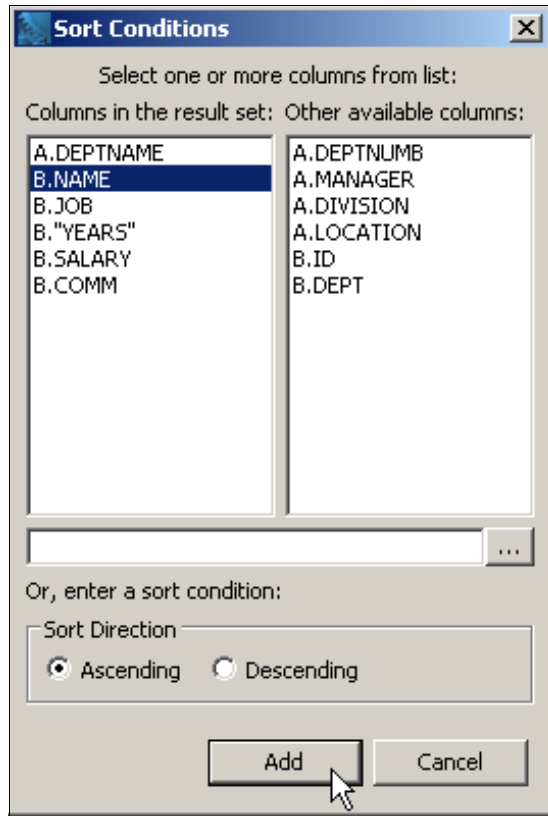


Figure 9-10 Adding sort conditions to query columns

Any number of row conditions can be defined using the last section in the prompted query view, which provides a wide variety of options, as shown in Figure 9-11 on page 200. Column conditions can comprise fixed values (such as YEARS less than or equal to 12) or a set of values (such as DEPTNUMB is equal to 10, 20, or 56). Possible matching values can also be supplied by a CSV file or by running a secondary query.

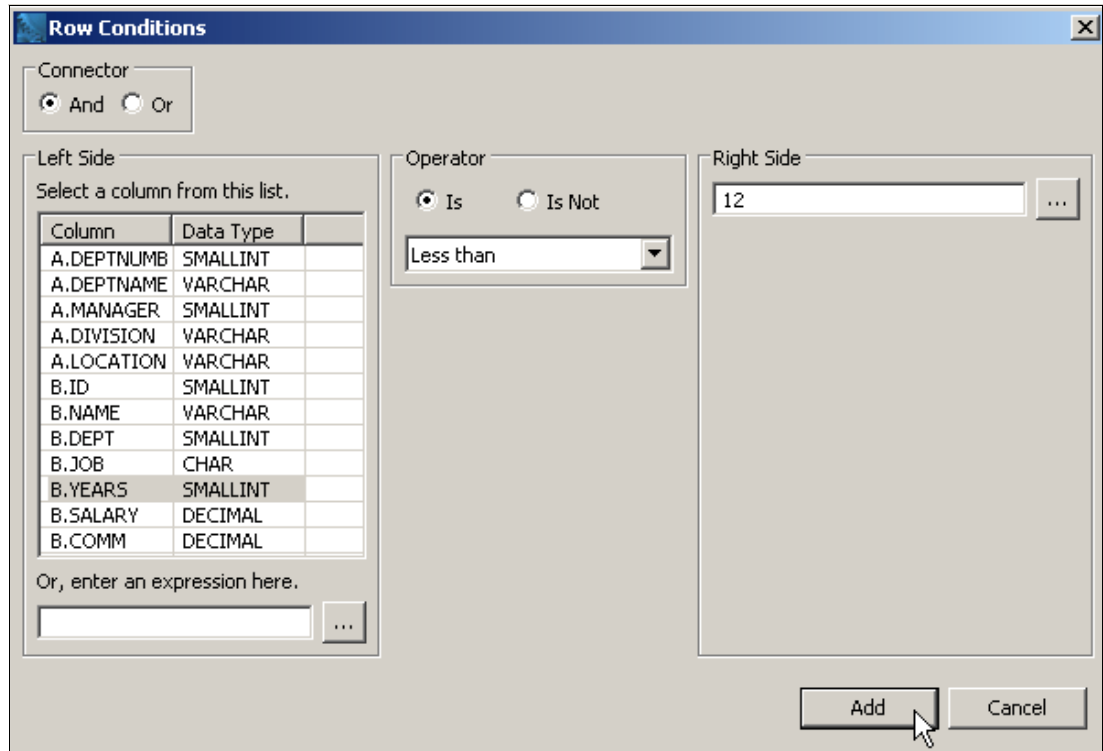


Figure 9-11 Adding row conditions using the prompted query view

When defined, the query can be run using the same methods outlined previously.

9.1.3 Developing queries using the SQL view

Users who are familiar with SQL statement syntax have the option of developing or refining queries using the SQL editor, which is accessed by clicking the SQL tab at the base of the query window. For example, a user might want to type the following in the SQL editor:

```
SELECT * FROM Q.STAFF
```

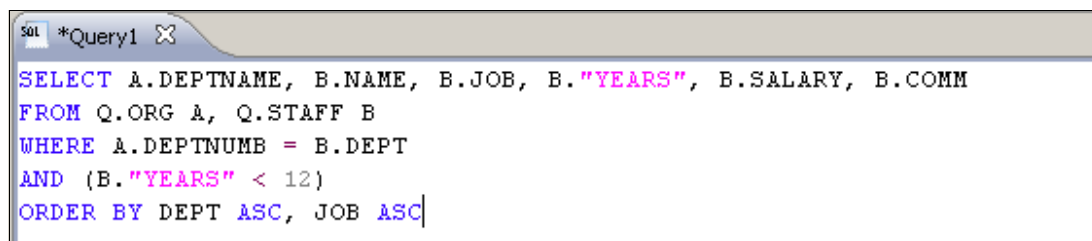
The user can then add sorting and column aggregations by the Prompted or Diagram tabs before returning to the SQL editor to finalize the query by adding row conditions. It allows QMF users to move among the three query editors at any point, giving them the ultimate flexibility in the development process. It also provides a means for a more technical user to assist and embellish queries for others who might have started a query but require more sophisticated functions.

The QMF SQL editor is also capable of handling more complex query structures that are not supported by the diagram and prompted views. For example, queries that utilize unions or extensive nested SQL (subqueries) can only be developed in the SQL view.

The SQL editor offers a number of productive features, including:

- ▶ Colorization of query commands and constants, increasing the clarity of SQL statements
- ▶ Built-in query reformatting (invoked by the user), which assists in methodically laying out a query
- ▶ Ability to comment and uncomment multiple rows of query text with a single menu action

Figure 9-12 shows the query we developed earlier in the SQL editor. The SQL statements and keywords are colorized in blue.



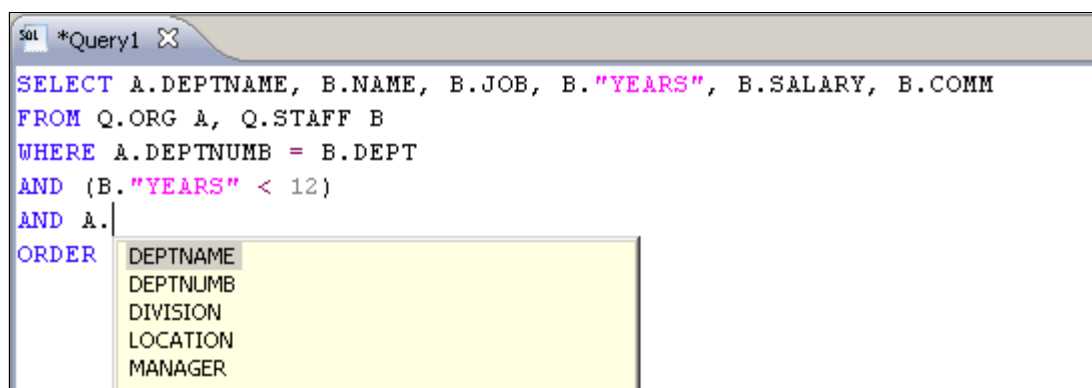
```

SELECT A.DEPTNAME, B.NAME, B.JOB, B."YEARS", B.SALARY, B.COMM
FROM Q.ORG A, Q.STAFF B
WHERE A.DEPTNUMB = B.DEPT
AND (B."YEARS" < 12)
ORDER BY DEPT ASC, JOB ASC

```

Figure 9-12 Entering SQL directly into a QMF query

The SQL editor includes a content-assist feature, allowing the user to quickly refer to columns in a given table without the need to remember their names. As the query is typed, pressing Ctrl+Enter either completes the item under edit (if there are no ambiguities) or presents a list of potential matches to the item under edit. Figure 9-13 shows this feature in action. In this example, content assist has been invoked while adding a condition to the query. Content assist displays all possible columns that can be selected from the table that has been aliased to A.



```

SELECT A.DEPTNAME, B.NAME, B.JOB, B."YEARS", B.SALARY, B.COMM
FROM Q.ORG A, Q.STAFF B
WHERE A.DEPTNUMB = B.DEPT
AND (B."YEARS" < 12)
AND A.
ORDER

```

DEPTNAME
DEPTNUMB
DIVISION
LOCATION
MANAGER

Figure 9-13 Invoking content assist while typing a query in the SQL editor

The SQL editor provides a productive means for power users to create or modify a QMF query. In the next topic, we'll show you how the SQL editor can be used to refine a query that calls a database stored procedure.

9.1.4 Calling stored procedures from QMF queries

QMF for Workstation and QMF for WebSphere support calls to database stored procedures from regular QMF queries. A stored procedure can return an unlimited number of result sets, each of which is available for export or to serve as the basis for report or dashboard development.

For example, Spiffy's HR department uses a stored procedure to apply a sales bonus across a group of sales teams. This stored procedure, `BONUS_INCREASE`, is displayed in the workspace explorer, much the same as database tables. Upon double-clicking or opening the procedure, QMF for Workstation automatically generates the SQL required to call it, as shown in Figure 9-14 on page 202.

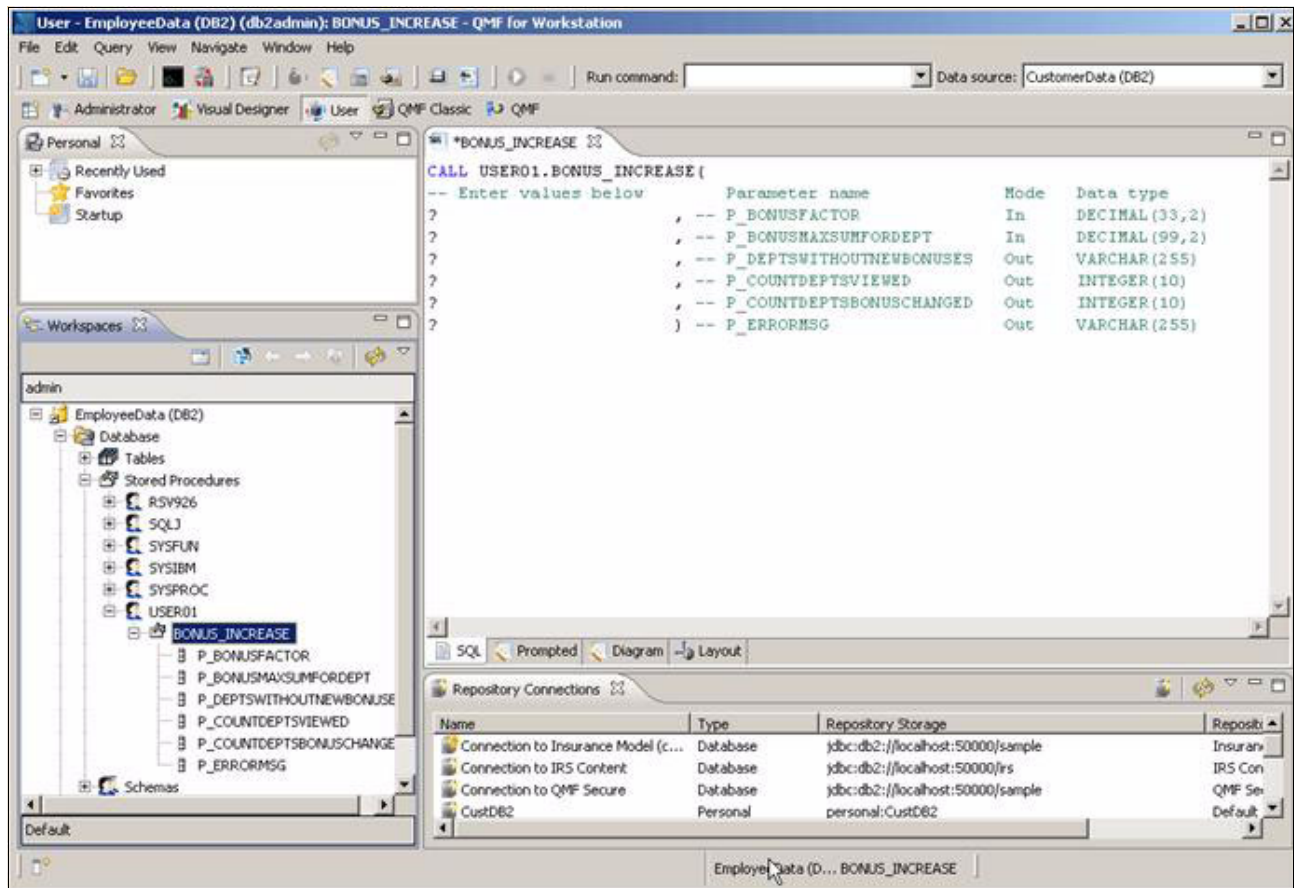


Figure 9-14 SQL automatically generated by QMF to call a database stored procedure

This auto-generated query also includes a summary of the parameters that must be supplied to the procedure. In this case, Spiffy's HR personnel supply the bonus factor and maximum bonus per department values by replacing the question marks with values, as shown in Figure 9-15.

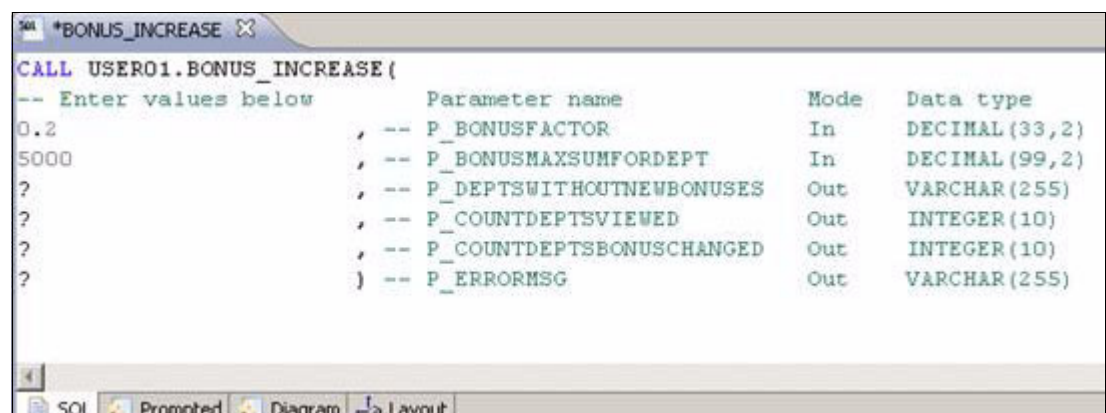


Figure 9-15 Adding parameter values to a stored procedure call

The remaining four parameters are supplied by the stored procedure. (Note how QMF automatically summarizes the parameter names, type, and whether or not they are input or output parameters.)

The procedure is executed by running the query. QMF presents a summary of the parameter values, as shown in Figure 9-16, allowing them to be changed if needed. If preferred, parameter values can be input at this stage, rather than by changing the SQL itself, as we did in Figure 9-15 on page 202.

The dialog box titled "Confirm Stored Procedure Parameters" displays a table of parameters for the stored procedure USER01.BONUS_INCREASE. The table has four columns: Name, Type, Mode, and Value. Six parameters are listed, each with a checked checkbox in the first column. The parameters are: P_BONUSFACTOR (DECIMAL (3, 2), In, 0.2), P_BONUSMAXSUMFORDEPT (DECIMAL (9, 2), In, 5000), P_DEPTSWITHOUTNEWBONUSES (VARCHAR (255), Out, -), P_COUNTDEPTSVIEWED (INTEGER, Out, -), P_COUNTDEPTSBONUSCHANGED (INTEGER, Out, -), and P_ERRORMSG (VARCHAR (255), Out, -). Below the table are OK and Cancel buttons.

Name	Type	Mode	Value
<input checked="" type="checkbox"/> P_BONUSFACTOR	DECIMAL (3, 2)	In	0.2
<input checked="" type="checkbox"/> P_BONUSMAXSUMFORDEPT	DECIMAL (9, 2)	In	5000
<input checked="" type="checkbox"/> P_DEPTSWITHOUTNEWBONUSES	VARCHAR (255)	Out	-
<input checked="" type="checkbox"/> P_COUNTDEPTSVIEWED	INTEGER	Out	-
<input checked="" type="checkbox"/> P_COUNTDEPTSBONUSCHANGED	INTEGER	Out	-
<input checked="" type="checkbox"/> P_ERRORMSG	VARCHAR (255)	Out	-

Figure 9-16 Stored procedure parameter summary dialog, which is presented prior to calling the procedure

After QMF confirms the parameter values, it runs the stored procedure and returns the output data in the results grid, as shown in Figure 9-17 on page 204. In this case, the user is asked to confirm the database updates, because the procedure applies bonus values to database records. Spiffy's HR personnel have the ability to review the returned data and roll back the action by clicking Cancel if they find that the input parameter values were not what they wanted or were entered incorrectly.

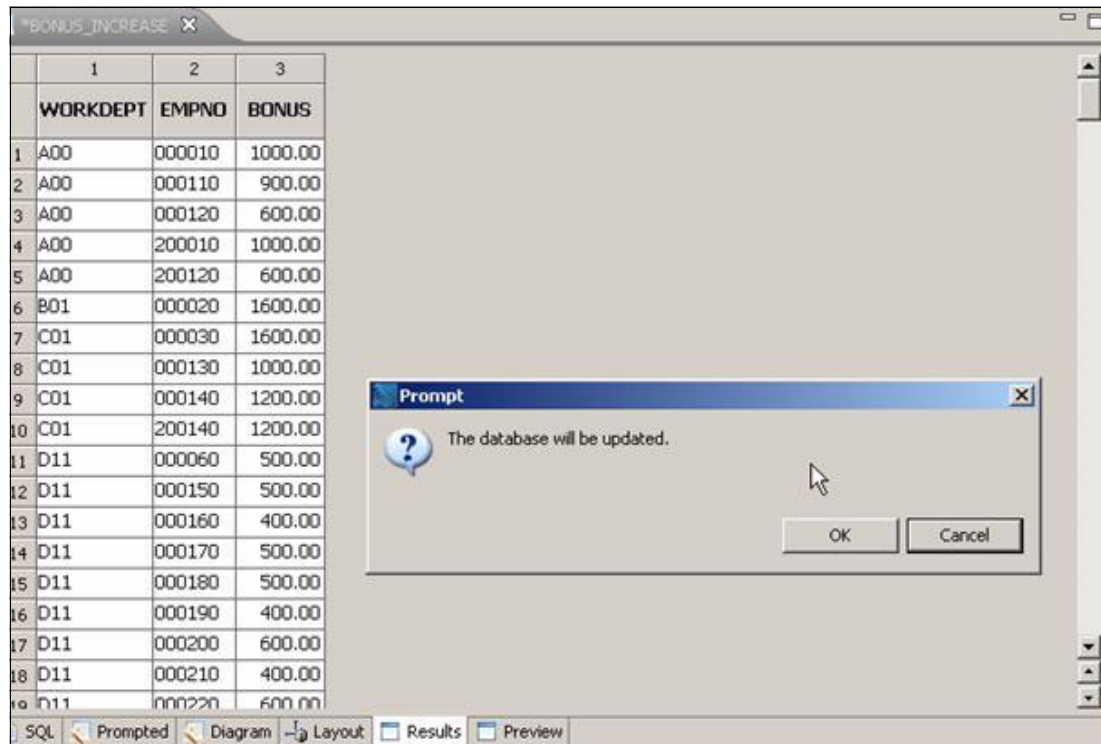


Figure 9-17 Returned data from the stored procedure, with the ability to roll back proposed updates

9.2 Queries against hierarchical data

Beginning with IMS 11 and continuing with IMS 12, a new set of possibilities for providing analytics directly within IMS was introduced in the form of open database access with JDBC, which is how QMF supports IMS directly. In Chapter 6, “Configuring access to data sources and populating user workspaces” on page 89, we looked at how to add an IMS data source within QMF. After the data source has been defined, QMF sees the IMS system as nothing more than a relational data source. Tables appear in the navigation tree as they do for relational data sources, and queries against IMS data can then be built using the diagram, prompted, or SQL editors explained earlier, just as they can for relational data.

9.3 Transforming data using analytical queries

Using QMF analytical queries, you can transform and merge data in a variety of ways, including:

- ▶ Joining disparate queries together using append and join operations, including queries that are sourced from independent databases
- ▶ Cross-tabulating one or more columns in the result sets across and down two or more result set columns
- ▶ Conditionally cross-tabulating columns, which is much like regular cross-tabulation explained before, but with the ability to place conditions on the rows to be tabulated

For example, a conditional cross-tabulation can sum salaries by job for those who have served less than 5 years and another can be done for those who have served more than

10 years. For comparison, a third can be added for all staff members, regardless of service years.

► Normalizing data from column to row layouts

For example, a result set that reports sum of sales by month for four products in a column layout can be transformed to a row layout, as shown in Figure 9-18.

Original result set layout					Transformed result set layout		
Month	ProdA	ProdB	ProdC	ProdD	Month	Product	Sales
Jan	100	120	80	200	Jan	ProdA	100
Feb	500	650	890	180	Jan	ProdB	120
					Jan	ProdC	80
					Jan	ProdD	200
					Feb	ProdA	500
					Feb	ProdB	650
					Feb	ProdC	890
					Feb	ProdD	180

Figure 9-18 Sum of sales by month for four products

To show an example of how to transform data using analytical queries, we will start with two independent queries, one drawing data from the Q.STAFF sample table and one drawing data from the Q.ORG sample table. Clearly, we could have used a single query to join these two tables together, because they are in the same database. However, the point of this example is that the two tables need not be in the same database.

In our scenario, we actually have the Q.ORG table in DB2 and the Q.STAFF table in SQL Server to illustrate the federation capabilities in QMF. So here we need two independent queries.

After the two queries have been created, we create an analytical query to join them together. It is done by clicking the **New analytical query** icon in the toolbar, as shown in Figure 9-19 on page 206, or following the **File** → **New** menu path.

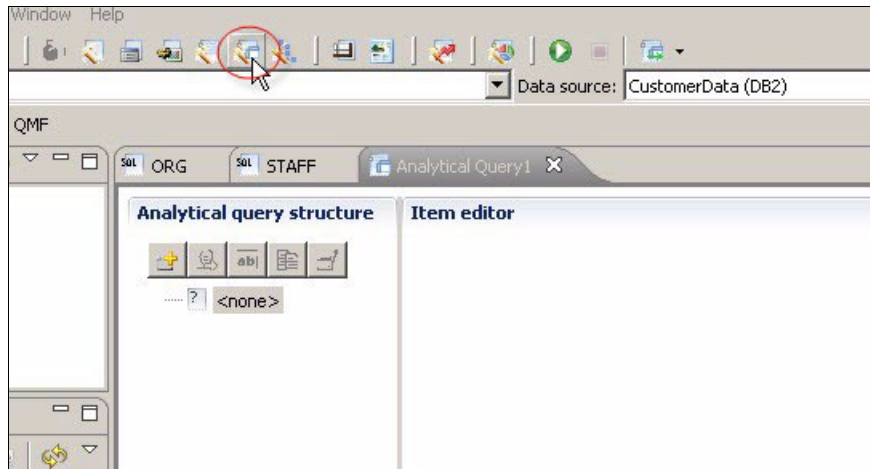


Figure 9-19 Creating a new analytical query

An analytical query can perform a number of actions, as mentioned earlier. Upon creating a new query, these actions can be set by clicking the **Add** icon in the toolbar (or by right-clicking the <none> tree item). The resulting context menu offers various options, as shown in Figure 9-20.

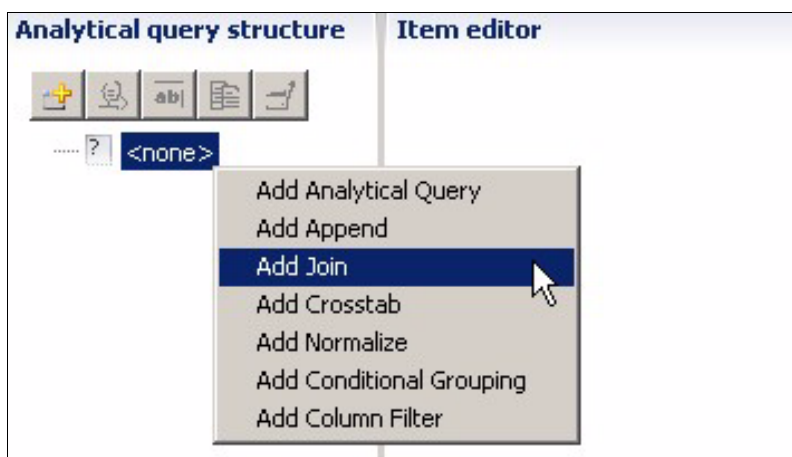


Figure 9-20 Selecting the function for the analytical query

In our case, we select a join query, as shown before. When selected, the query presents placeholders for each side of the join, as shown in Figure 9-20. Queries can be assigned to each placeholder by right-clicking them. Because we just created the two source queries, we can select **Add from opened** for each of them, as shown in Figure 9-21 on page 207.

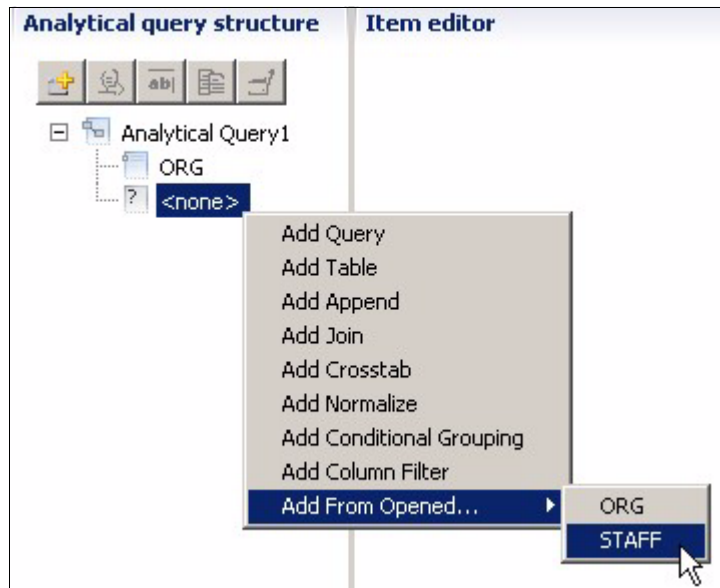


Figure 9-21 Adding the source queries

Now that the two source queries have been added, we need only define how the two data sets are to be joined, followed by the columns that we want to be returned in the joined result set. To define the join, we select the corresponding columns and then click the **Add Join Key** icon, which is highlighted in Figure 9-22.

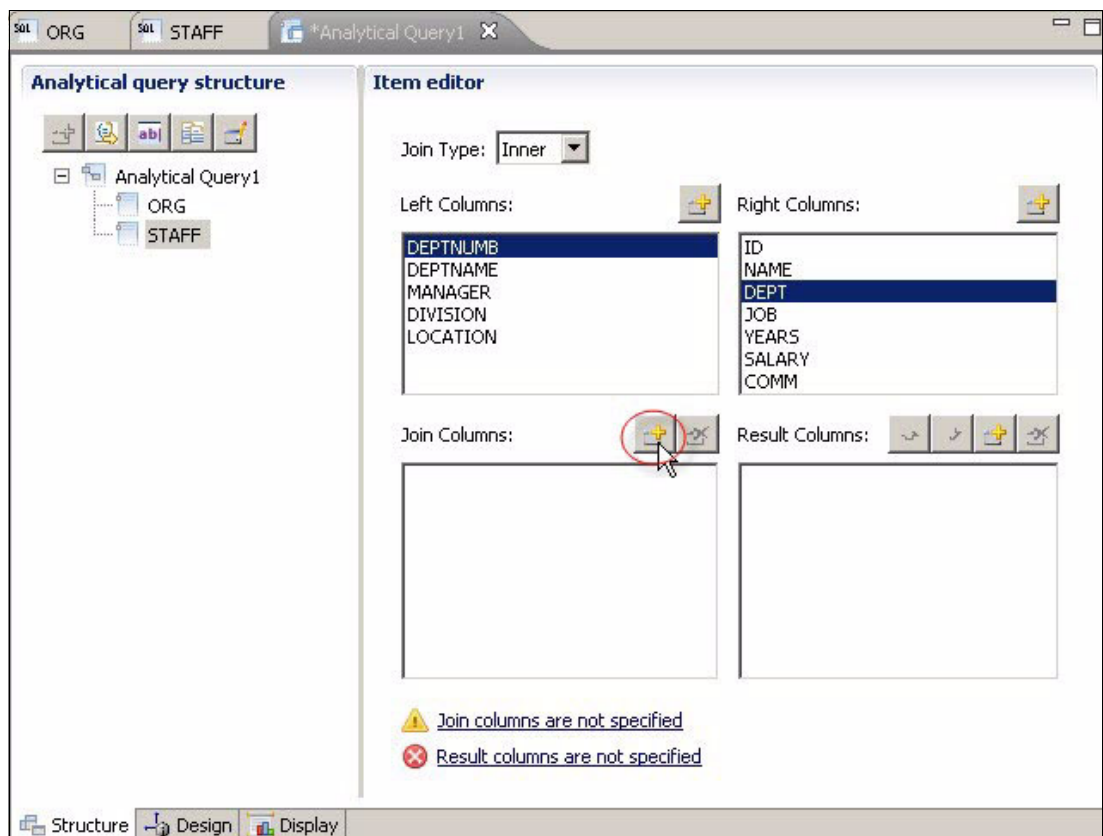


Figure 9-22 Defining the join conditions in an analytical query

When the join conditions have been defined and the result columns selected, the query is ready to be run (Figure 9-23). Notice that the original source queries can now be discarded because the analytical query stores its own copy of them. These queries can be further edited by right-clicking them in the navigation tree and selecting **Open in separate editor**. Changes to the query are automatically saved back into the parent analytical query.

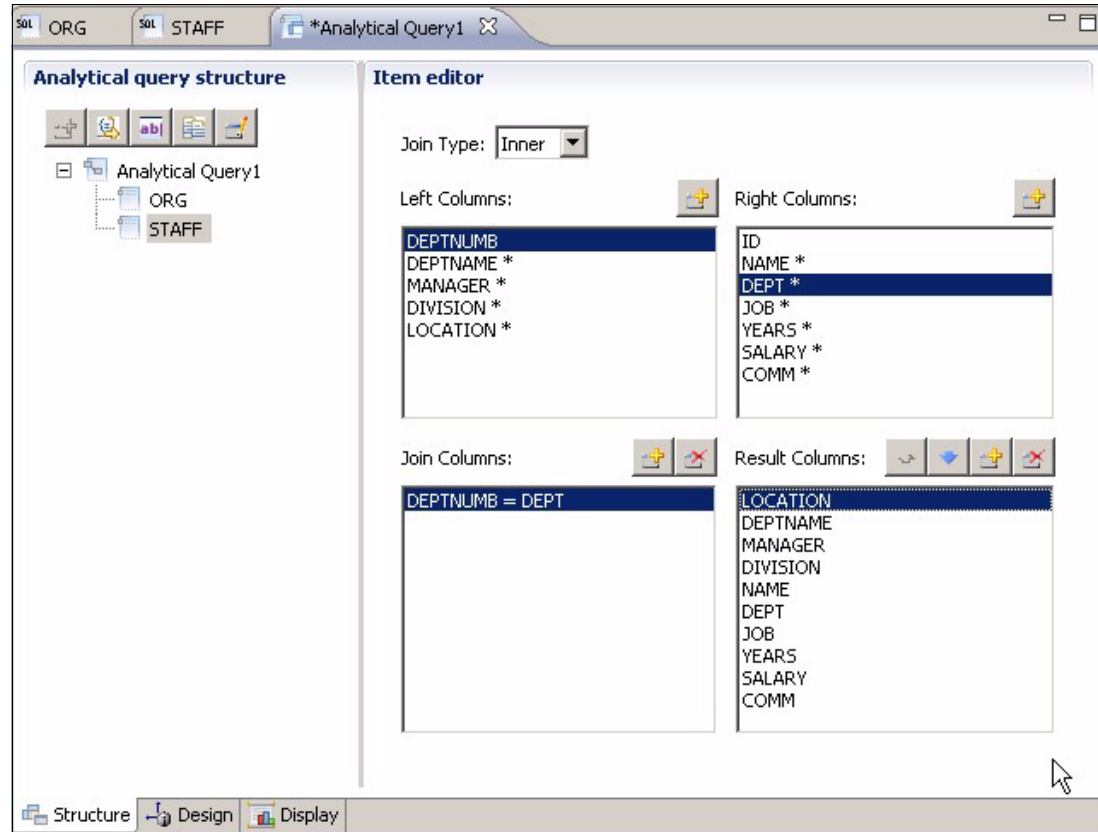


Figure 9-23 Defining the join conditions in an analytical query

Upon running this new analytical query, QMF independently runs the two source queries and assembles the two data sets into a single result set using the join defined in Figure 9-22 on page 207. This object is now just another QMF query to the end user. It can be used as the basis of a QMF form or report, or used within a dashboard to present content in a grid or chart. In addition, the blended result set can be joined or appended to another query, which itself can be a simple or analytical query.

We'll now take the result of the join query and use another analytical query to cross-tabulate the result. As before, we open a new analytical query (Figure 9-19 on page 206) and use the context menu (Figure 9-20 on page 206) or application menu to set it to a cross-tabulation. For a cross-tabulation, we need only select the source query to be tabulated. We will select **Add from Opened** and select the open analytical query that we created earlier, as shown in Figure 9-24 on page 209.

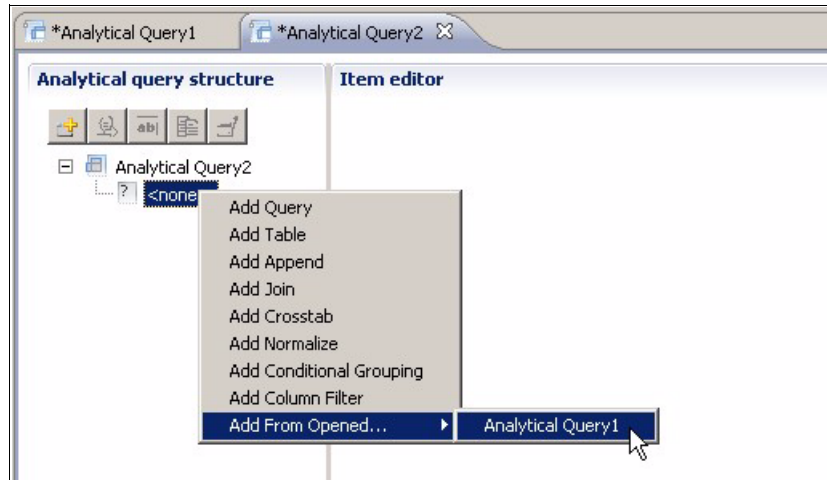


Figure 9-24 Cross-tabulating an analytical query

Any query can be cross-tabulated, whether it is an analytical query or a simple query that directly retrieves data from a single source. The cross-tabulation options allow users to select one or more grouping rows, the cross-tabulation column, and the values to be tabulated. In this example, we've elected to compute a sum of the salary for each division, across each job type (Figure 9-25).

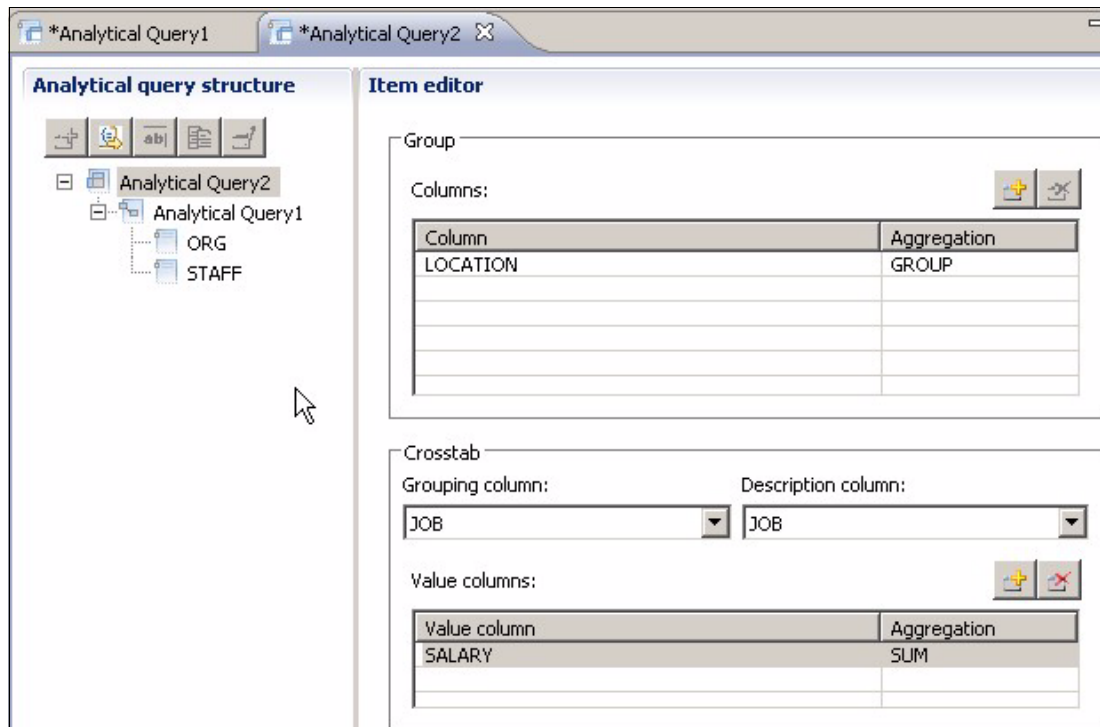


Figure 9-25 Defining the join conditions in an analytical query

The query structure in the navigation tree on the left reflects all component parts of this query. You can see that the data starts with two independent queries (ORG and STAFF). Data from these two queries is joined by Analytical Query1, before being cross-tabulated by Analytical Query2. Users can edit any of these stages in the query by merely clicking the respective node. For example, to edit the join, we can click in Analytical Query1 in the tree, as shown in Figure 9-26 on page 210.

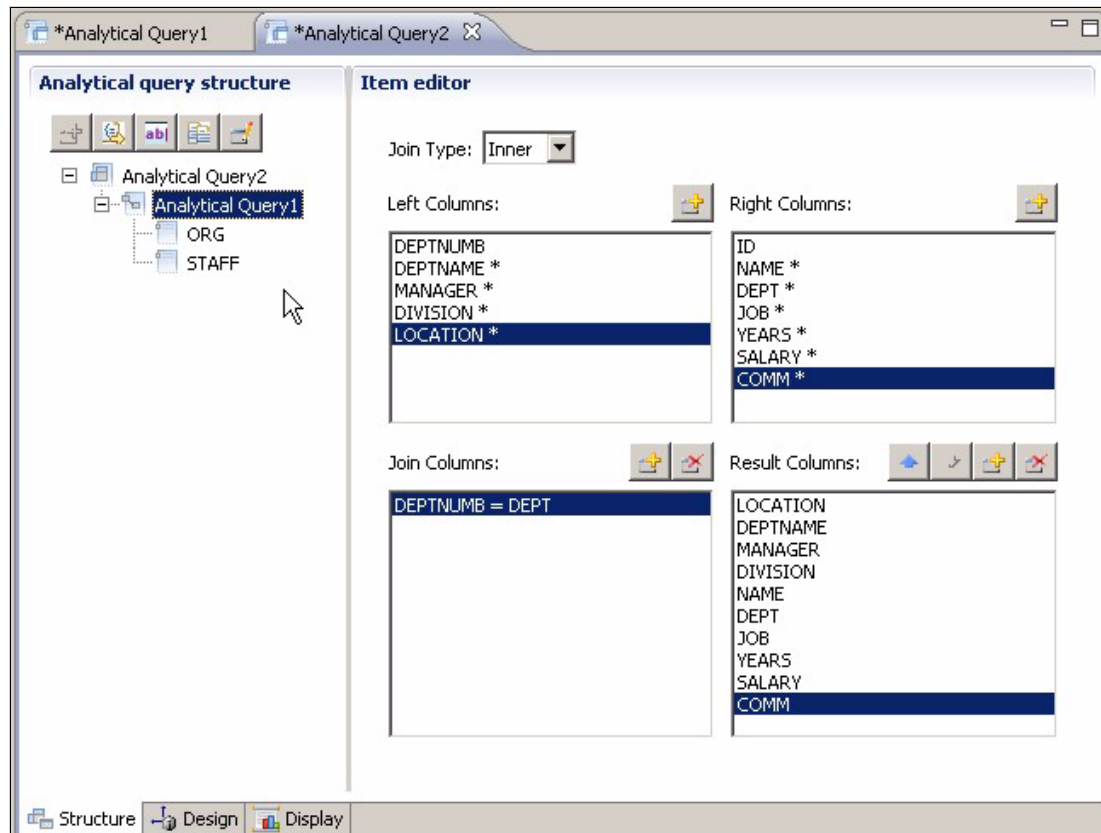


Figure 9-26 Editing an analytical query embedded within another query

As before, this analytical query contains all component parts — the original Analytical Query1 (which performed the join) can be discarded because the second query contains its own copy of it.

When we run the query, the joined result set is cross-tabulated, as shown in Figure 9-27.

	1	2	3	4
	LOCATION	SALARY(CLERK)	SALARY(MGR)	SALARY(SALES)
1	ATLANTA	24964.50	17506.75	34814.30
2	BOSTON	24766.70	20659.80	16502.83
3	CHICAGO	22014.50	18352.80	18001.75
4	DALLAS	27829.80	21150.00	37111.00
5	DENVER	13030.50	19818.00	33298.50
6	NEW YORK	<NULL>	83463.45	<NULL>
7	SAN FRANCISCO	10988.00	18555.50	56532.70
8	WASHINGTON	27757.35	18357.50	18171.25

Figure 9-27 The result of a cross-tabulation of a joined pair of result sets

As you can see, the result of these two steps is, again, nothing more than query results. These results can be used to define query charts, a QMF form, a visual report, or content within a QMF dashboard. The final query can also be used as a component part of another

analytical query. There is no limit to the number of times independent queries can be amalgamated together into a single query object. For example, Spiffy Insurance can use the following approach when gathering information from across the enterprise for a QMF report:

- ▶ Sales by customer ID is retrieved from DB2 on z/OS.
- ▶ Customer ID, name, and sales rep ID is retrieved from DB2 on LUW.
- ▶ The foregoing two result sets are joined to give sales by customer name and sales rep ID.
- ▶ The resulting data is further joined with a query that retrieves sales rep name and ID from a SQL Server database, resulting in sales by customer name and sales rep name.
- ▶ The foregoing information is conditionally cross-tabulated to produce a sum of sales by sales rep name, across customers, breaking out the sums across the following categories: 0 to \$10K, \$10K to \$100K, and \$100K and greater.
- ▶ The same information is gathered directly from IMS, covering the 'real-time' data gathered in the past 24 hours.
- ▶ The real-time and historical data tabulations are appended to one another, setting a column which indicates the nature of each row (transactional or historical).
- ▶ The resulting query is used as the basis for a QMF query.

Though this example might be a little more contrived than some real-world scenarios, it illustrates the power of QMF analytical queries and the ability to indefinitely cascade them together.

9.4 What can I do with my query results?

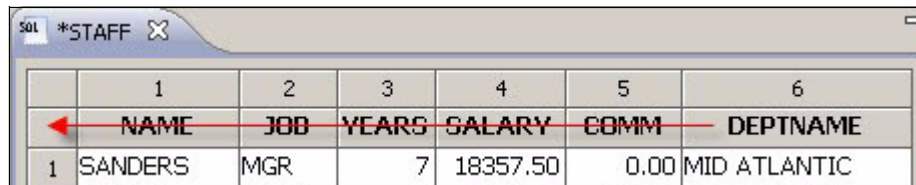
After you have created a query using one of the approaches outlined in the prior topics, you can use the resulting data in many ways, from invocation of built-in charts and drill-downs to further data manipulation operations. This topic uses a query developed earlier in this chapter to take you through the broad set of options that can be applied to a query result set from a relational, hierarchical, or analytical query.

9.4.1 Manipulating data in the query results grid

The Results tab allows the user to apply groupings, calculated columns, aggregations, and conditional formatting to the query data.

Grouping data

We'll start by applying groupings. To do so, click the DEPTNAME column header and drag it all the way to the left edge, as shown in Figure 9-28, until the insertion marker turns blue.



	1	2	3	4	5	6
	NAME	JOB	YEARS	SALARY	COMM	DEPTNAME
1	SANDERS	MGR	7	18357.50	0.00	MID ATLANTIC

Figure 9-28 Creating a side group by dragging DEPTNAME to the left edge of the results

A side group is created, as shown in Figure 9-29. Repeat this process for the JOB column, creating a secondary side group, as shown in Figure 9-29. Double-click the top left cell to auto-size the grid cells to fit their content.

SQL

*STAFF

Prop

Outlin

	1	2	3	4	5	6
	DEPTNAME	JOB	NAME	YEARS	SALARY	COMM
1	GREAT LAKES	CLERK	SCOUTTEN	5	11508.60	84.20
2			YAMAGUCHI	6	10505.90	75.60
3						
4		MGR	PLOTZ	7	18352.80	0.00
5						
6		SALES	KOONITZ	6	18001.75	1386.70
7						
8	All values for GREAT LAKES					
9	HEAD OFFICE	MGR	JONES	12	21234.00	0.00
10			DANIELS	5	19260.25	0.00
11			LU	10	20010.00	0.00
12			MOLINARE	7	22959.20	0.00
13		All values for HEAD OFFICE				
14	MID ATLANTIC	CLERK	SNEIDER	8	14252.75	126.50
15			JAMES	5	13504.60	128.20
16						
17		MGR	SANDERS	7	18357.50	0.00
18						
19	SALES	PERNAI	8	18171.25	612.45	

Result set

NAME

JOB

YEARS

SALARY

COMM

DEPTNAME

Layout

Top Groups

Side Groups

DEPTNAME

JOB

Data Columns

NAME

YEARS

SALARY

COMM

Columns can also be grouped by dragging them into the top and side groups in the Outline view

SQL

Prompted

Diagram

Layout

Results

Preview

Figure 9-29 Defining query side groups by the outline view

Adding calculated columns

You can also add calculated columns to the query. These columns are added to the result set after it is retrieved from the database (as opposed to adding the calculated column in the SQL). Right-click the COMM column header and select **Add Calculated Column Before** from the resulting context menu. The Calculated Column window is displayed. Enter TOTAL for the column name and SALARY+COMM for the expression, as shown in Figure 9-30.

Calculated Column

Name:

Expression:

Type:

OK Cancel

Figure 9-30 Adding a calculated column

Because the calculated column is computed by QMF for Workstation, rather than the database, the query does not need to be rerun; the new column values are immediately displayed when the dialog is closed. Click the TOTAL column label and drag it to the right edge to place it immediately after the COMM column. Again, double-click the upper left cell to auto-size the grid (or select **Results** → **Autofit** → **All** from the menu).

Applying conditional formatting

Now let's apply conditional formatting to the query results to call out specific exceptions. In this case, we'll highlight rows that have a total that exceeds \$20,000. This conditional formatting specification can be defined by right-clicking the TOTAL column label and selecting Font from the resulting context menu. The Layout Properties window is displayed, as shown in Figure 9-30 on page 212.

Click the **Add** icon in the toolbar (highlighted in Figure 9-30 on page 212) to add a new format option. Enter the following text into the **Condition expression** field:

TOTAL > 20000

Tip: The conditional expression can refer to any column or combination of columns, not merely the one to which the formatting is being applied.

For example, the TOTAL column could be formatted based on the percentage relationship between the SALARY and COMM fields, as shown here:

COMM / (SALARY+COMM) > 20%

When the expression has been defined, you can set the format to be used when the expression is met. For this example, click the background color and select a pale red from the resulting color picker, as shown in Figure 9-31.



Figure 9-31 Defining conditional formatting for a query column

Click **OK** to close the window. The results automatically display matching cell values, as shown in Figure 9-32 on page 214.

	1	2	3	4	5	6	7	
	DEPTNAME	JOB	NAME	YEARS	SALARY	COMM	TOTAL	
1	GREAT LAKES	CLERK	SCOUTTEN	5	11508.60	84.20	11592.80	
2			YAMAGUCHI	6	10505.90	75.60	10581.50	
3				2	6	22014.50	84.20	22174.30
4		MGR	PLOTZ	7	18352.80	0.00	18352.80	
5				1	7	18352.80	0.00	18352.80
6		SALES	KOONITZ	6	18001.75	1386.70	19388.45	
7				1	6	18001.75	1386.70	19388.45
8		All values for GREAT LAKES			4	6	58369.05	1386.70
9	HEAD OFFICE	MGR	JONES	12	21234.00	0.00	21234.00	
10			DANIELS	5	19260.25	0.00	19260.25	
11			LU	10	20010.00	0.00	20010.00	
12			MOLINARE	7	22959.20	0.00	22959.20	
13				4	9	83463.45	0.00	83463.45
14		All values for HEAD OFFICE			4	9	83463.45	0.00
15	MID ATLANTIC	CLERK	SNEIDER	8	14252.75	126.50	14379.25	
16			JAMES	5	13504.60	128.20	13632.80	
17				2	7	27757.35	128.20	28012.05
18		MGR	SANDERS	7	18357.50	0.00	18357.50	
19				1	7	18357.50	0.00	18357.50
20		SALES	PERNAL	8	18171.25	612.45	18783.70	

Figure 9-32 Query results with conditional formatting

There's no limit to the number of conditional formatting specifications that can be applied to a given result set. For example, you might want to highlight a given column using red, yellow, or green, depending on the row values. In addition, formatting can be applied to the entire row if a given column value triggers a formatting condition.

Aggregating data

As a final step, we will apply aggregations to the columns that will be used at the summary row levels for both the JOB and DEPTNAME groups. Aggregations can be applied by right-clicking the column label and selecting Grouping and Aggregation from the resulting context menu, as shown in Figure 9-33 on page 215.

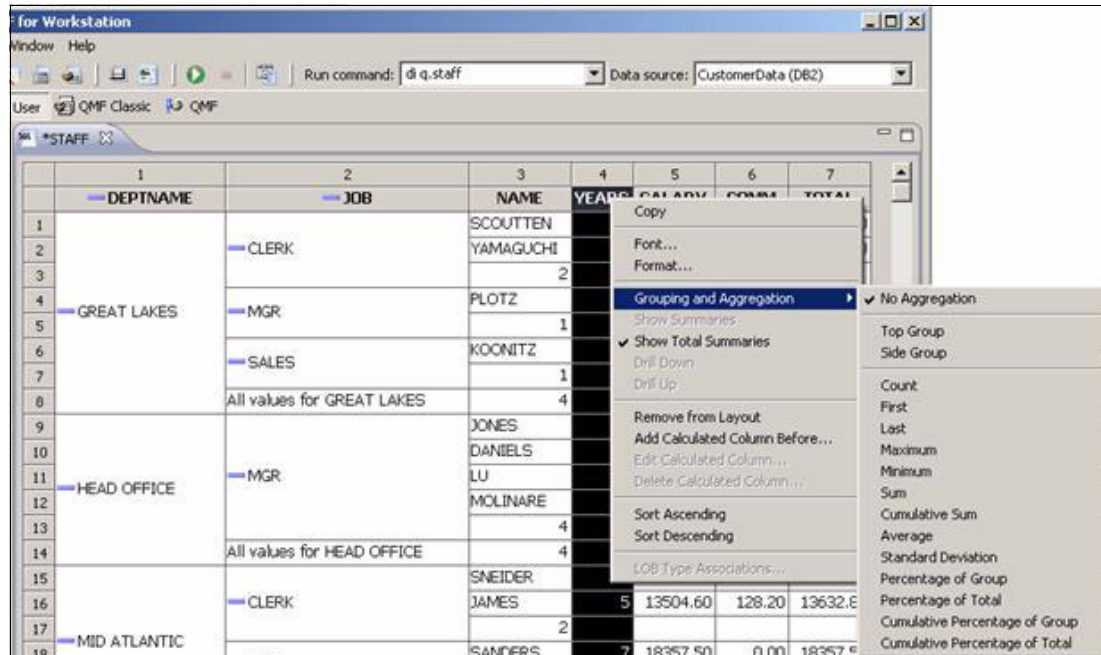


Figure 9-33 Adding column aggregations

Using this approach, try adding the following aggregations to the query columns if you are following along as a live exercise:

- ▶ NAME: count
- ▶ YEARS: average
- ▶ SALARY: sum
- ▶ COMM: max
- ▶ TOTAL: sum

You see that the totals are computed at the job-type level within each department, and also at the department level within the organization.

All of the settings applied to the query results are saved with the query and are reapplied whenever the query is reopened and executed both in QMF for Workstation and QMF for WebSphere. When formatted, the layout can be used to automatically define the structure of a report. The results can also be exported to data files and automatically transferred to a spreadsheet, as we will look at in the next topic.

9.4.2 Transferring query results and formatting to Microsoft Excel

QMF for Workstation can directly transfer query results and formatting to Microsoft Excel. Data can be exported to Excel by pressing Ctrl + B or selecting **Results** → **Display Excel Sheet** from the menu. If your data is arranged with side groups, as in our example, QMF for Workstation creates a pivot chart and a pivot table in Excel, along with the raw data and grouped/aggregated/formatted data (four worksheets total for our example data). If the data is not grouped, only the flat result set is exported into a worksheet. Figure 9-34 on page 216 shows the results of pressing Ctrl + B in our example.

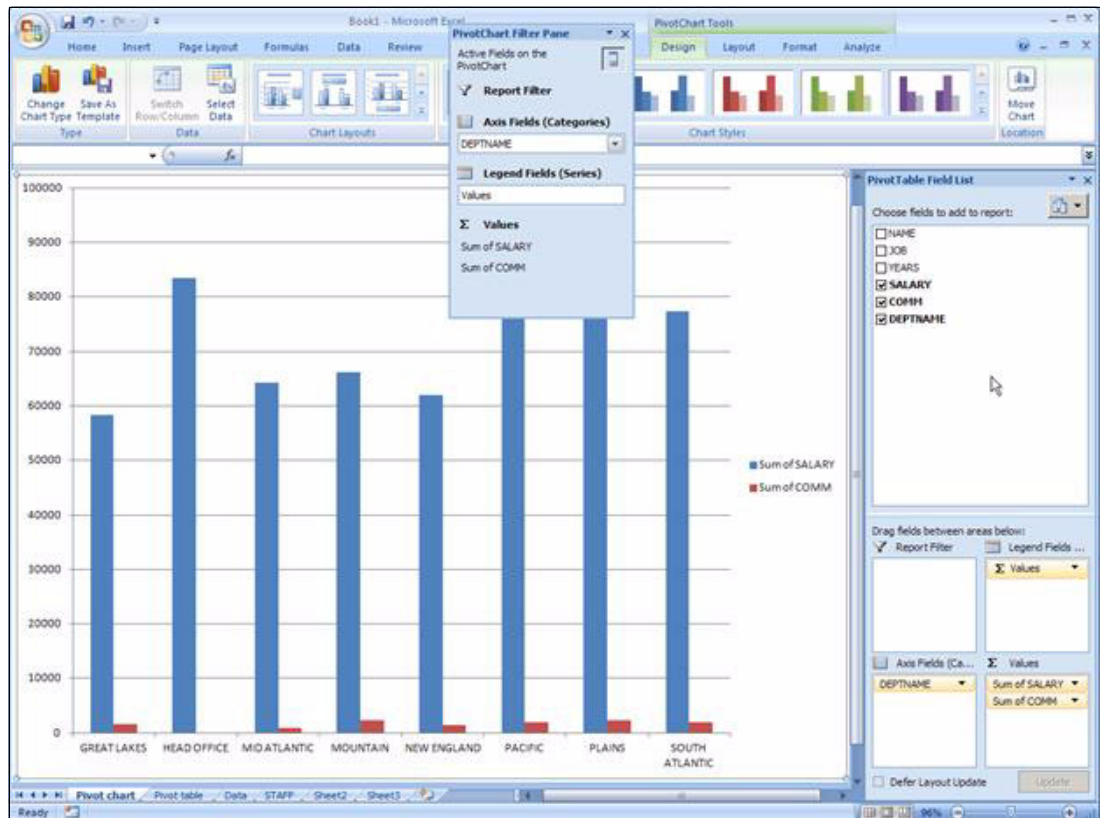


Figure 9-34 Spreadsheet workbook automatically generated by QMF for Workstation

Using Excel's automation interface, the export is performed live, application to application, without the need to create and open an external file.

9.4.3 Adding charts and graphs to visualize query data

QMF for Workstation and QMF for WebSphere provide users with the ability to directly chart the results returned from a query. Any number of charts can be created from a single query result set and each of the defined charts is stored with the query, allowing them to be accessed when the query is reopened. For example, we might want to use the staff query in the preceding topic to include the following charts:

- ▶ Total salary by department
- ▶ Average years by job
- ▶ Maximum and average commissions by department

Charts are supported by a new visual query object type. QMF visual queries not only include multiple charting options but also support an expanded set of functions that can be used to compute calculated columns. Because this object type is limited to QMF for Workstation and QMF for WebSphere, QMF visual queries can be stored in the QMF repository but are not eligible to be saved to the QMF catalog.

A QMF query can be converted into a visual query using the following procedure:

- a. Open the QMF query (if not already open).
- b. Select Query->Convert Query to Visual Query from the main menu.

Charts can be added to a visual query by selecting **Results** → **Display Chart** or by clicking the **Display Chart** icon in the toolbar, highlighted in Figure 9-35 on page 217. When either

action is performed, the Chart wizard is displayed. Users can save some time by preselecting the columns to be charted before invoking the wizard.

In Figure 9-35, the DEPTNAME and SALARY columns have been preselected before clicking the **Display Chart** icon. The wizard then appears with the preselected dimension and values for the value columns automatically selected.

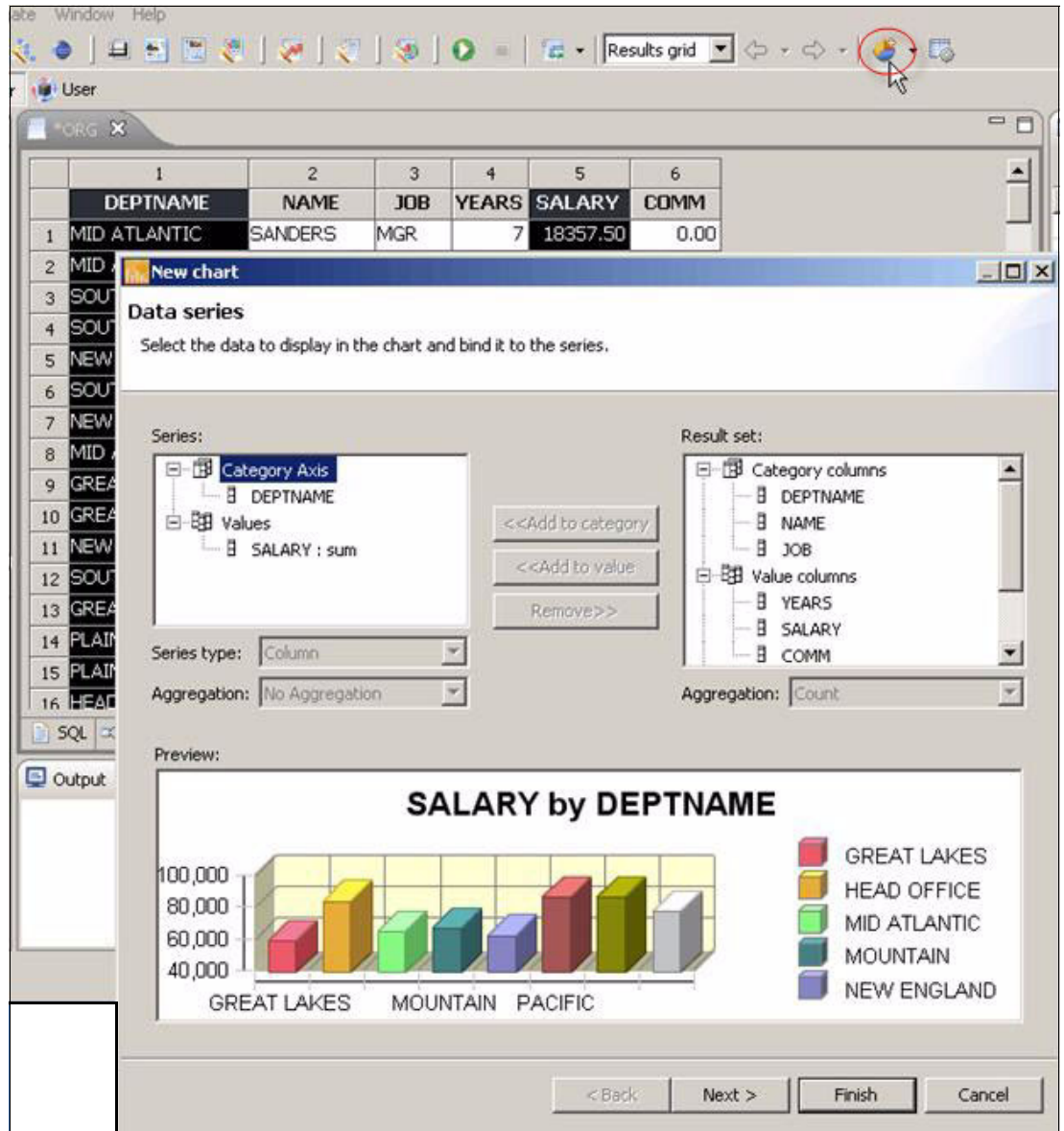


Figure 9-35 Generating a graph from QMF query result data

The Chart wizard allows additional values to be charted against the selected dimension (for example, charting a sum of the salary and maximum commission against the department name). Users can also change the series type on a per-value basis (for example, columns for the sum of salary and an area or line trace for the maximum commission).

When the wizard is closed, the chart is rendered in the query view. Users can return to the results grid by selecting it in the **Results Navigator** view, shown in the lower right pane in Figure 9-36 on page 218.

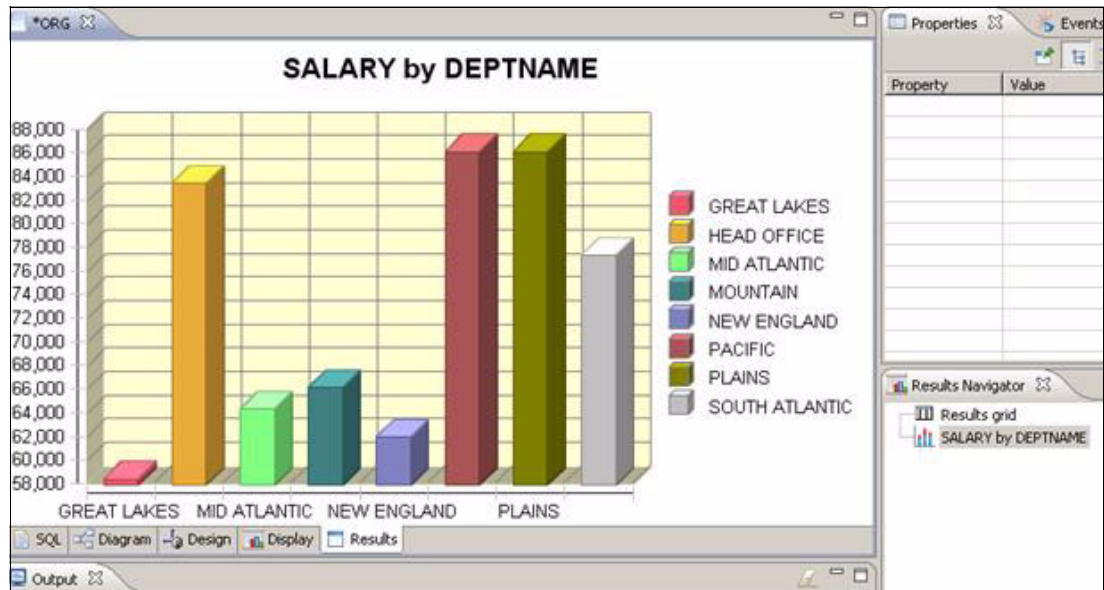


Figure 9-36 Returning to the results grid from a chart

A second chart can be added by selecting new columns in the results grid and invoking the Chart wizard using the toolbar icon or menu item, as before. Using this approach, we've created the three charts that we set out to produce from this query, as shown in the **Results Navigator** view in Figure 9-37. All three charts, as well as the results grid, are now part of the query and are available whenever the query is opened or run.

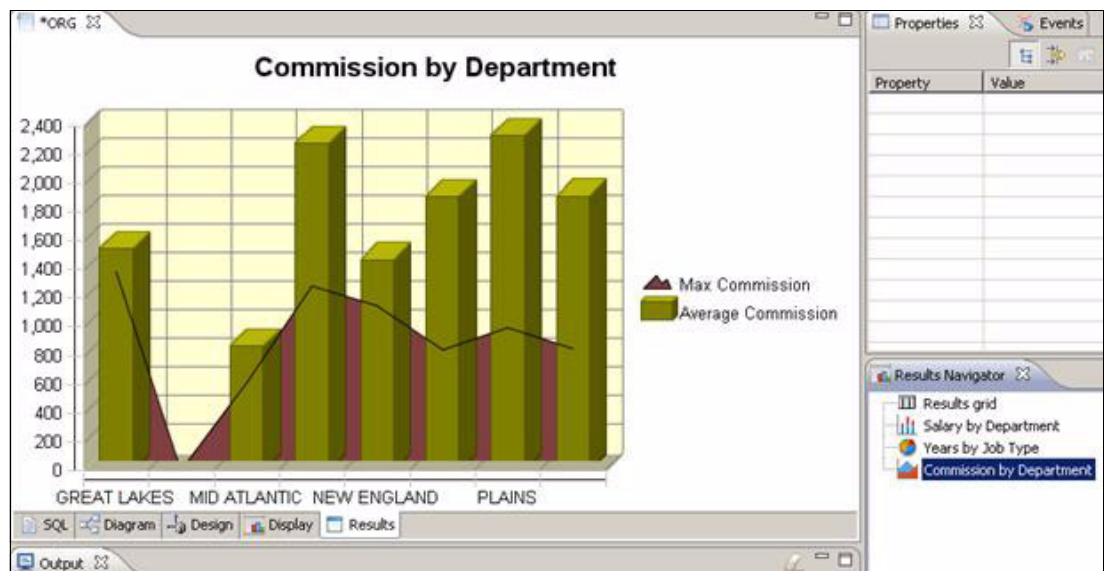


Figure 9-37 Reviewing three charts associated with a single query result set

As you've seen in this topic, charts can be used to provide a summary (aggregate) of one or more columns against a selected dimension (department name, in the case of Figure 9-37). When charting aggregate information, QMF rolls up individual row values in the result set before displaying the information in the chart. In the next topic, we'll cover how QMF drill-down paths allow users to progressively disaggregate data by clicking through a query chart.

9.4.4 Developing reports and dashboards from query data

In addition to directly charting query results, you can use your results to drive the development of visually rich reports and interactive dashboards. These QMF functions are so robust that we've devoted entire chapters to each of them:

See Chapter 11, "Creating reports" on page 233 for more information about how to develop reports

See Chapter 14, "Putting it all together: Developing dashboards" on page 289 for more information about developing interactive dashboards

9.4.5 Defining drill-down paths through query data

Drill-down paths allow a user to view query results in summary form and progressively drill into a specific summary to view the component parts behind it. By way of example, we will use the STAFF and ORG tables to initially display a sum of salary on a division basis. It returns four bars, each of which sums the salaries of people in the respective divisions, as shown in Figure 9-38.

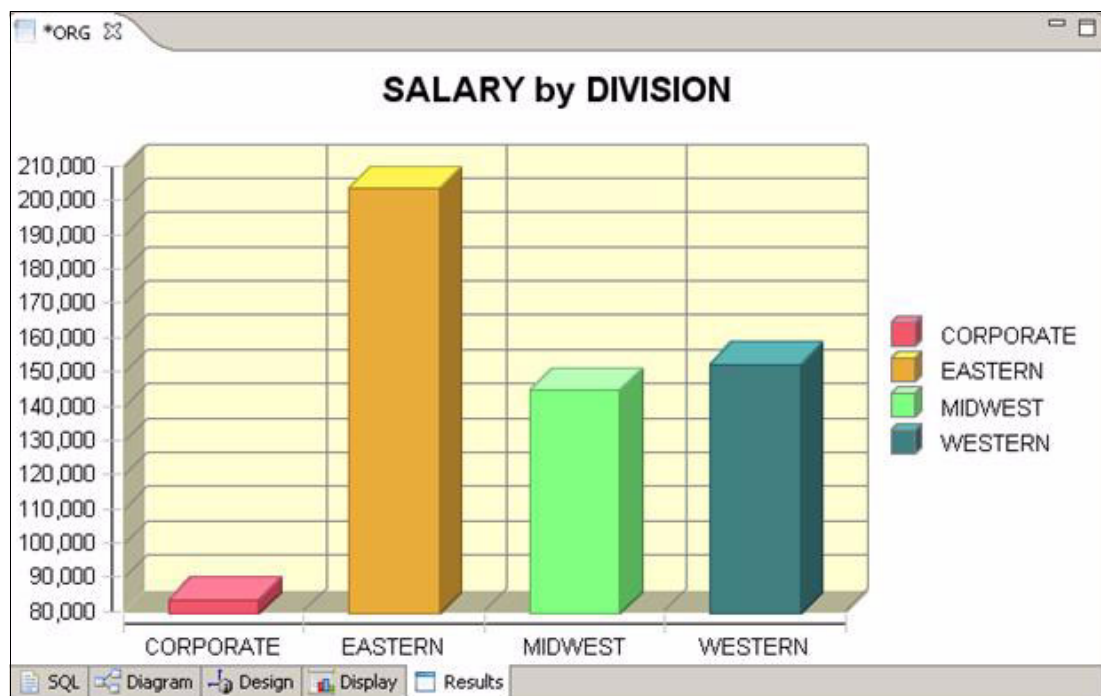


Figure 9-38 Rolling up salary data to view sums at the division level

The information presented in the summary chart is useful, but users might also want to view additional details by drilling into a specific area to see the underlying data. In this case, divisions contain multiple departments, departments contain multiple jobs, and jobs contain multiple people. A user might want to view the sum of salary across each of these aggregation levels. It can be achieved by defining a QMF drill-down path.

A drill-down path can be created by clicking the **Transfer To** toolbar icon and selecting **Drill-Down Path** from the resulting menu (Figure 9-39 on page 220) or by selecting **Query → Transfer To → Drill-down Path** from the menu.

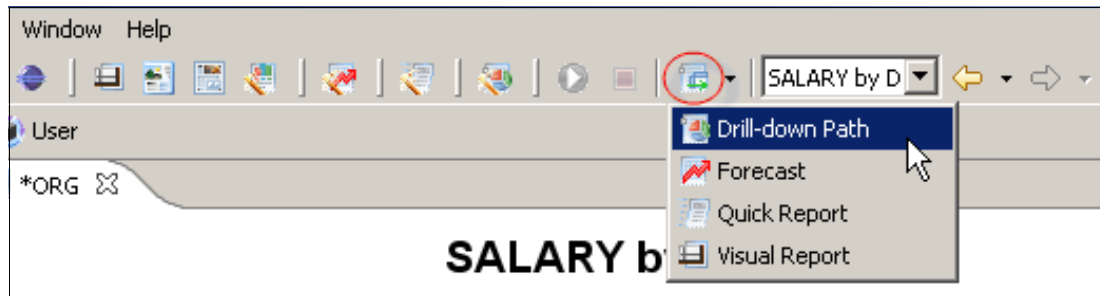


Figure 9-39 Defining a drill-down path

When selected, a new drill-down path is created and initialized with the summary chart from the active query and QMF then starts 'record mode'. Users then define the drill path by clicking an item in the chart (the bars in our example). As soon as an item is clicked, the Chart wizard is displayed so that the user can define what should be presented when this click action is performed.

The Chart wizard automatically carries forward the value column (sum of salary), so users need only define the new aggregation level that the operation should be performed over. In our case, we select DEPTNAME (Figure 9-40) because it is the second level of rollup in our particular query.

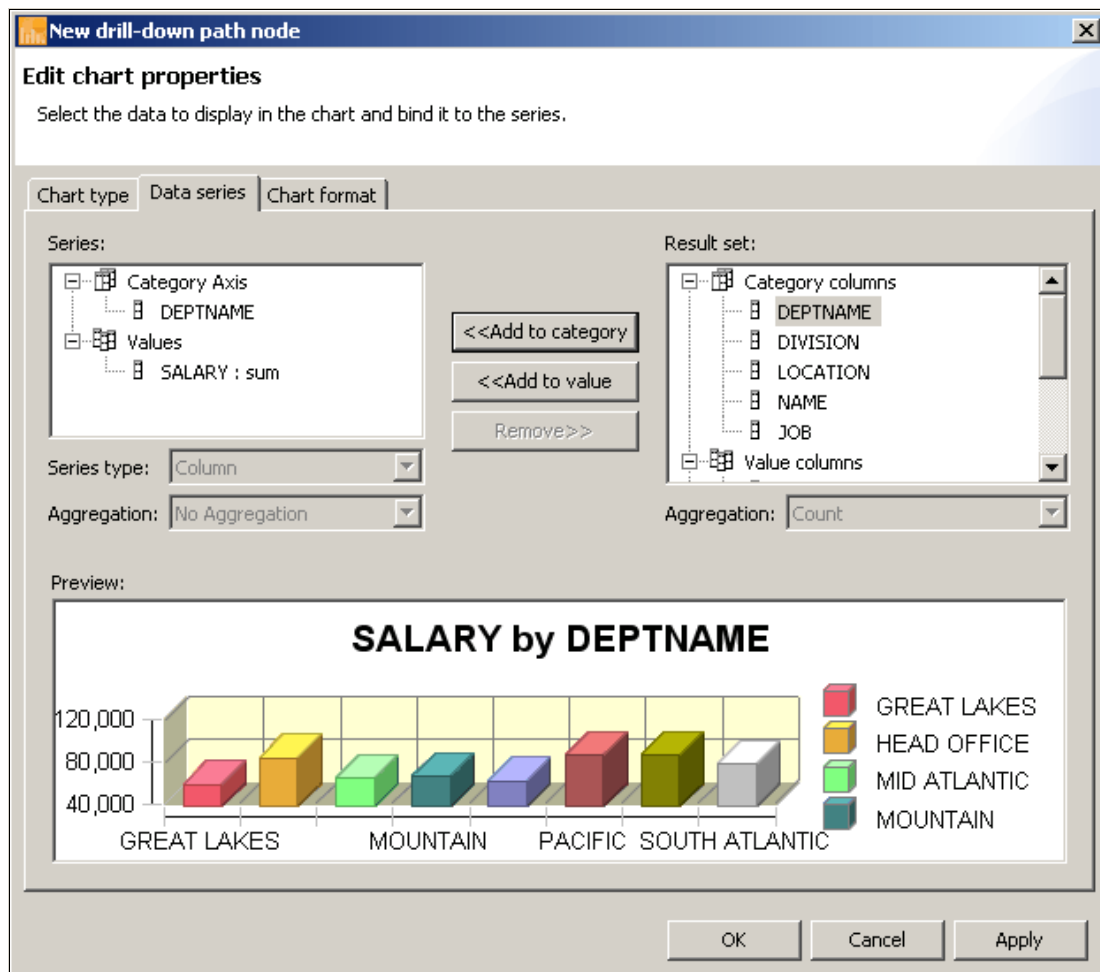


Figure 9-40 Defining the first level of drill-down — sum of salary by department name

Each drill-down level allows the user to change the chart type and the category axis. For example, you could display the sum of salary by division in a column chart. When clicking a division bar, the user is presented with an average of years against job type, presented in a 3-dimensional pie chart.

Now that we've defined the second level of drill-down, it replaces the original chart, as shown in Figure 9-41. The chart title reflects the fact that the chart is showing salary by department for the eastern division only (because it is the bar that we clicked on when performing the drill-down).

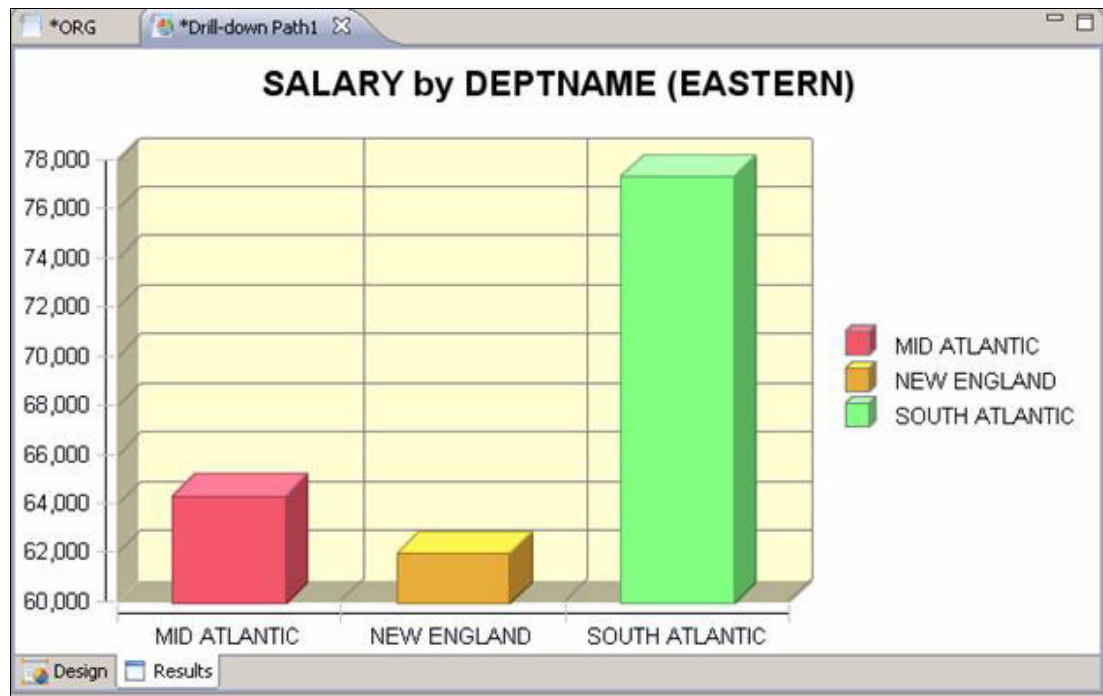


Figure 9-41 Viewing the second level of detail in a drill-down path

The next level of detail can be defined by repeating the process of clicking a chart element and defining the data to be displayed. Using this approach, we complete the drill-down path by defining the sum of salary by job, followed by salary by individual. After the last chart has been defined, the drill-down path recording mode can be stopped by clicking the **Stop recording** icon in the toolbar, shown at the far right of Figure 9-42.

When the recording has been stopped, users can return to the top-level aggregation (salary by division) by clicking the **SALARY by DIVISION** listing in the **Results Navigator** view. Users can then click through the chart and use the forward and back navigation arrows highlighted in Figure 9-42 to move up and down the data. This drill-down path can now be saved and shared with other workstation or web users.

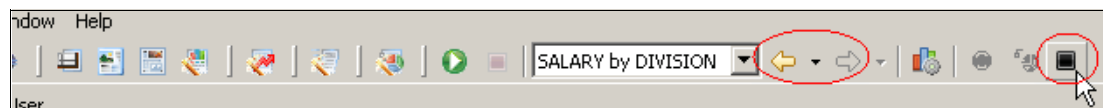


Figure 9-42 Stopping the recording (definition) of a drill-down path

Changes to a drill-down path can be made by clicking the **Design** tab at the base of the view. The **Design** tab allows users to review the drill path and make changes to each defined level (perhaps changing the chart type, properties, or data series that are presented).

9.5 Queries against multidimensional data

In addition to accessing relational and hierarchical data sources, QMF can access data from multidimensional (OLAP) data sources. These queries use MDX, rather than SQL, and multidimensional data is typically deployed to end users by way of the results grid or OLAP-aware charts within QMF dashboards.

In this example, we will add an OLAP query to a visual project. OLAP queries can also be created independently of a visual project (much like a relational query). To do so, simply select the **File** → **New** → **OLAP Query** from the main menu.

To add an OLAP query to a visual project, proceed as follows:

1. Right-click **Queries** in the Project Explorer tree and select **Insert Query** from the context menu, as shown in Figure 9-43.

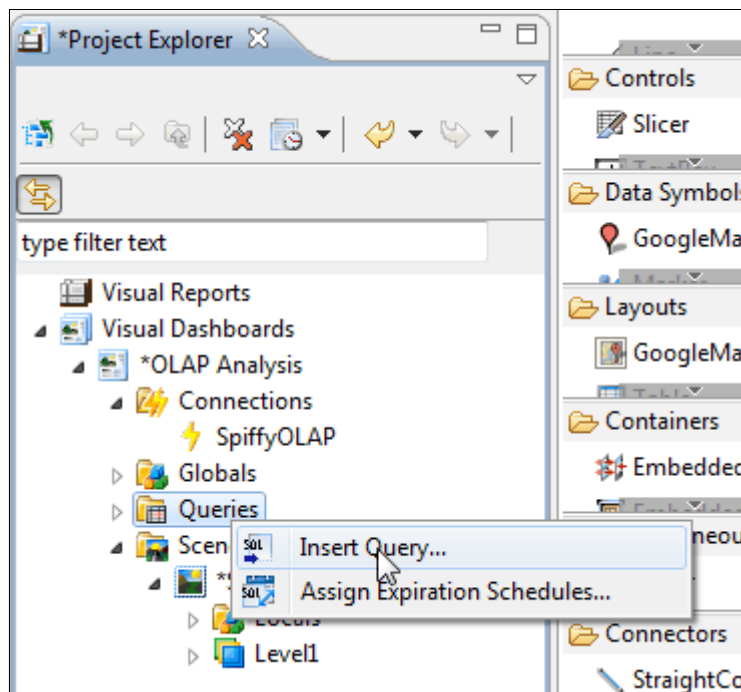


Figure 9-43 Adding an OLAP query to a visual project

2. Change the query name to **qryGrid** and click **Finish**. QMF now displays the OLAP query editor (with its Cube Structure tree), as shown in Figure 9-44.
3. Select the cube that you want to use for the query by right-clicking **Cube** in the tree and selecting **Set Cube** from the context menu. In this example, we select Spiffy, as shown in Figure 9-44, and click **Finish**.

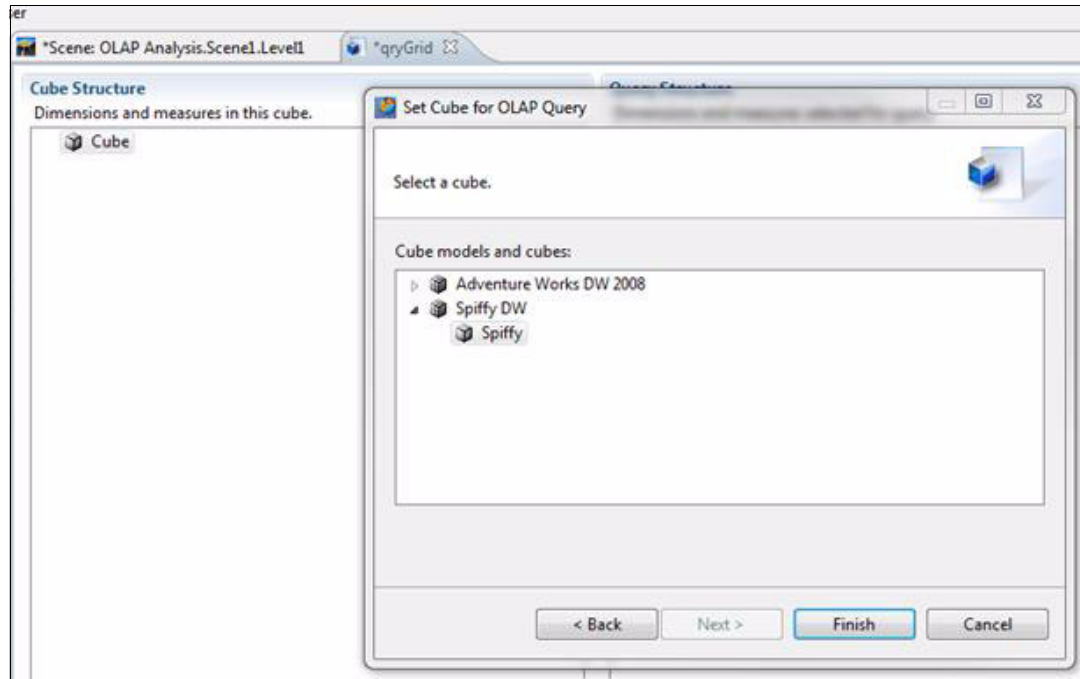


Figure 9-44 Selecting the cube to use for the OLAP query

4. We now see a list of dimensions and measures on the left and query structure on the right, where we define which dimensions and measures to use (Figure 9-45).

For this example, set them as follows:

- a. Drag Time to be the top dimension
- b. Drag Market to be the side dimension
- c. Drag Sales to be the measure

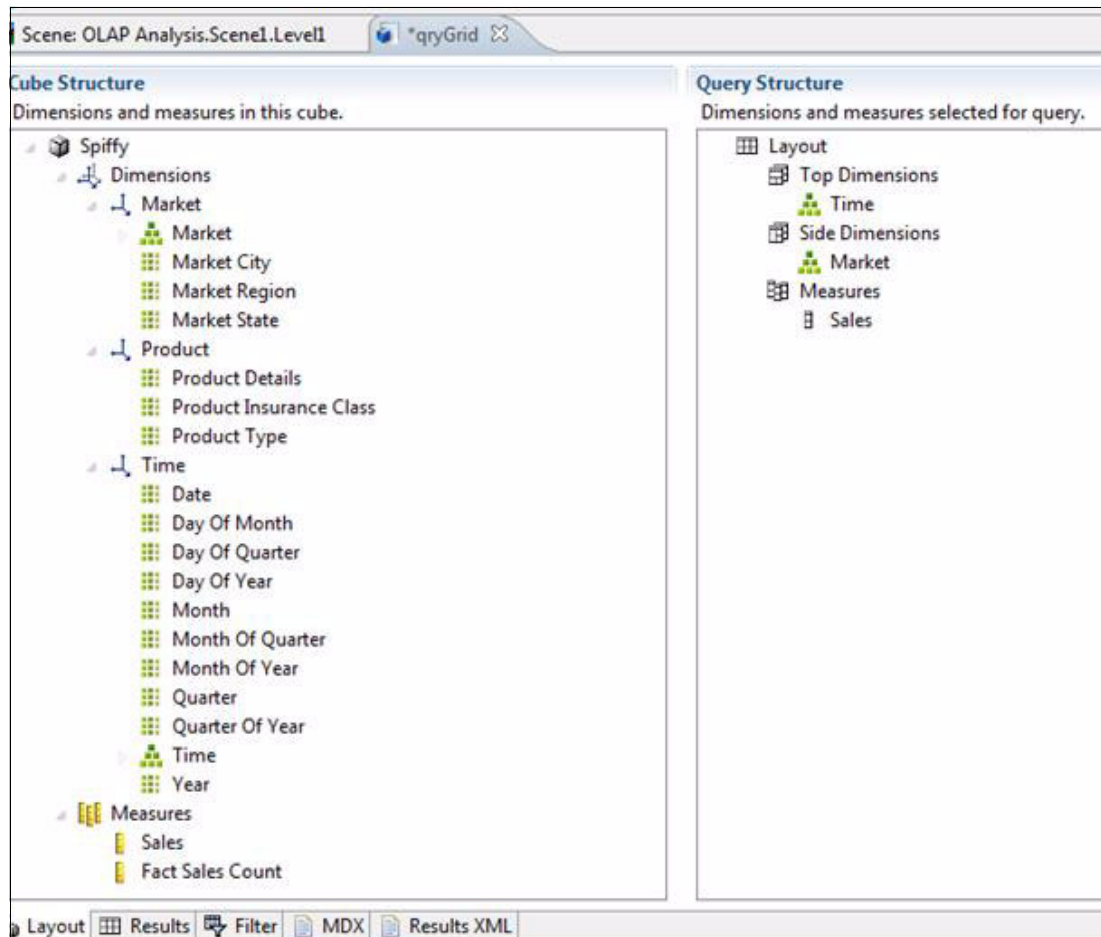


Figure 9-45 Creating the OLAP query structure

5. Click the **Results** tab to see the data (Figure 9-46).

The screenshot shows the IBM DB2 Query Management Facility (QMF) interface with the 'Results' tab selected. The table displays sales data for three markets (Central, East, West) across three years (2009, 2010, 2011). The table has four columns: 1 (Market), 2 (Sales), 3 (Sales), and 4 (Sales). The rows are labeled 1, 2, and 3, corresponding to Central, East, and West markets respectively.

	1	2	3	4
	Time	+ Calendar 2009	+ Calendar 2010	+ Calendar 2011
	Market	Sales	Sales	Sales
1	+ Central	26159488	27103385	24598108
2	+ East	82514186	79672850	82208937
3	+ West	43897568	42813736	41829842

The bottom toolbar includes icons for Layout, Results, Filter, MDX, Results XML, and Preview.

Figure 9-46 OLAP query results

This query shows a cross-tab of sales by year and market. Clicking the + signs expands the grid and shows an additional level of data (Figure 9-47).

1	2	3	4	5	6	7
Time	Calendar 2009	Quarter 1, 2009	Quarter 2, 2009	Quarter 3, 2009	Quarter 4, 2009	Calendar 2010
Market	Sales	Sales	Sales	Sales	Sales	Sales
1 - Central	26159488	7419072	5994881	5670053	7075482	27103385
2 + AR	1262012	473265	237961	338586	212200	1215495
3 + CO	2450654	758151	444503	457304	790696	3053515
4 + IA	8957830	2497563	2195278	1829423	2435566	9228614
5 + KS	1999124	376307	420663	493859	708295	2405487
6 + LA	1397663	346930	361251	296246	393236	875857
7 + MN	430991	87882	57704	103215	182190	728519
8 + MO	700148	309473	91676	181242	117757	908637
9 + MT	12844		12844			75500
10 + ND	1111				1111	11699
11 + NE	290650	139627		58471	92552	274427
12 + OK	457163	49396	175034	63678	169055	220526
13 + SD	134561	44148		67265	23148	127853
14 + TX	7836235	2209541	1917495	1771410	1937789	7649368
15 + WY	228502	126789	80472	9354	11887	327888
16 + East	82514186	21006903	20096992	20684247	20726044	79672850
17 + West	43897568	10236956	10237074	11759529	11664009	42813736

Figure 9-47 Expanded OLAP data, where calendar year 2009 and the central markets have been expanded

OLAP queries are typically developed visually by dragging and dropping dimensions and measures in the Layout tab. However, much like relational queries, power users can use the **MDX** tab to directly enter the query text if that method is preferred.



Updating data with the QMF table editor

QMF for Workstation has a built-in table editor that provides a convenient means of editing data across all database types to which a JDBC connection can be configured. Using the table editor, you can easily search for, add, edit, and delete data without writing SQL statements and without additional programming.

This chapter shows you how to access and use the table editor, as well as how to restrict access to table editing functions, given that these functions are not typically in the hands of business or casual users.

10.1 Controlling access to the QMF table editor

Given the power of this feature, QMF provides the ability for administrators to strictly control its access and use. All editing operations remain compliant with your established database security settings. In addition, QMF administrators can enable or disable this feature on a per-user and group basis and subjectively grant access when working with specific data sources. You can also restrict SQL statements, such as INSERT, UPDATE, DELETE, and more.

Access to the table editor capability is granted in the Options tab of the Edit Resource Limits Group Schedule dialog, as shown in Figure 10-1. To access this dialog, right-click the data source name and select Properties from the resulting context menu.

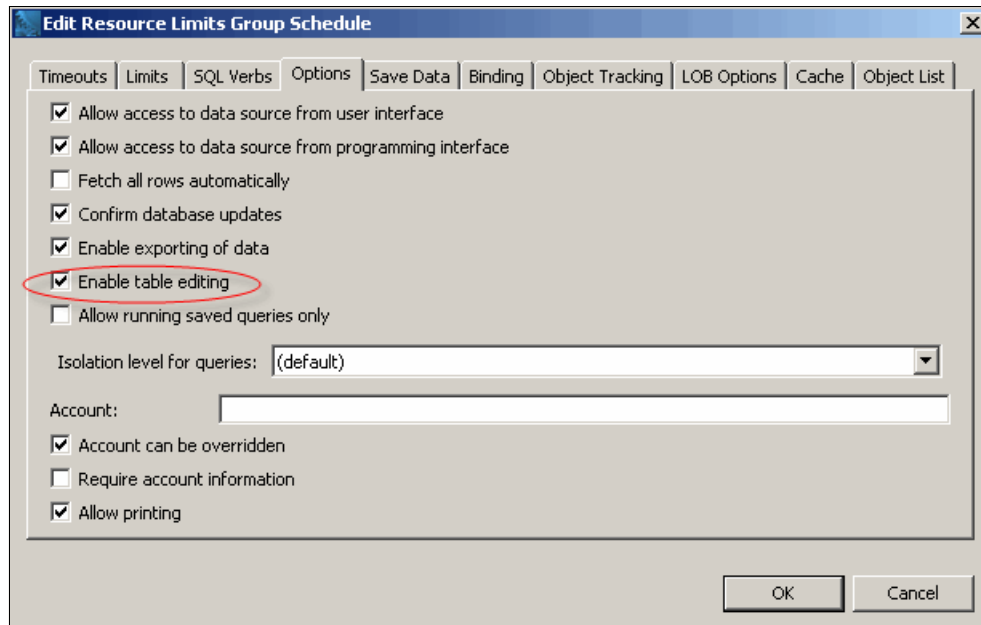


Figure 10-1 Controlling access to QMF for Workstation's table editor

QMF administrators can also use resource limits to define the actions that specific users or groups are permitted to perform on a global or per data source basis, as well as on a time of day basis. For example, a given class of user can be limited to retrieving no more than 1,000 rows of data from any given query when running against a specific production database during business hours. The same user might be permitted to retrieve 5,000 rows from that same database during non-business hours and 10,000 rows if executing a query against a non-production database. The tabs in Figure 10-1 show the scope of what can be controlled with QMF resource limits.

10.2 Using the QMF table editor

Users who have been granted access to the table editor can invoke it by right-clicking a database table and selecting **Open With** → **Table Editor** from the resulting context menu, as shown in Figure 10-2.

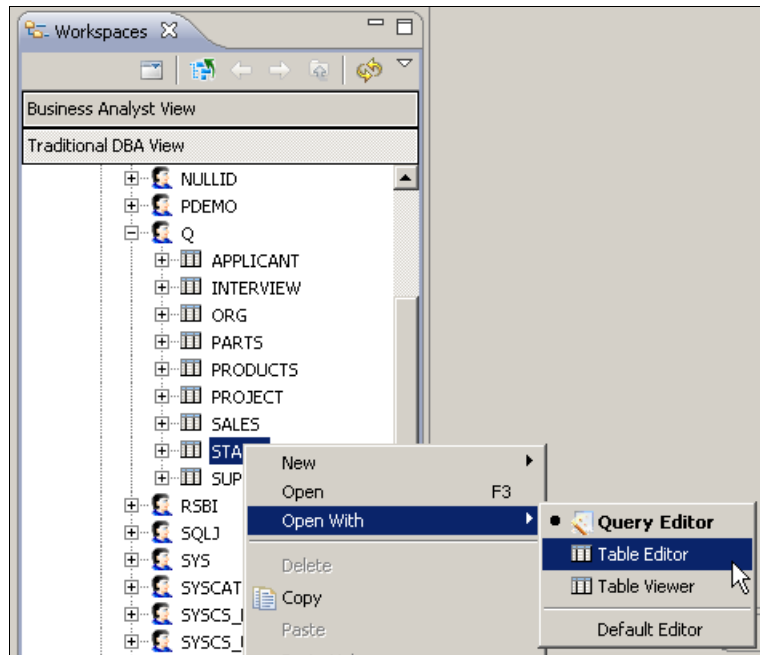


Figure 10-2 Invoking the table editor in QMF for Workstation

The table editor presents the first 100 rows of the table in an editable grid (Figure 10-3).

	1	2	3	4	5	6	7
	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
1	10	SANDERS	20	MGR	7	18357.50	0.00
2	20	PERNAL	20	SALES	8	18171.25	612.45
3	30	MARENGHI	38	MGR	5	17506.75	0.00
4	40	O'BRIEN	38	SALES	6	18006.00	846.55
5	50	HANES	15	MGR	10	20659.80	0.00
6	60	QUIGLEY	38	SALES	5	16808.30	650.25
7	70	ROTHMAN	15	SALES	7	16502.83	1152.00
8	80	JAMES	20	CLERK	5	13504.60	128.20
9	90	KOONITZ	42	SALES	6	18001.75	1386.70
10	100	PLOTZ	42	MGR	7	18352.80	0.00
11	110	NGAN	15	CLERK	5	12508.20	206.60
12	120	NAUGHTON	38	CLERK	5	12954.75	180.00
13	130	YAMAGUCHI	42	CLERK	6	10505.90	75.60
14	140	FRAYE	51	MGR	6	21150.00	0.00
15	150	WILLIAMS	51	SALES	6	19456.50	637.65
16	160	MOLINARE	10	MGR	7	22959.20	0.00

Figure 10-3 Invoking the table editor on the STAFF sample table

Typically, users will be interested in editing rows that meet specific criteria. For example, a particular user might want to review and make changes to rows associated with employees in a given department. Because manually paging through row after row of data is generally very time-consuming, the table editor includes a Prompted tab that allows users to constrain the editor to specific row criteria. For those familiar with SQL, it is effectively using a WHERE clause to display a subset of the data.

Using this function, we have narrowed the table editor to employees in Department 10 only, as shown in Figure 10-4.

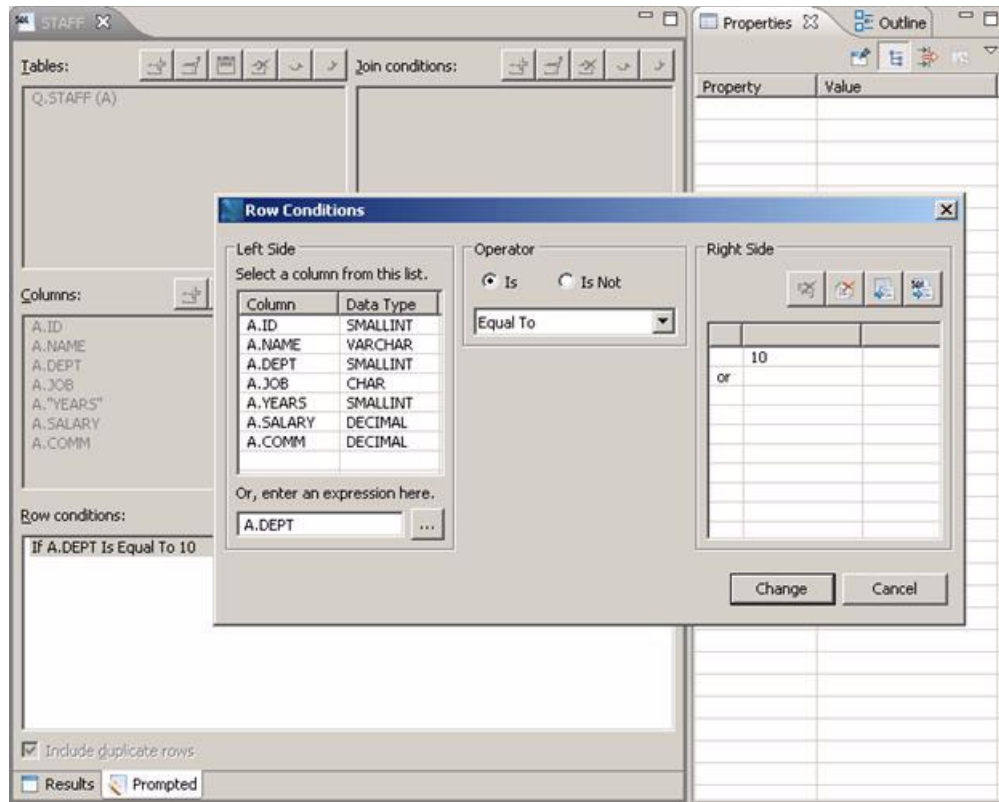


Figure 10-4 Applying a condition to narrow the editor to specific rows of interest

When one or more conditions have been applied, users can review the matching rows by clicking the Results tab at the bottom of the table editor.

The table editor allows users to perform the following actions:

- ▶ Insert rows into a table
- ▶ Delete rows from a table
- ▶ Edit existing column values within an existing row

The foregoing actions can be applied to the table immediately or they can be rolled up into a transaction that the user subsequently commits or rolls back when finished with the table. By default, the table editor is set to immediately commit changes. You can change this option by right-clicking the top-left cell in the grid and unchecking **Immediate Commit** in the resulting context menu, as shown in Figure 10-5.



Figure 10-5 Setting the table editor's commit mode

Be aware that QMF for Workstation retains the commit mode preference and restores this option each time that the user opens the table editor, whether in the same application session or in future application sessions. If it is a one-time change, you might want to set it back before you exit.

Users can edit row values by simply double-clicking the desired cell and typing in a new value, as shown in Figure 10-6. When clicked, the cell changes appearance to indicate that it is editable.

	1	2	3	4	5	6	7
	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
1	160	MOLINARE	10	MGR	7	22959.20	0.00
2	210	LU	10	MGR	10	20010.00	0.00
3	240	DANIELS	10	MGR	5	19260.25	0.00
4	260	JONES	10	MGR	12	21234.00	0.00

Figure 10-6 Editing a column value in the table editor

New rows can be added or edited by right-clicking the top-left cell and selecting **Insert Row** from the resulting context menu shown in Figure 10-5 on page 230. The same context menu actions can also be accessed from the Results menu. Rows can be deleted by right-clicking the row number, as shown in Figure 10-7, and selecting **Delete Row** from the context menu. Changed or added rows are marked with an asterisk, as shown in row 3.

	1	2	3	4	5	6	7
	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
1	160	MOLINARE	10	MGR	7	22959.20	0.00
2	210	LU	10	MGR	10	20010.00	0.00
3*	240	DANIEL	10	MGR	5	19260.25	0.00
4	260	JONES	10	MGR	12	21234.00	0.00

Figure 10-7 Deleting a table row

After all edits have been made, users not operating under the Immediate Commit mode can then commit or roll back their changes using the Commit or Rollback options on the Results menu or in the top-left cell's context menu, as shown in Figure 10-8.

	1	2	3	4	5	6	7
	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
1	160	MOLINARE	10	MGR	7	22959.20	0.00
2	210	LU	10	MGR	10	20010.00	0.00
3*	240	DANIEL	10	MGR	5	19260.25	0.00
4*	260	JONES	10	MGR	12	21234.00	0.00

Figure 10-8 Committing or rolling back changes made in the table editor



Creating reports

If you are already familiar with QMF Classic Edition, you have probably seen tabular QMF reports produced in QMF for TSO or CICS, such as those shown in Chapter 4, “Maximizing your existing System z investment” on page 39. Such reports are called classic reports and are generated in QMF for TSO or CICS by running a query with a set of formatting specifications known as a QMF form.

QMF for Workstation not only allows you to produce classic reports, it also allows you to create visually rich reports that can include almost any element you can imagine. This chapter walks you through some examples that show how to create both classic and visual reports with some basic features. We cover the following topics:

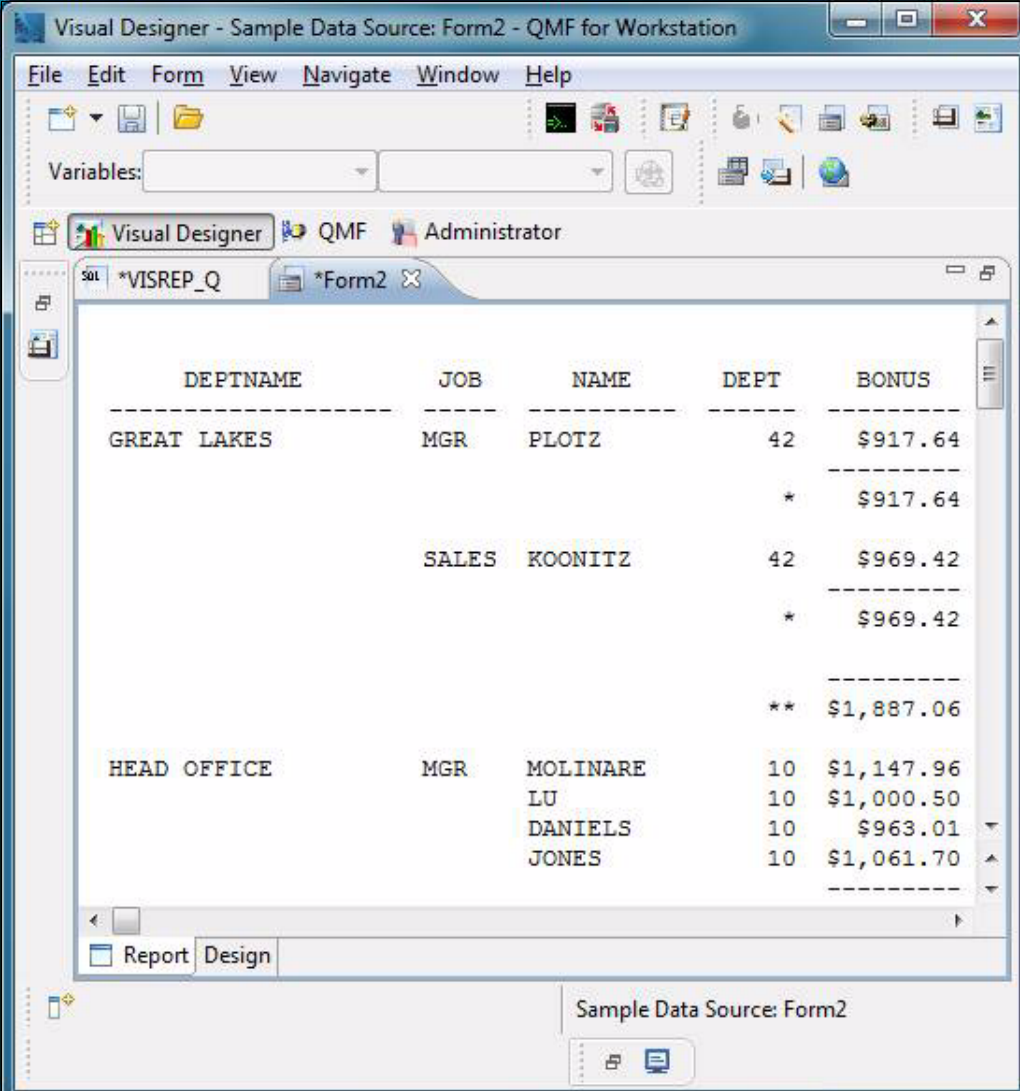
- ▶ Creating classic reports
- ▶ Creating visual reports

For a discussion of additional features not covered here, see *Getting Started with QMF for Workstation and WebSphere* at the following website:

<http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp>

11.1 Creating classic reports

QMF for TSO and CICS users are accustomed to working with the QMF REPORT object, which is generated by applying a QMF form to a query result set. This topic shows you how to use QMF for Workstation to create a classic report such as the one shown in Figure 11-1.



The screenshot shows the Visual Designer interface for a QMF report. The main window displays a report titled "Sample Data Source: Form2" with a menu bar (File, Edit, Form, View, Navigate, Window, Help) and a toolbar. The report is generated from the query "*VISREP_Q" and is displayed in a "Report" view. The report data is as follows:

DEPTNAME	JOB	NAME	DEPT	BONUS
GREAT LAKES	MGR	PLOTZ	42	\$917.64
			*	\$917.64
	SALES	KOONITZ	42	\$969.42
			*	\$969.42
			**	\$1,887.06
HEAD OFFICE	MGR	MOLINARE	10	\$1,147.96
		LU	10	\$1,000.50
		DANIELS	10	\$963.01
		JONES	10	\$1,061.70

Figure 11-1 A QMF for Workstation classic report

To create this report, follow these steps:

1. Create the query shown in Figure 11-2. (Refer back to Chapter 9, “Getting to the data you need: Query methods” on page 191 for help if necessary.)

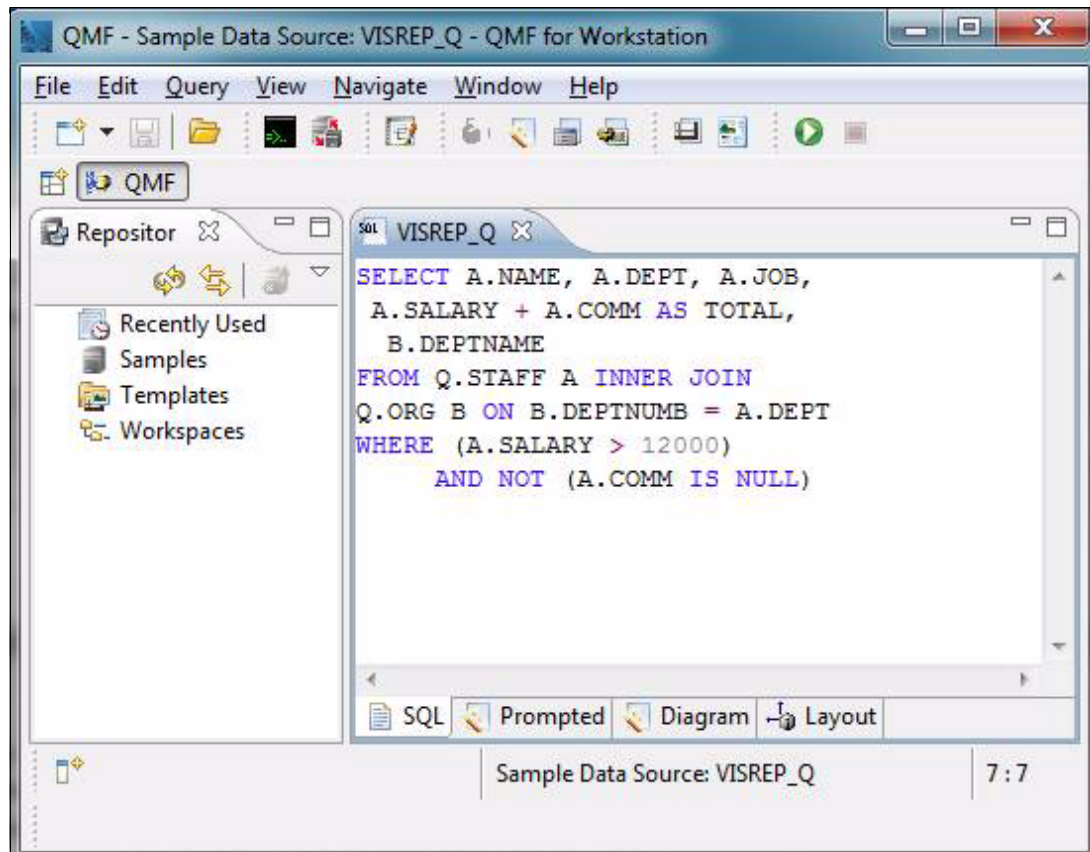


Figure 11-2 Query used to produce the classic report shown in Figure 11-1 on page 234

2. Run the query. Your results are displayed in the query results grid. Manipulate the results as follows:
 - Make DEPTNAME and JOB into side groups by right-clicking the headings and selecting **Grouping and Aggregation** → **Side Group** from the context menu.
 - Add a column by right-clicking the TOTAL heading and selecting **Add Calculated** column from the context menu. Name the new column Bonus and make it an expression of $0.05 * TOTAL$.
 - Add summary totals to the BONUS and TOTAL columns by right-clicking and selecting **Grouping and Aggregation** → **Sum** from the context menu.
 - Add currency formatting to the BONUS and TOTAL columns by right-clicking the headings and selecting Format. **Change As is** to **Currency** and select the **Apply to all levels** check box.

After these modifications, the results grid should look like the one shown in Figure 11-3.

	1	2	3	4	5	6
	DEPTNAME	JOB	NAME	DEPT	BONUS	TOTAL
17	MID ATLANTIC	SALES	PERNAL	20	\$939.19	\$18,783.70
18					\$939.19	\$18,783.70
19		All values ...			\$3257.66	\$65,153.25
20	MOUNTAIN	CLERK	GAFNEY	84	\$660.93	\$13,218.50
21					\$660.93	\$13,218.50
22		MGR	QUILL	84	\$990.90	\$19,818.00
23					\$990.90	\$19,818.00
24		SALES	DAVIS	84	\$813.03	\$16,260.60
25			EDWARDS	84	\$956.45	\$19,129.00
26					\$1769.48	\$35,389.60
27		All values ...			\$3421.31	\$68,426.10
28	NEW ENGLAND	CLERK	NGAN	15	\$635.74	\$12,714.80

Figure 11-3 The query results grid after applying grouping, adding sums and calculations, and applying currency formatting

3. Select **Results** → **Display Report** from the menu. The Display Report wizard opens.
4. Select **Create a new report**, as shown in Figure 11-4, and click **Next**.

If you are connected to a data source where an existing QMF for TSO and CICS catalog resides, you can use any existing QMF for TSO and CICS forms in the catalog for the report by selecting **Use an existing QMF report stored in the QMF catalog**.

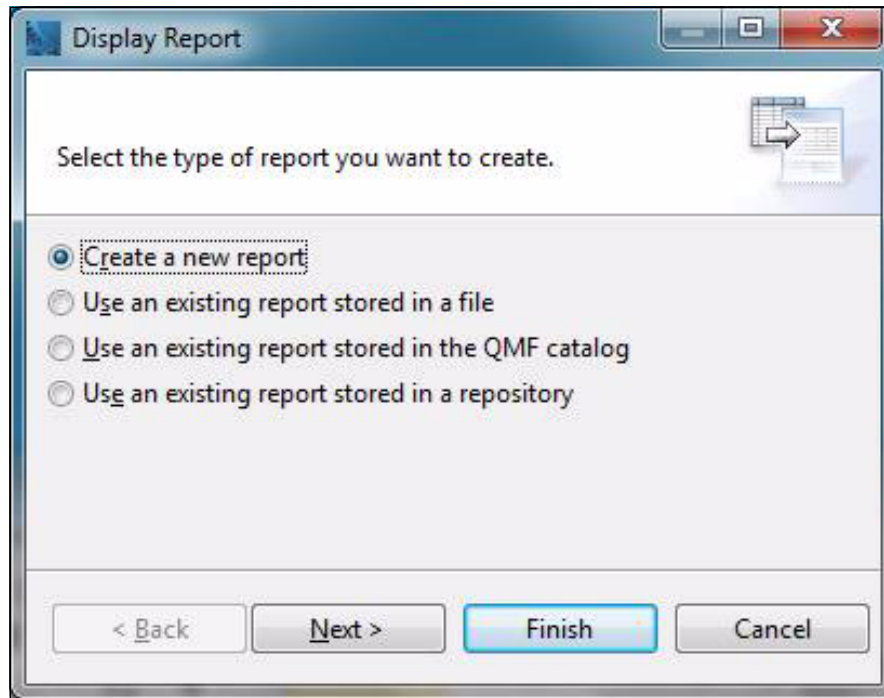


Figure 11-4 Creating a new report from query results

5. As shown in Figure 11-5, select **Create a classic report** from the **Select the type of report** drop-down list. Ensure that Create from query is selected, as shown. Then click **Finish**.

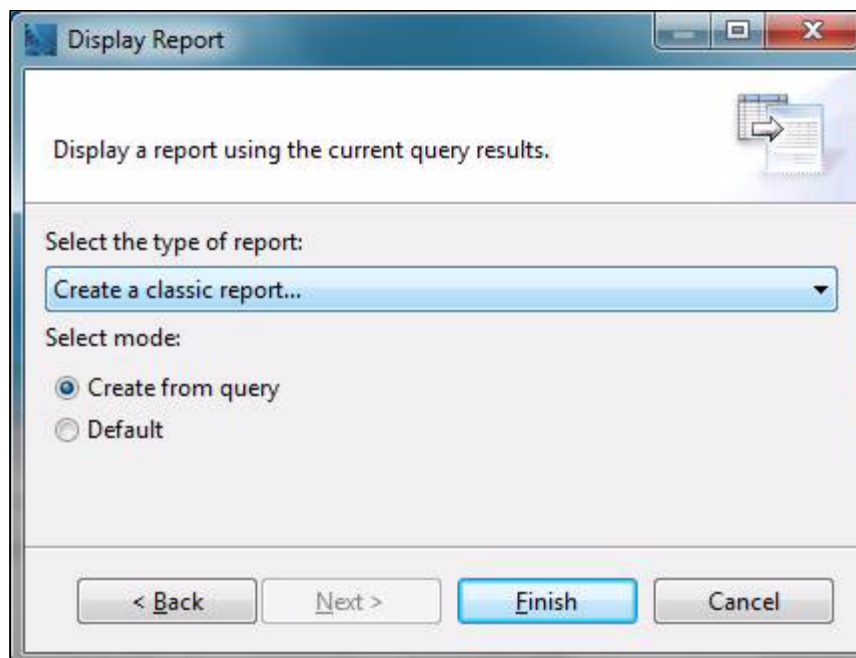


Figure 11-5 Setting the report type to classic

QMF displays the resulting classic report. Notice the two tabs at the lower left in Figure 11-6: **Report** and **Design**. Toggle between them, returning to the **Design** tab.

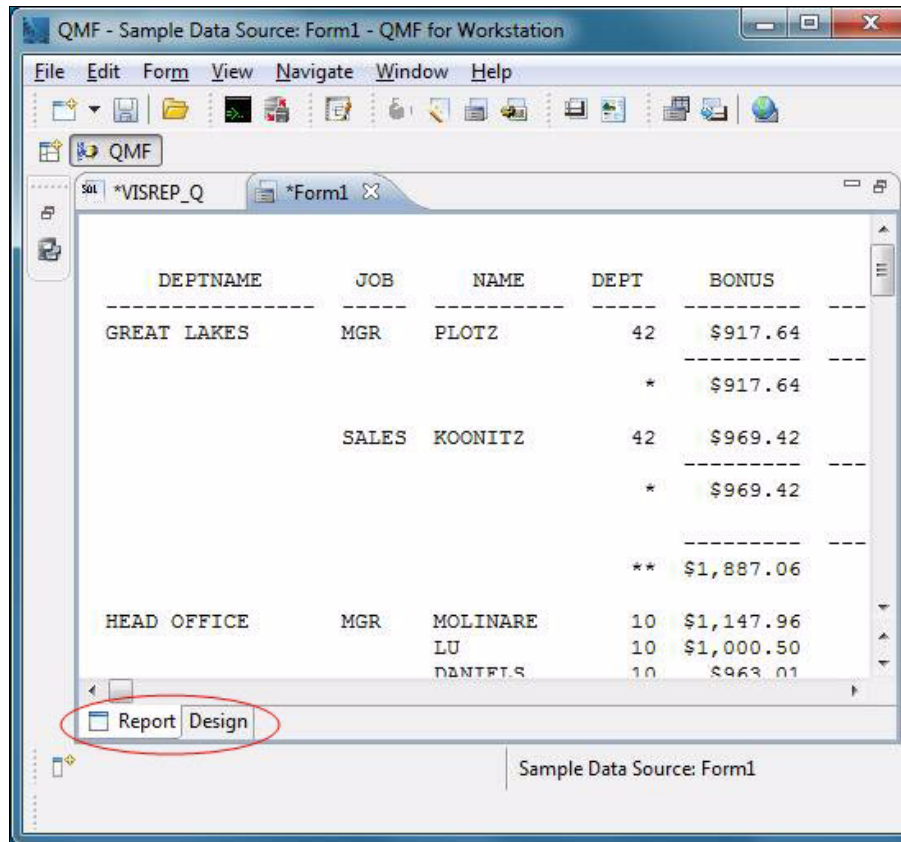


Figure 11-6 The Report tab allows you to immediately see changes made in the Design tab

- Survey various QMF form elements by selecting them from the Form Structure pane, as shown in Figure 11-7.

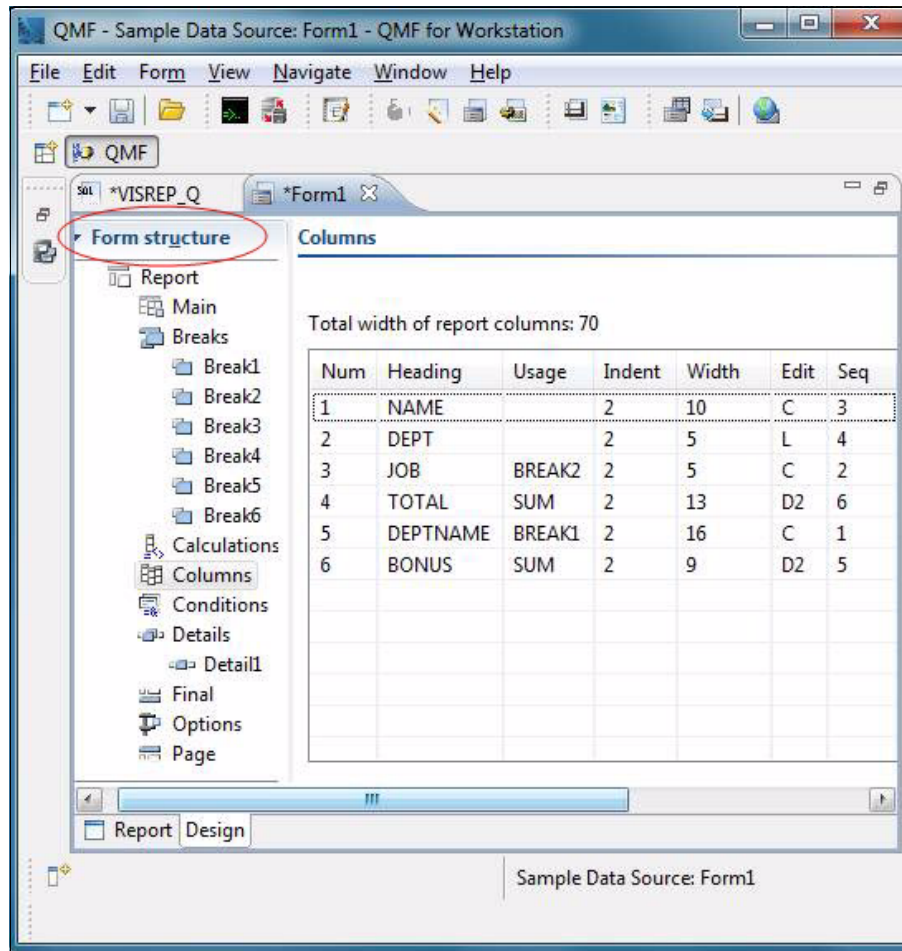


Figure 11-7 Formatting elements for the classic report, shown in the Form Structure pane

The formatting specifications applied earlier to the results in the query results grid were automatically transferred to the form because we selected **Create from query** in Figure 11-5 on page 237.

In the Form Structure pane:

- ▶ All of the classic QMF for TSO and CICS FORM panels are represented in an easy to access fashion.
- ▶ Forms created in QMF for TSO and CICS can be run or edited.
- ▶ New or changed formatting specifications done in this pane will be available in QMF for TSO and CICS if they are saved to the QMF catalog.

You can continue to make changes to the classic report as desired, toggling between changes made in the **Design** tab and viewing your results in the **Report** tab.

Next we look at how to turn this example result set into a QMF visual report.

11.2 Creating visual reports

Visual reports are page-based, printable reports that include both formatted text and graphics to display persistent data to a wide variety of users. In addition to static graphics, visual reports can contain data-driven graphics, such as maps and charts, inserted in different

sections of the report. Each of the data-driven graphics can present data from multiple queries that are run across the enterprise.

It is usually best practice to start out with a storyboard or a design of the finished report before you begin. The basic report we create in this topic is shown in Figure 11-8. The annotations describe the design changes that we intend to make.

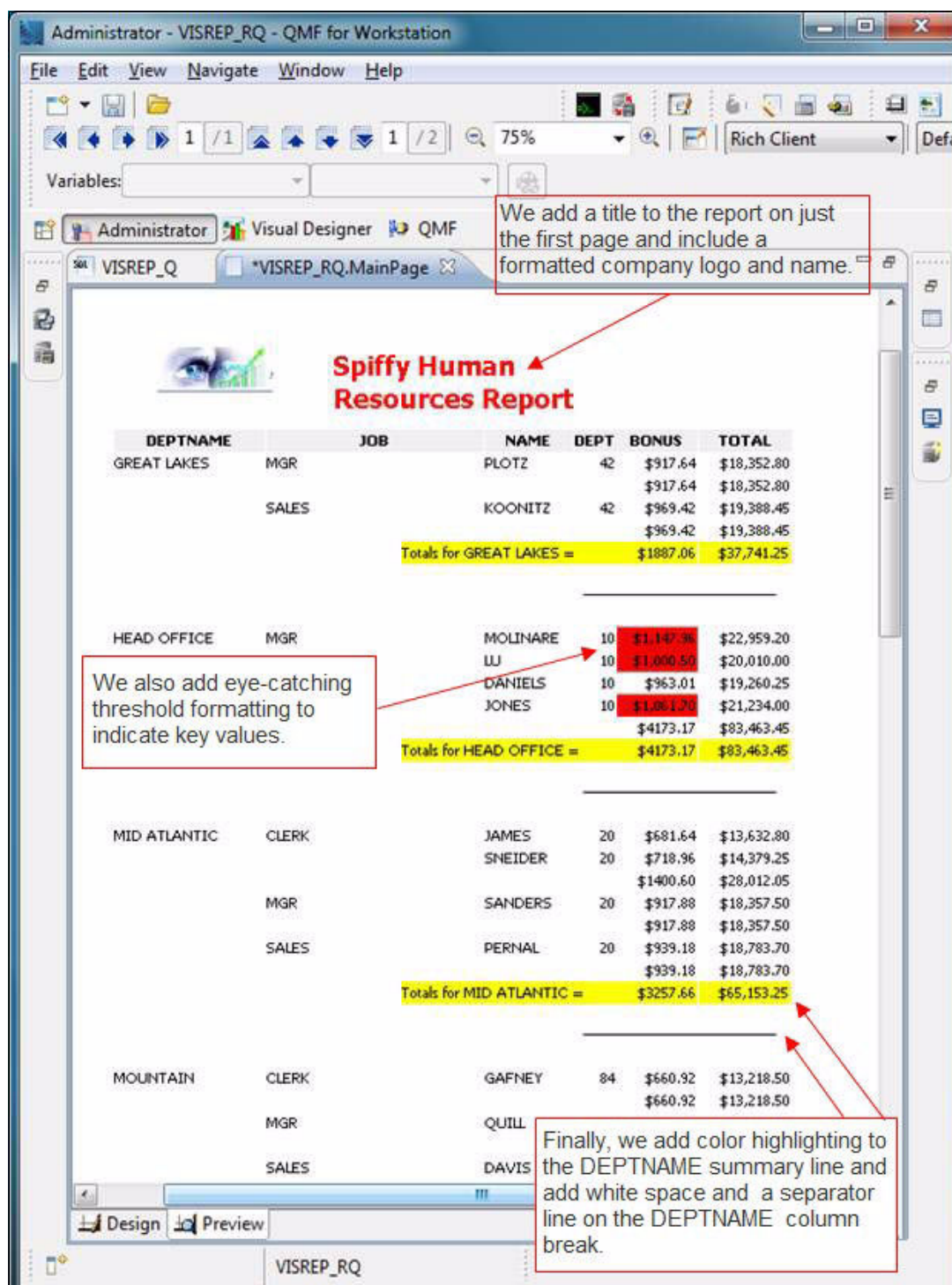


Figure 11-8 Sample visual report

11.2.1 Getting started

We start with the query result set returned from the query we ran to create the classic report shown previously. You can return to Figure 11-2 on page 235 to see this query.

1. Click the tab with the query name to bring the result set to the front again, as shown in Figure 11-9.

	1	2	3	4	5	6
	DEPTNAME	JOB	NAME	DEPT	BONUS	TOTAL
17	MID ATLANTIC	SALES	PERNAL	20	\$939.19	\$18,783.70
18					\$939.19	\$18,783.70
19		All values ...			\$3257.66	\$65,153.25
20	MOUNTAIN	CLERK	GAFNEY	84	\$660.93	\$13,218.50
21					\$660.93	\$13,218.50
22		MGR	QUILL	84	\$990.90	\$19,818.00
23					\$990.90	\$19,818.00
24		SALES	DAVIS	84	\$813.03	\$16,260.60
25			EDWARDS	84	\$956.45	\$19,129.00
26					\$1769.48	\$35,389.60
27		All values ...			\$3421.31	\$68,426.10
28	NEW ENGLAND	CLERK	NGAN	15	\$635.74	\$12,714.80
29			KERMISCH	15	\$618.43	\$12,368.60
30					\$1254.17	\$25,083.40
31		MGR	HANES	15	\$1,032.99	\$20,659.80

Figure 11-9 Returning to the query results by clicking the tab with the name of the query

2. Select **Query** → **Display Report** from the menu. The Display Report wizard opens.
3. Select **Create a new report** and click **Next**.
4. In the **Select the type of report** drop-down list, make sure that **Create a visual report** is selected. Also be sure that **Create from query** is selected, then click **Finish**.

QMF displays the resulting report. Notice the two tabs at the lower left in Figure 11-10: **Design** and **Preview**. Toggle between them, returning to the **Design** tab.

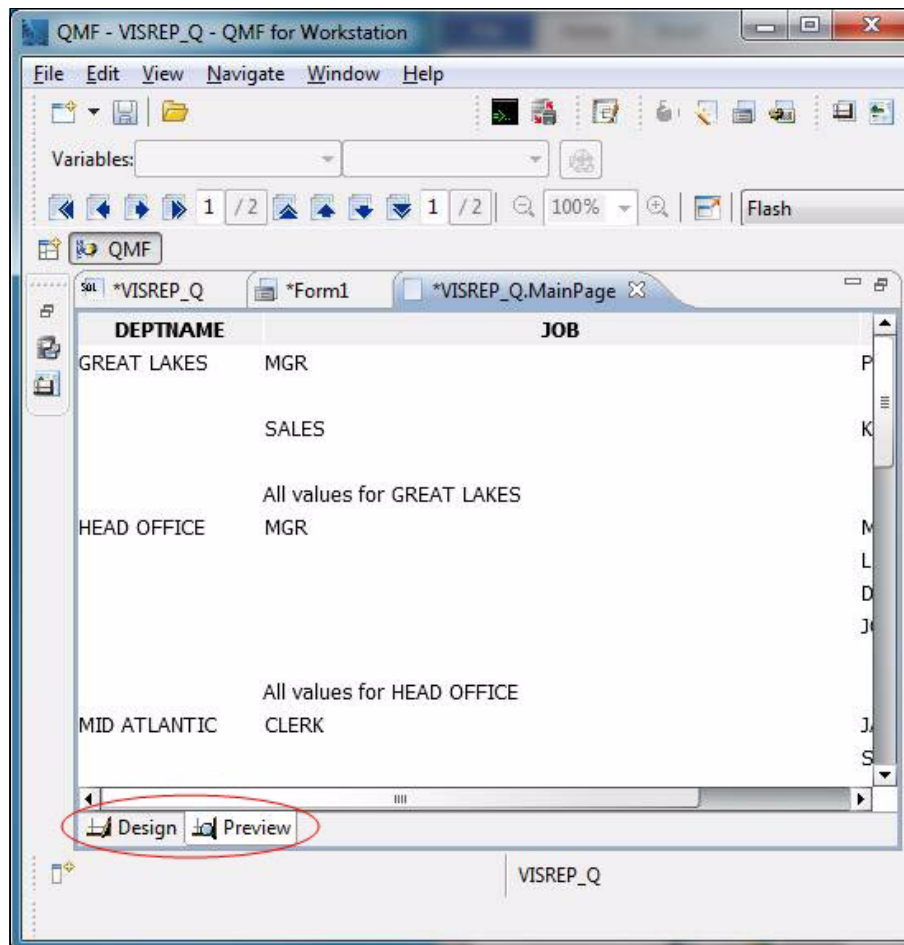


Figure 11-10 Visual report from the same query that was used to produce the earlier classic report

11.2.2 The visual designer

Visual Reports are best crafted in the Visual Designer perspective. To switch to this perspective, select **Window** → **Open Perspective** → **Other** from the menu, then select **Visual Designer**, as shown in Figure 11-11, and click **OK**.

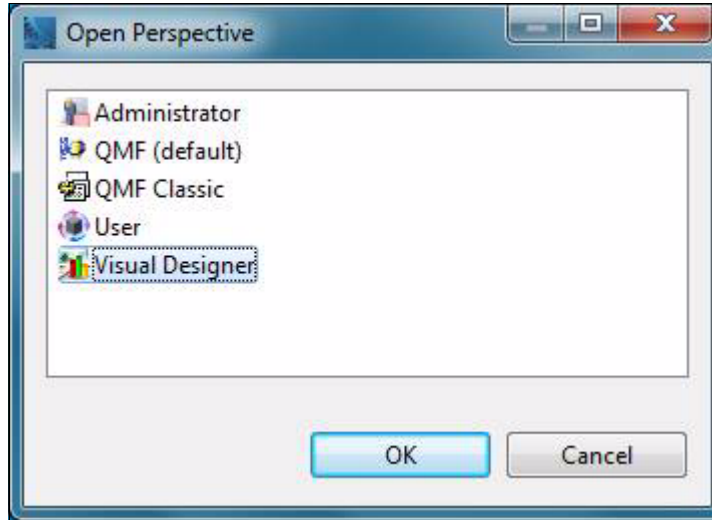


Figure 11-11 Switching to the Visual Designer perspective

The user interface for designing visual reports, shown in Figure 11-12, has several components that are similar to other GUI report designers. If you are not familiar with this type of GUI report designer, take the time to study the next few pages. Notice the following features of the interface:

- ▶ The Project Explorer tree in the left pane allows navigation and display of the sections and groups and their contents.
- ▶ The Properties sheet on the right allows control of various property values for the contents of the sections and groups.
- ▶ The Palette view, at the left of the center pane, includes objects and controls that can be used to populate the sections and groups of the report. These objects and controls can be tied to the results grid data by their properties.
- ▶ The Report canvas in the center pane is where the objects from the palette and the Project Explorer tree are laid out and displayed.

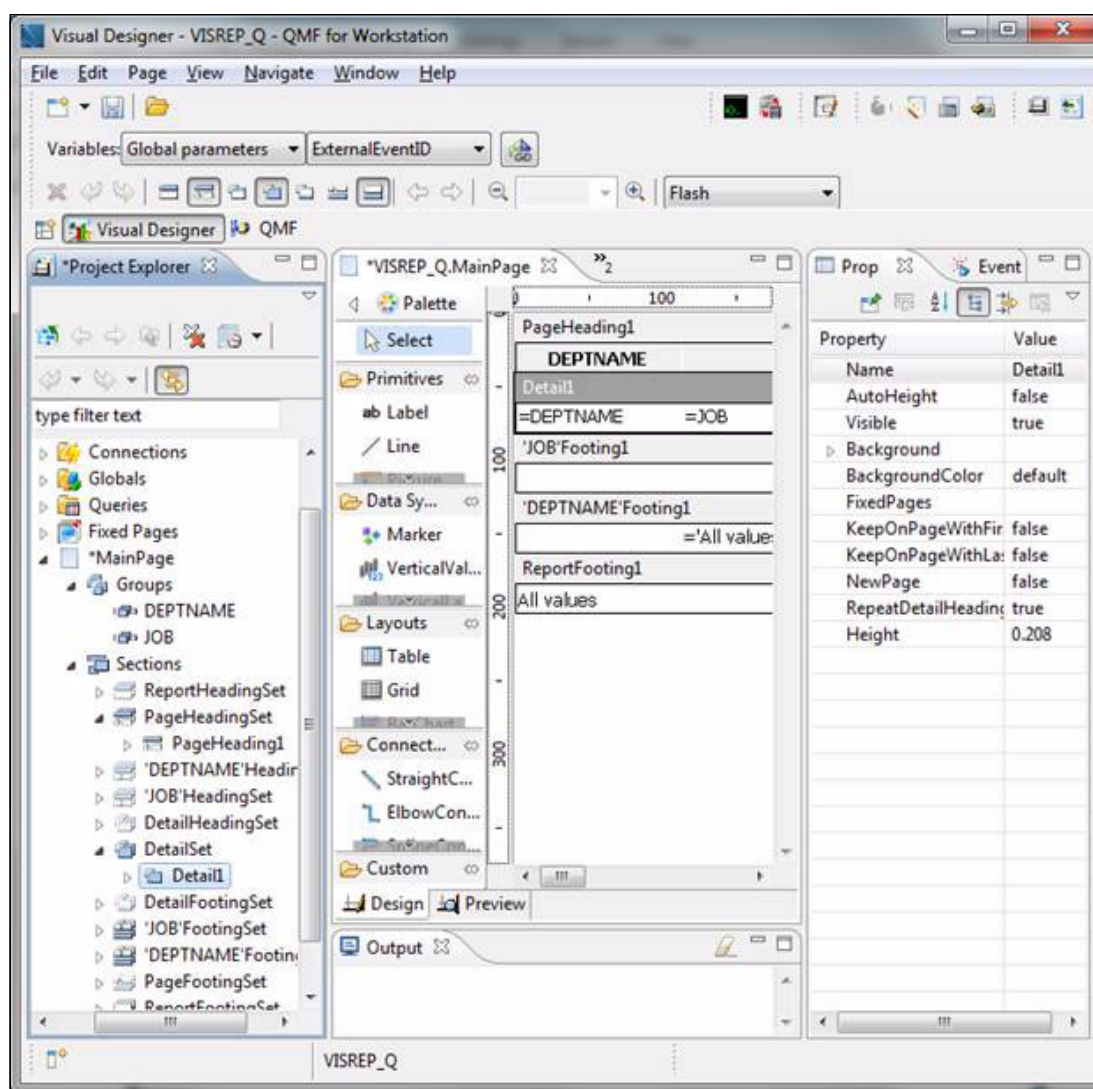


Figure 11-12 The Visual Designer perspective

Only those sections marked as visible, such as PageHeadingSet in the **Project Explorer** tree shown in Figure 11-13, are shown in the **Design** view and in the final report. The sections whose icons are disabled, such as ReportHeadingSet, are set to invisible and are not displayed in the report.

The expressions “=DEPTNAME” and “=JOB” at the right of Figure 11-13 here refer to the value of the DEPTNAME and JOB fields in the current record being formatted.

The Groups tree item contains the columns that were the expandable side groups in the query results grid (see Figure 11-9 on page 241).

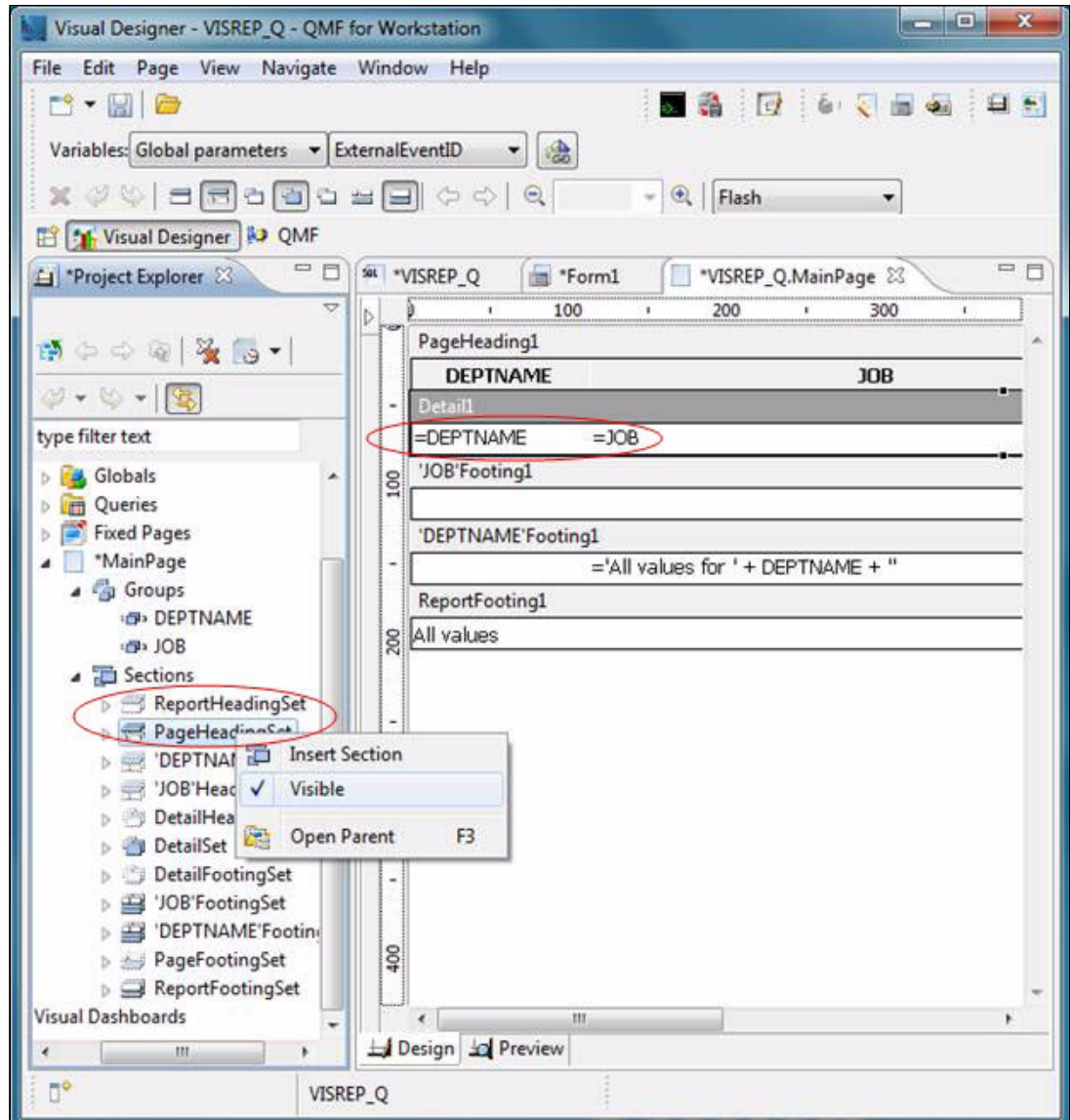


Figure 11-13 Making sections of the report visible

The pattern in the Project Explorer tree might be more obvious in the layout shown in Figure 11-14.

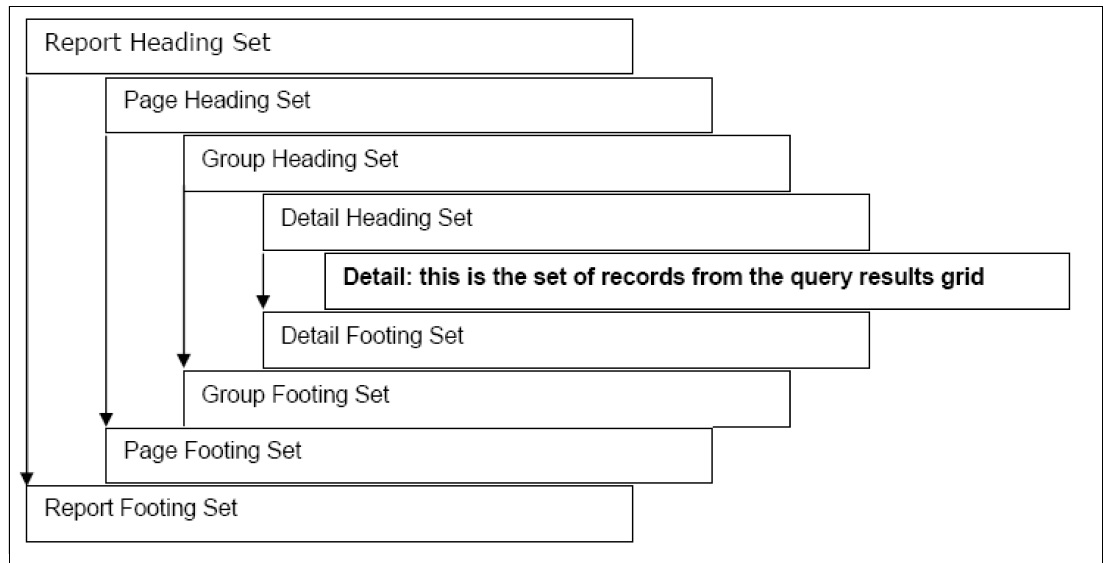


Figure 11-14 Sections of the report that are displayed in the Project Explorer tree

The visual report consists of a set of header-footer pairs framing the original records (Details) from the query results grid. The word “Set” refers to the collection of content that will make up that heading or footing.

11.2.3 Adding a heading with a graphic

To create the heading shown in the sample visual report shown in Figure 11-8 on page 240, follow these steps:

1. In the **Project Explorer** tree, right-click Report Heading Set and select **Visible** from the context menu, which causes the report heading area to appear on the report canvas in the **Design** view, as shown in Figure 11-15.

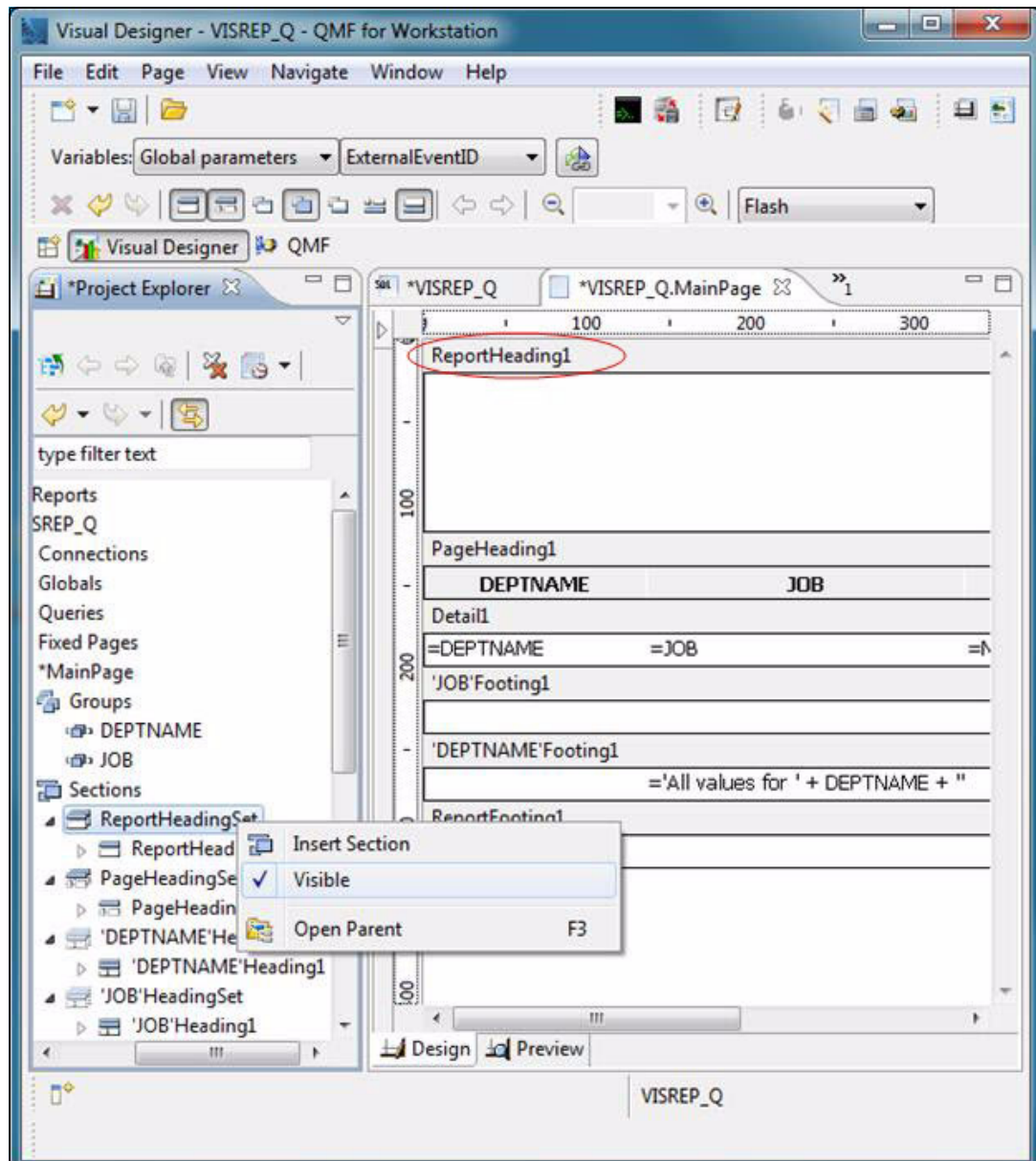


Figure 11-15 Setting the report heading to visible so that it appears on the report design canvas

2. Add a graphic to the report heading by following these steps:
 - a. To add the graphic to the global image library for the visual report, first expand **Globals**.
 - b. Right-click **Images** and select **Insert Image** from the context menu, as shown in Figure 11-16.

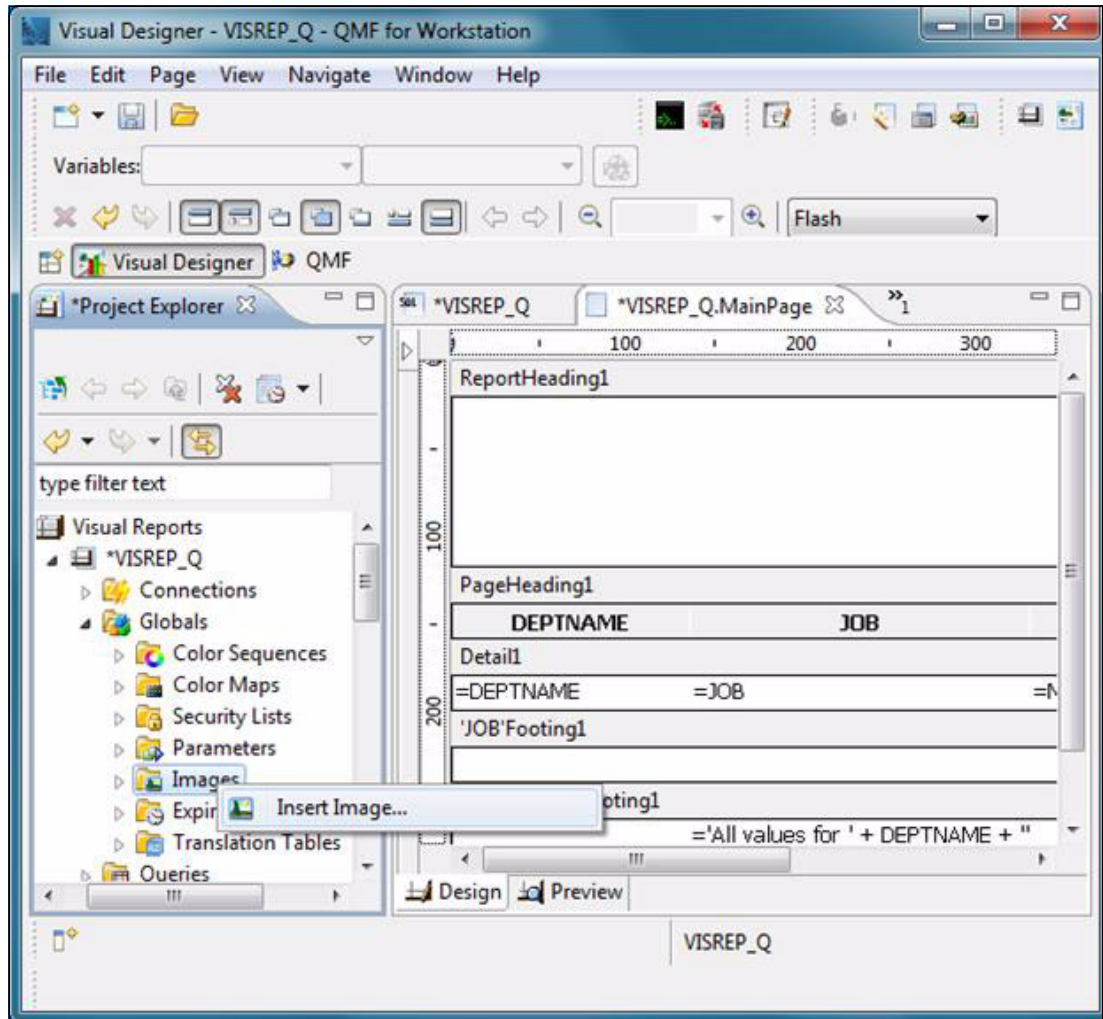


Figure 11-16 Adding an image to the global image library for the visual report

- c. Navigate to the image, as shown for our example graphic in Figure 11-17. Only check the box next to the image name if you need this image in the first section of the report.

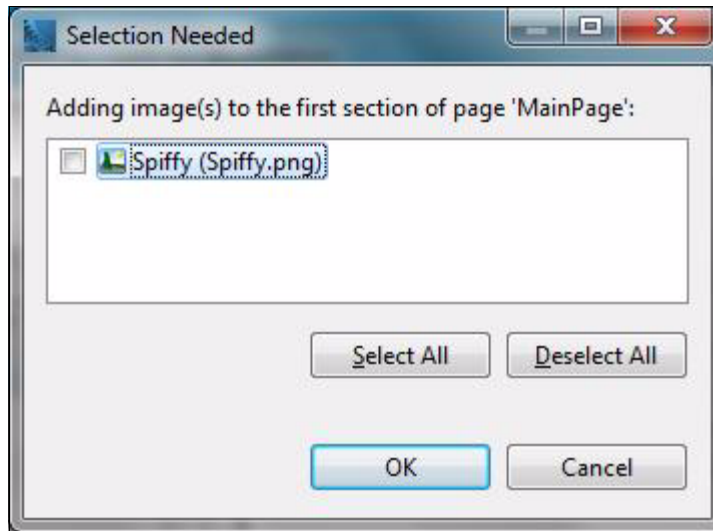


Figure 11-17 Adding an image named *Spiffy* to the report heading

For the sake of generality in this example, we add the image manually. To do so, drag the image to the report heading section on the report canvas, as shown in Figure 11-18.

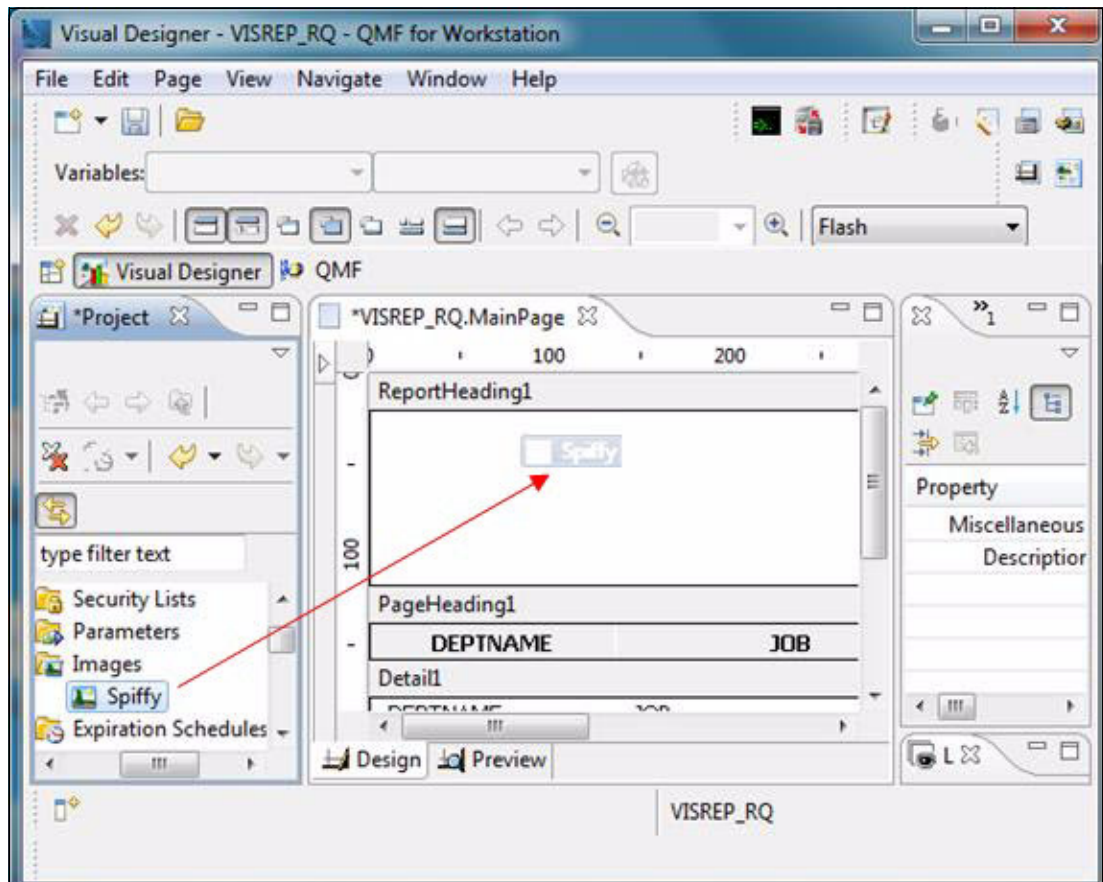


Figure 11-18 Dragging an image to the report heading section of the report canvas

3. Add report heading text in the following manner:
 - a. Click the **Show Palette** icon, highlighted in Figure 11-19.

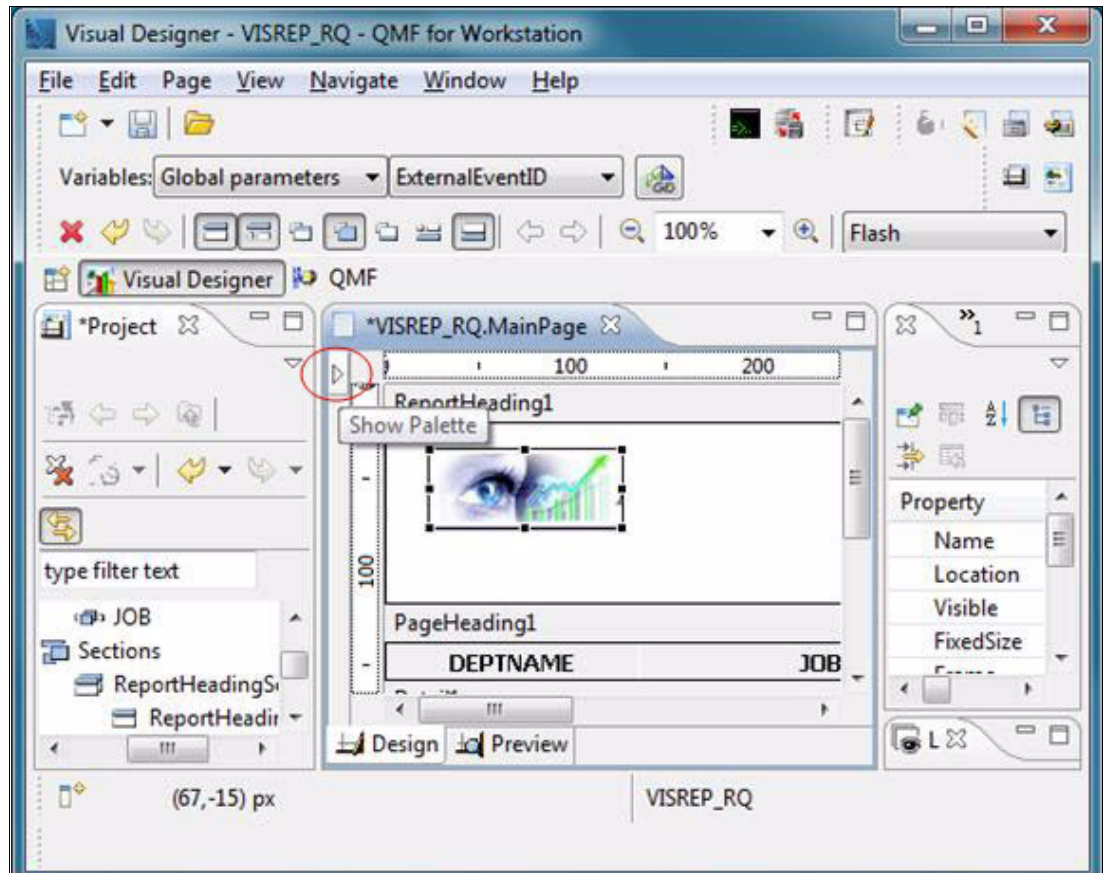


Figure 11-19 Displaying the Palette view

- b. In the **Palette** view, click the Label primitive, as shown in Figure 11-20, then click again in the report heading section of the report canvas to place the label on the canvas.

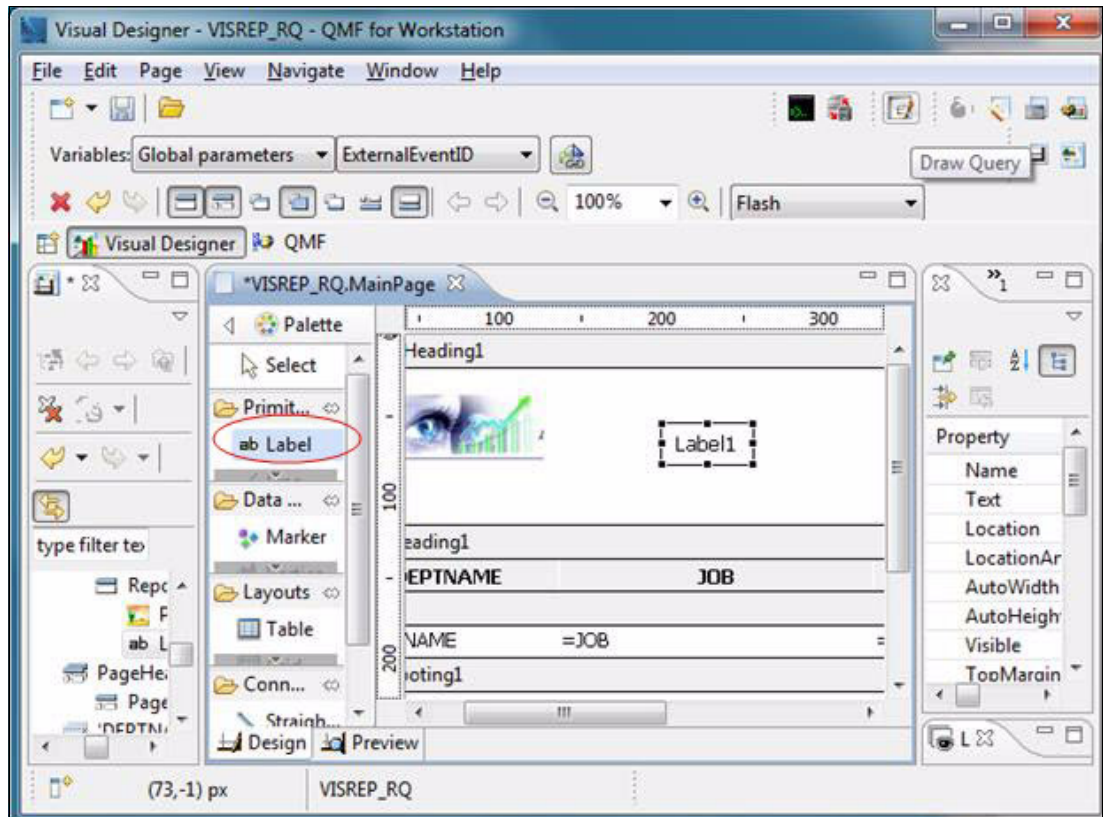


Figure 11-20 Adding a label primitive to the report heading section of the canvas for the text of the report heading

Tip: This area of the product is not a drag-and-drop interface. The action here is to click once on the label and once on the target area where you want to place the label.

- c. Click once on the text of the label on the canvas and change it to the text desired, as shown in Figure 11-21. In this example, we use “Spiffy Human Resources Report” for the heading. Resize the label as needed.

The new text displays in the **Text** property in the Properties view, as shown in Figure 11-21. Alternatively, you can enter the text in this field instead of clicking in the label text on the report canvas.

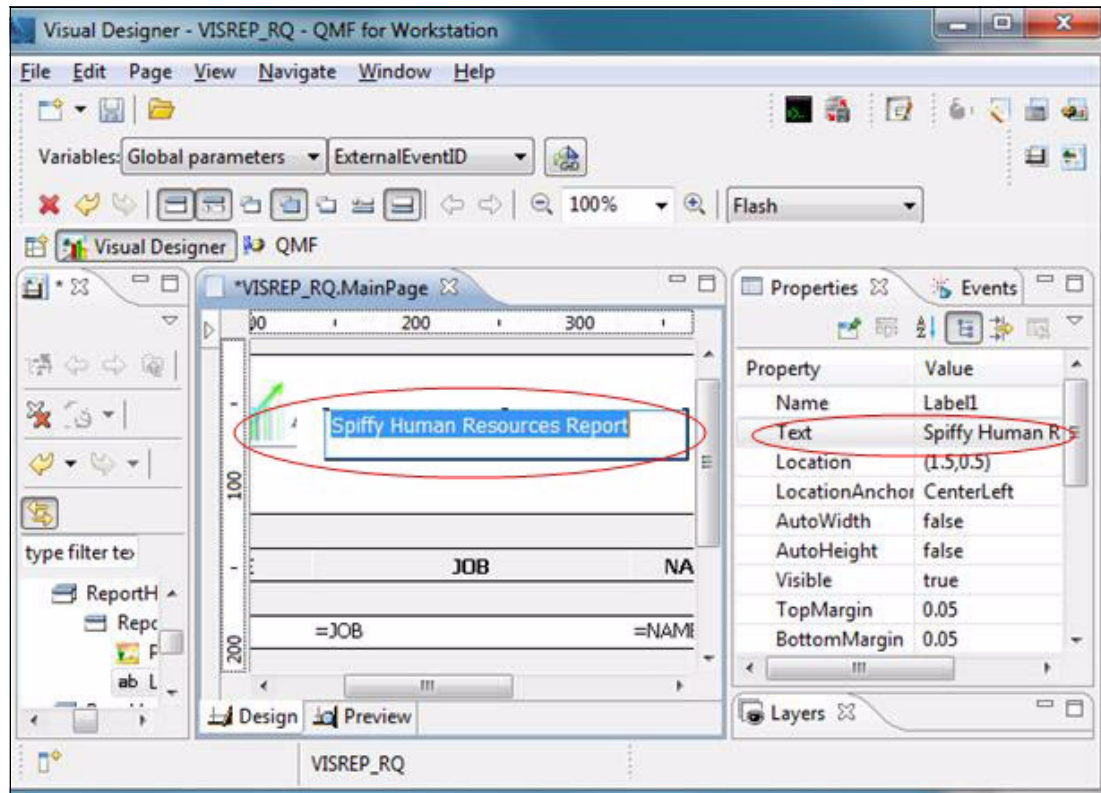


Figure 11-21 Changing the report heading text on the label

d. Expand the Font property in the **Properties** view, as shown in Figure 11-22.

Set the font properties for the report heading as follows:

Bold = true

Color = red

Size = 18 pt

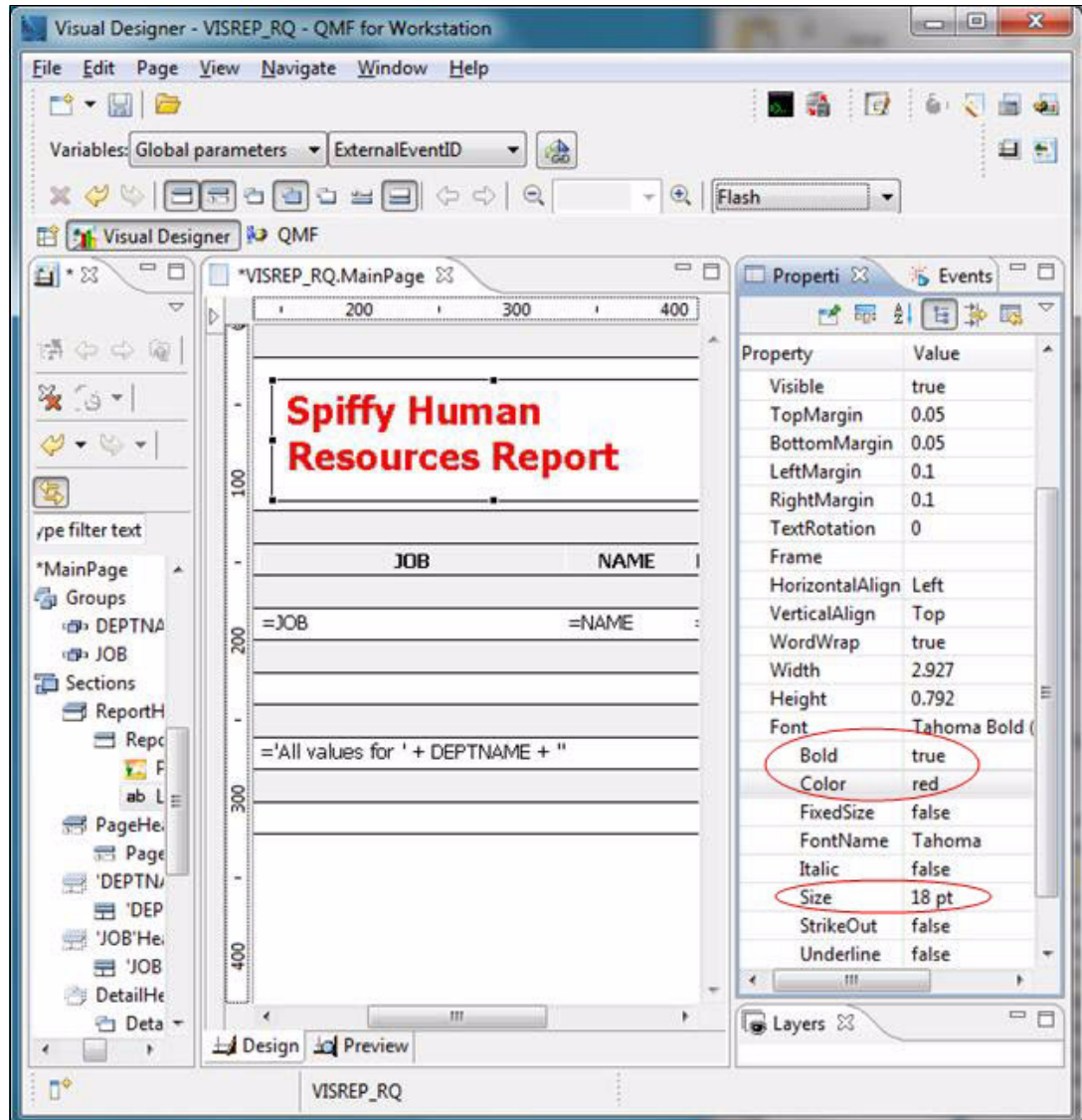


Figure 11-22 Setting font properties for the report heading

- e. Click the **Preview** tab to see how the report will look when run. Our preview is shown in Figure 11-23.

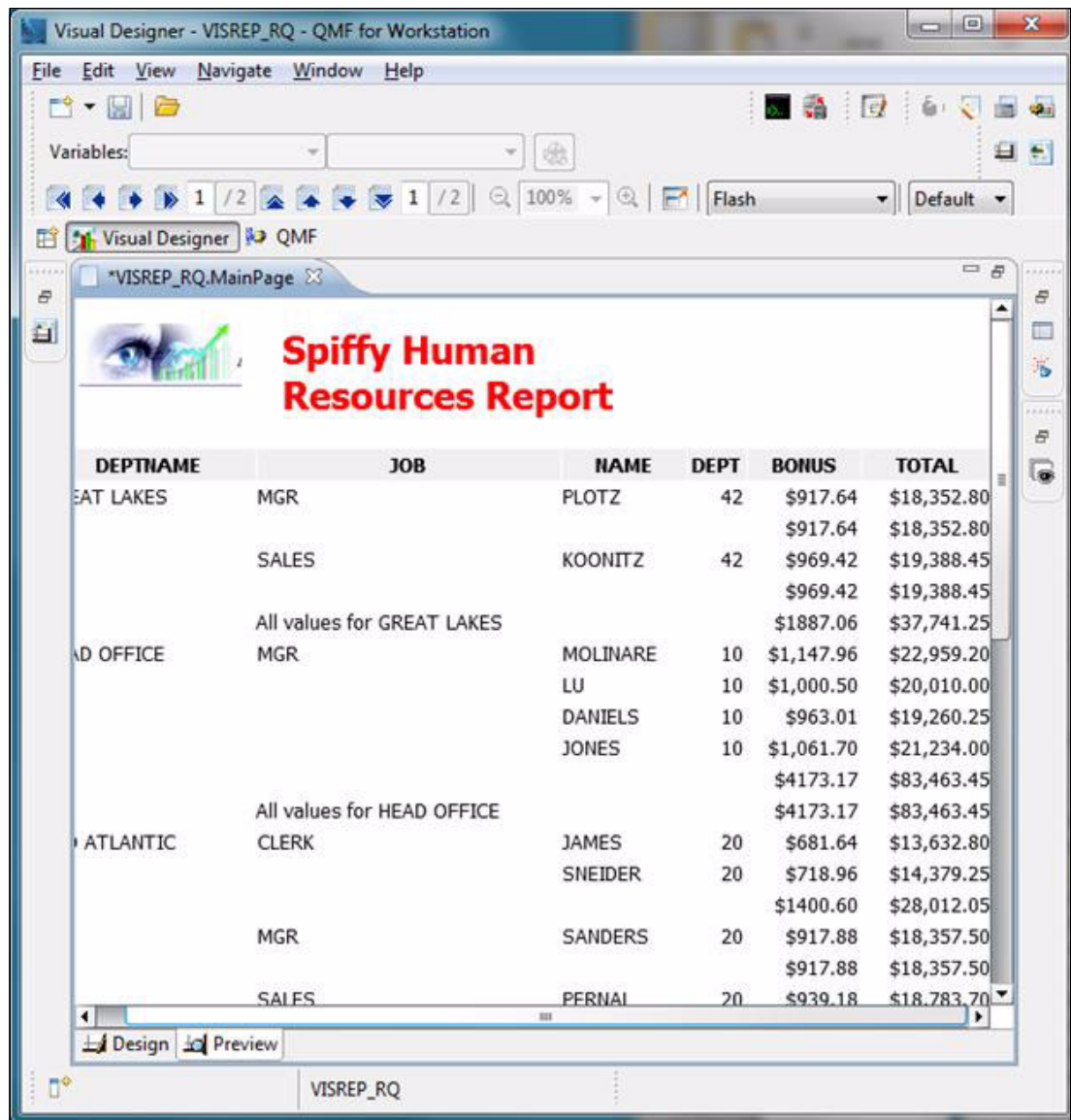


Figure 11-23 Previewing the report

11.2.4 Adding group summary highlighting and labels

In Figure 11-23, notice that there is no clear distinction between the report detail data and the group summaries. To add group summary label text and make it stand out in the report, we take the following steps:

1. Click the **Design** tab again.
2. Choose the section you want to work with and make it the active section.

Any section can be made the active section for editing by clicking the title bar of the section. In this example, we click the DEPTNAME Footing 1 title bar, as shown in Figure 11-24.

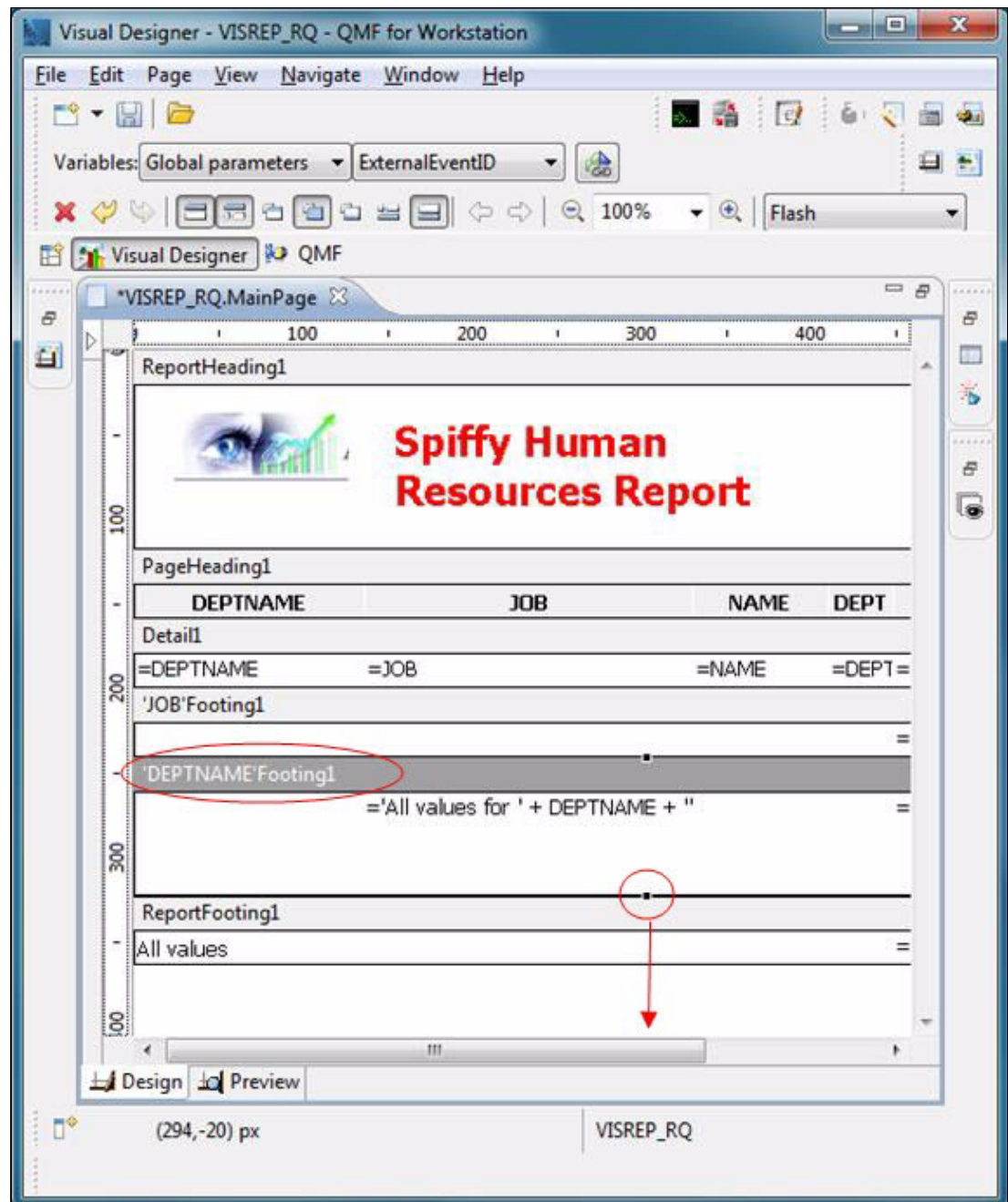


Figure 11-24 Clicking the title bar of the “DEPTNAME Footing 1” section to make it the active section

3. Expand the active section by dragging the lower handle (small black square on the outline, highlighted in Figure 11-24) down. In our case, we drag it down so that the height is roughly doubled.
4. Add a separator for the group summary footing.

As shown in Figure 11-25, we select the Line object from the **Primitives** palette to add a line that will act as a separation line for the DEPTNAME footing.

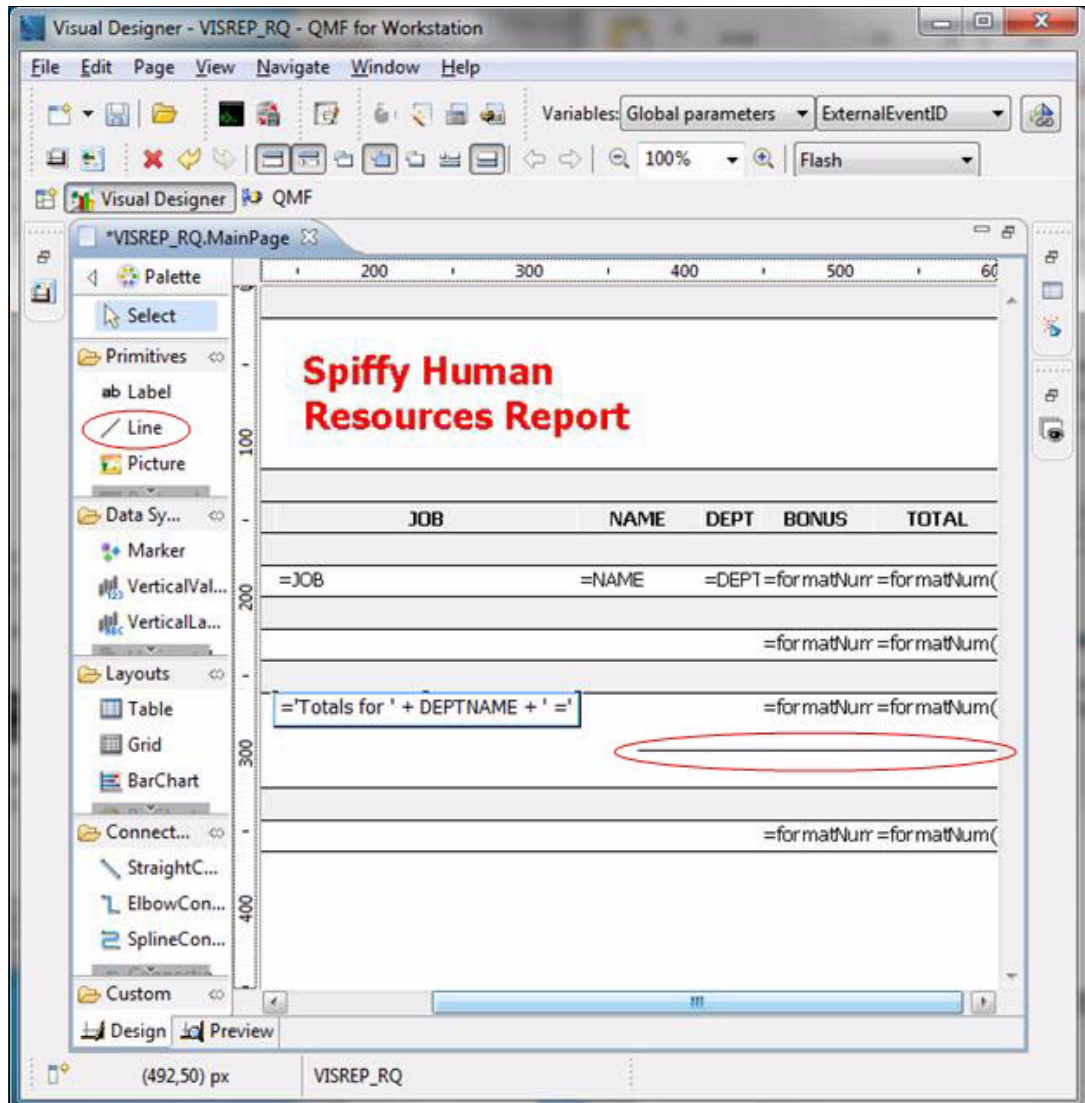


Figure 11-25 Adding a separation line for the DEPTNAME footing

5. Add a group summary label as desired.

To do so in our example:

- Click the text in the label in the DEPTNAME footing and change it to the following, taking care to include all characters and spaces shown:
 ='Totals for ' + DEPTNAME + ' ='
- Click the **Preview** tab again to see how the settings affect the report, as shown in Figure 11-26.

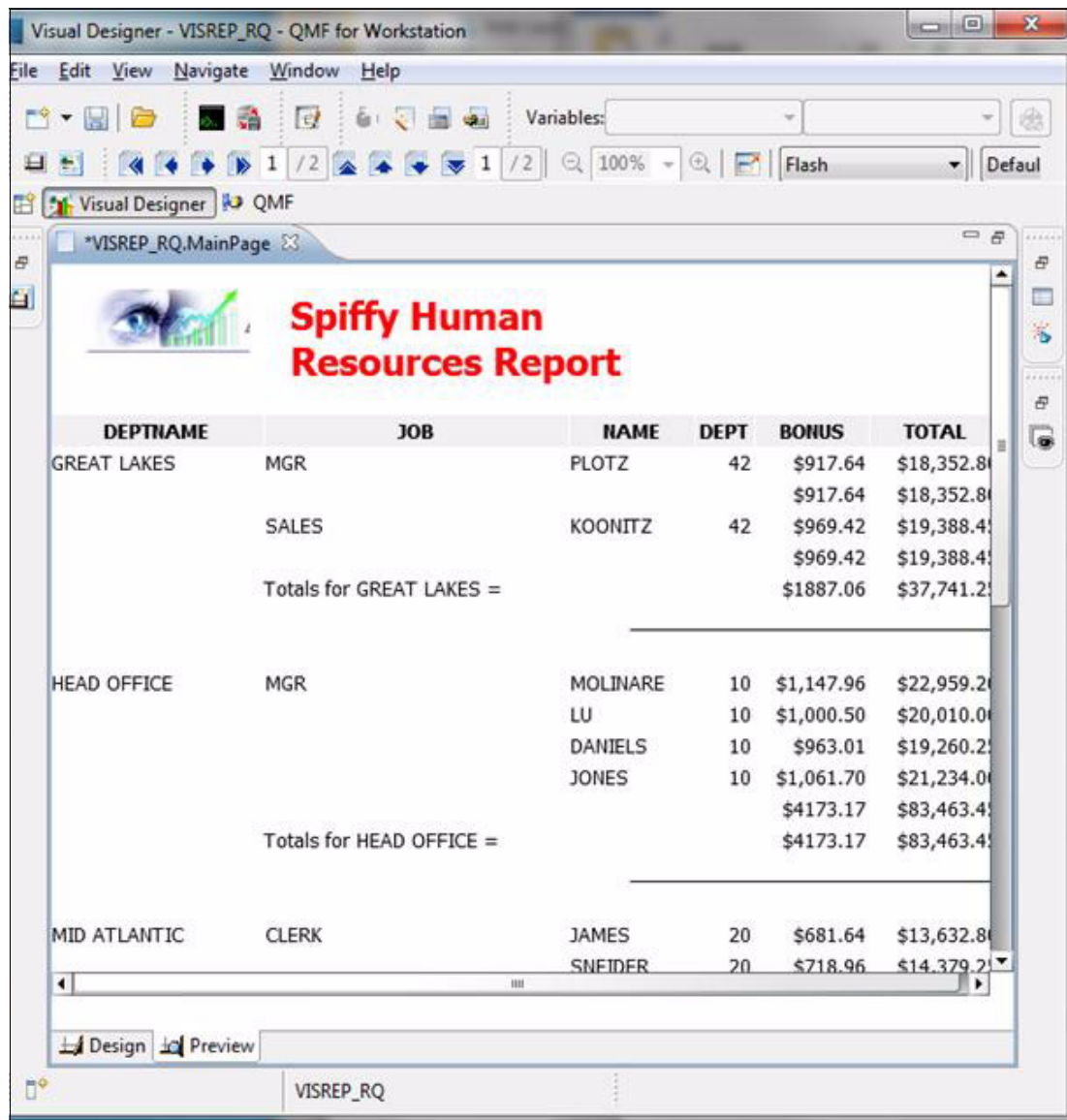


Figure 11-26 Previewing the separation line and text for the DEPTNAME footing

6. Add highlighting to the footing label as desired.

To do so in our example:

- Click again on the **Design** tab.
- While holding down the Ctrl key, select the three labels in the DEPTNAME footing.
- In the Properties view, expand **Frame** → **FillStyle** → **FillColor**.
- Click **white**, which is the default color, then click the selection button that appears and select a new color. We select yellow for our example, as shown in Figure 11-27.

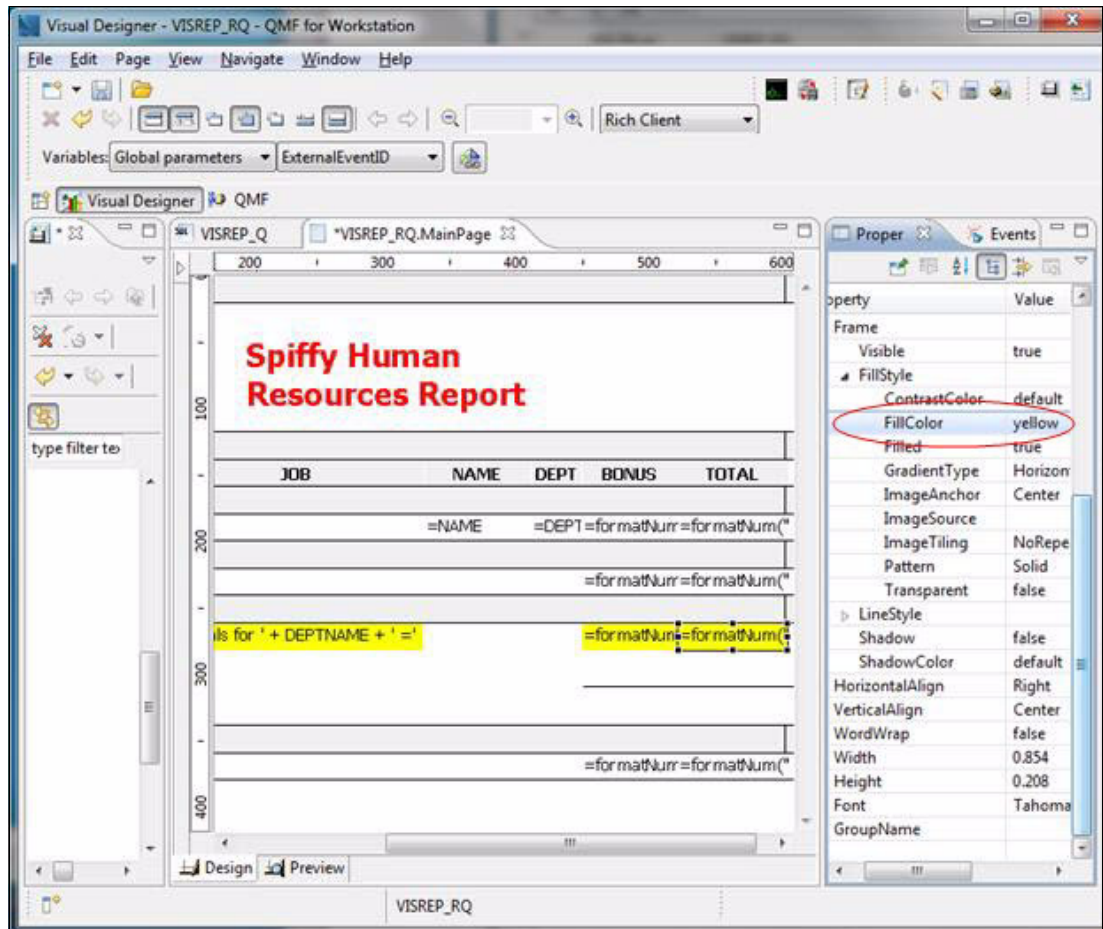


Figure 11-27 Setting highlighting for the DEPTNAME footing text

7. Reposition the label (= 'Totals for ' + DEPTNAME + ' =') as desired.

In our case, we move it to the right to be contiguous with the other labels, as shown in Figure 11-28.

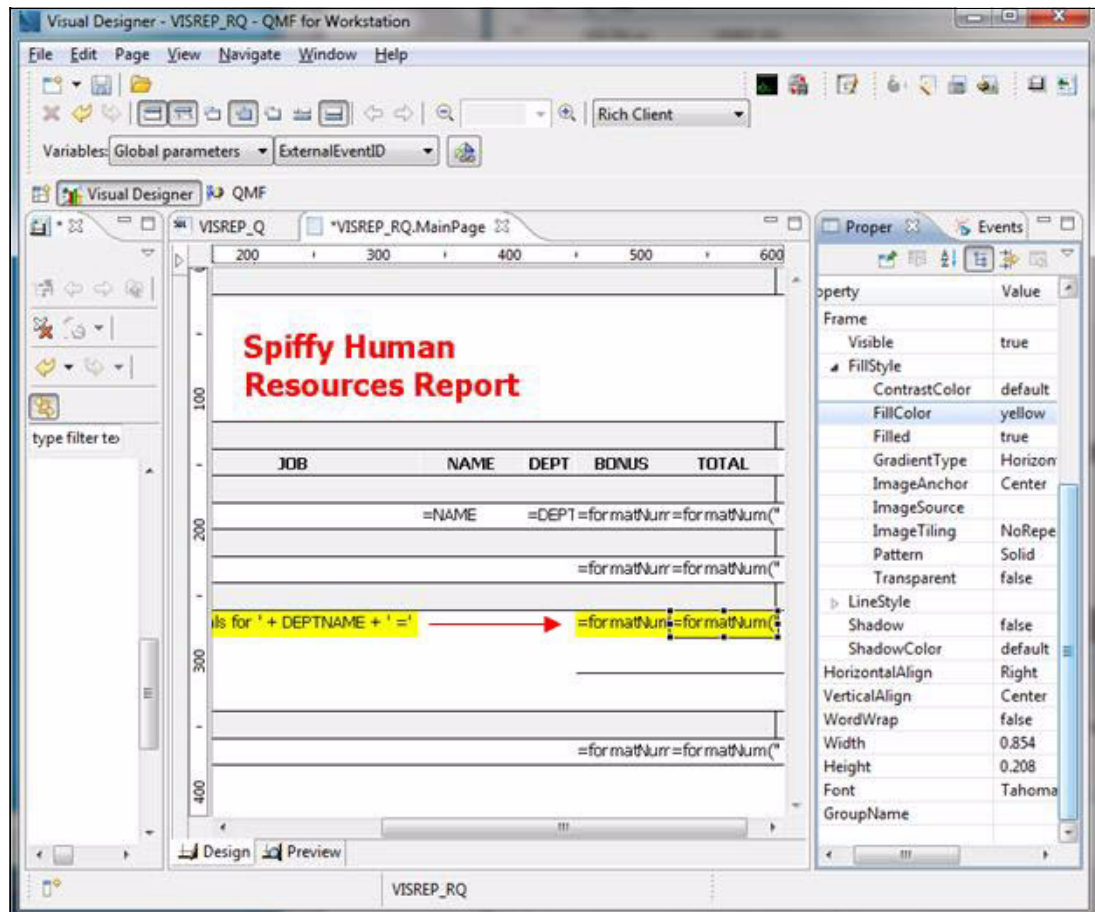


Figure 11-28 Repositioning the DEPTNAME footing label

Now we click the **Preview** tab again to see how it looks (Figure 11-29).

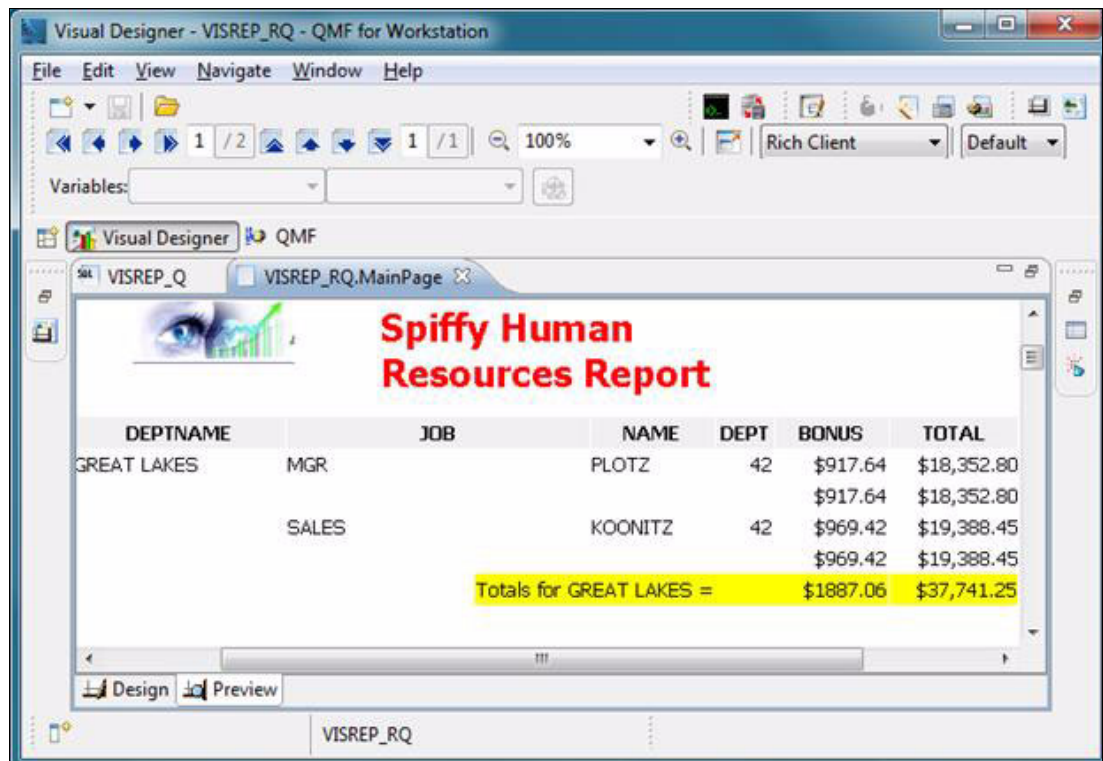


Figure 11-29 Previewing group summary text

11.2.5 Adding conditional formatting

Conditional formatting, also called threshold formatting, is often needed in reports such as the one shown in our example before. To add threshold formatting that highlights in red all bonuses above a threshold of \$1000, we take the following steps:

1. Click the **Design** tab again.
2. Locate DetailSet in the **Project Explorer** tree. Expand Detail1 and select BONUS, as shown in Figure 11-30.

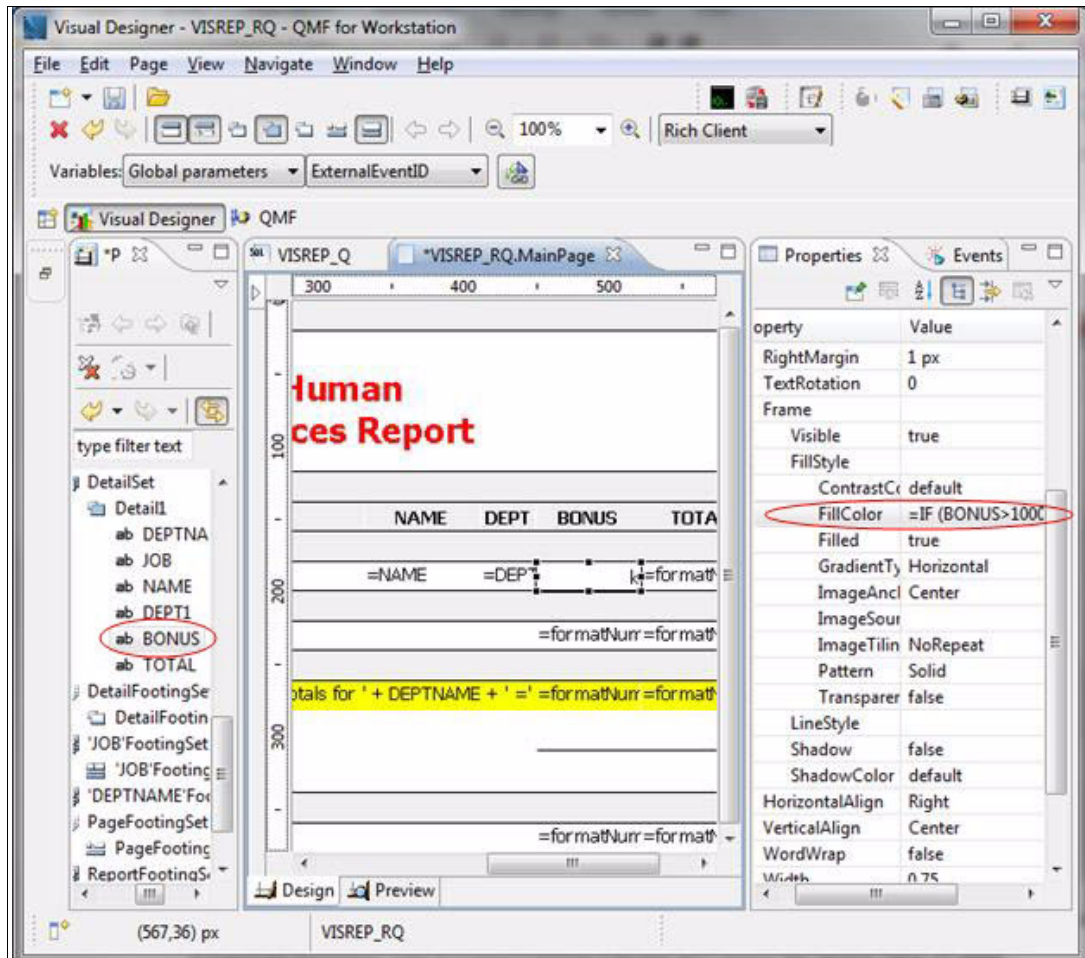


Figure 11-30 Adding a condition to the report

3. In the **Properties** view, expand **Frame** → **FillStyle** → **FillColor** and replace the default value of **white** with the following text, as shown in Figure 11-30.
=IF (BONUS>1000,red,white)

Click the **Preview** tab to see the formatted report, shown in Figure 11-31.

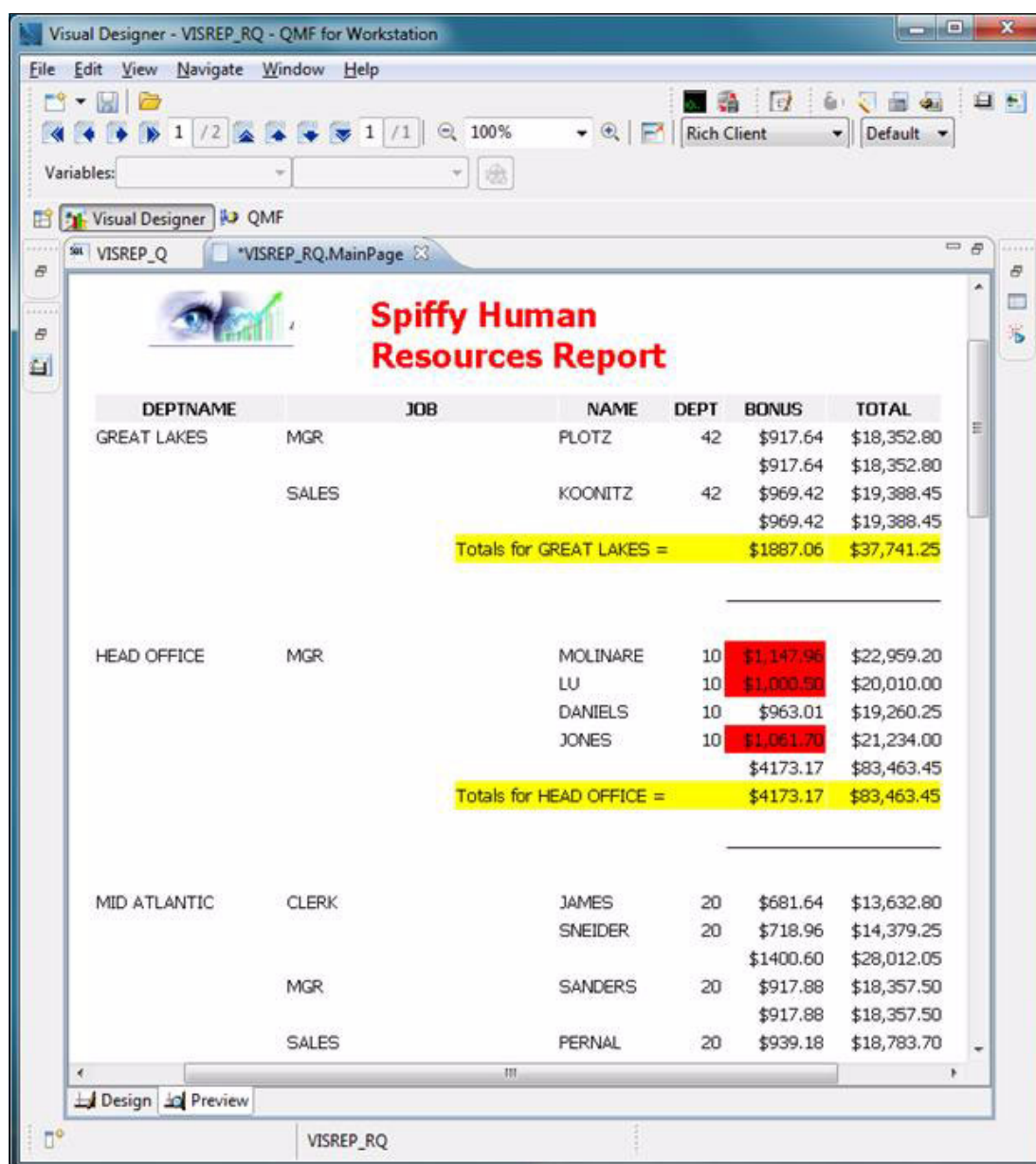


Figure 11-31 Report after conditional formatting has been applied to highlight bonuses in excess of \$1000

11.2.6 Including data from additional queries

So far, the work shown is in a part of the visual report called the main page. The main page is set up to format a single query as specified in the MainPage properties sheet, shown in Figure 11-32.

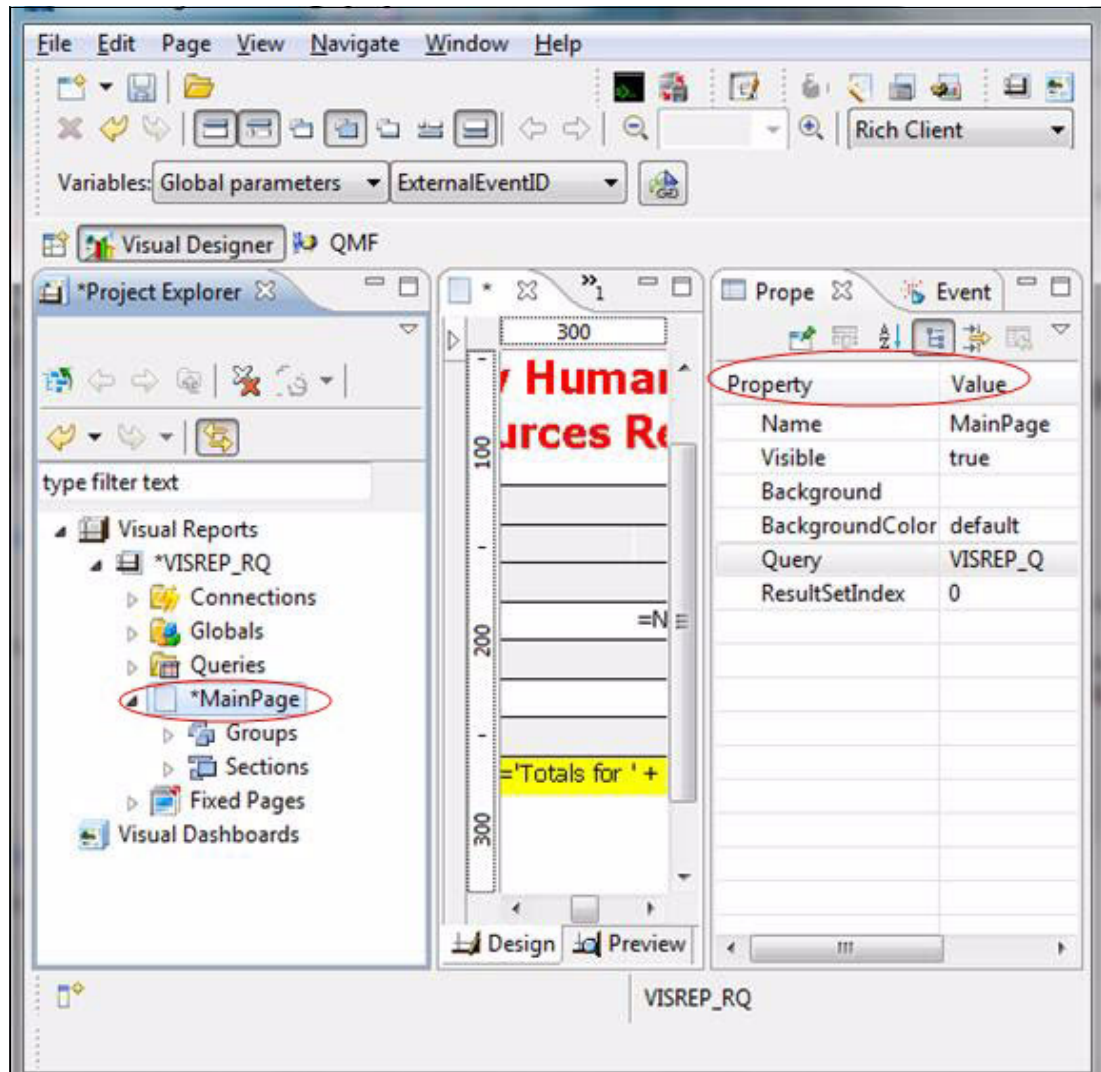


Figure 11-32 MainPage properties sheet

A visual report can contain results from more than one query. These other queries cannot be used to drive the sections of the main page. They are not in scope for the main page and cannot be embedded in groups, formulas for labels, and other items. Instead, the items in the Layouts palette can act as containers for the result sets of these other queries. These items can then be placed in any of the report sections, such as group headings or footings or report headings or footings.

The Staff Report in the sample database provided with QMF shows an example of using more than one set of query results in a visual report.

11.2.7 Including fixed pages in the report

Visual reports have a feature called a fixed page. Unlike the main page, the fixed page is not powered by a query, so it cannot reference result sets directly. The FixedPage item in the **Project Explorer** tree allows you to make a single page such as a title page or a summary page for the report or for any section. Like the report sections, it can hold static text or layouts that are powered by any of the queries.



Working with procedures

A procedure is a QMF object that enables you to issue multiple commands with a single RUN command. Any number of queries and reports (and even other procedures) can be run from within a single procedure.

Like QMF for TSO and CICS, QMF for Workstation and QMF for WebSphere offer *linear procedures*, which can contain only QMF commands. Additionally, QMF for Workstation on the Windows operating system offers *procedures with logic*, which can contain QMF commands as well as REXX statements and logic (when Open Object REXX for Windows is installed on the same Windows machine as QMF for Workstation).

This chapter provides the following information:

- ▶ Explain how to create a procedure
- ▶ Provide detailed examples of both linear procedures and procedures with logic
- ▶ Explain how QMF searches for objects referenced by a procedure and show you how to use object keys or relative paths in the procedure to uniquely reference like-named repository objects
- ▶ Show how to schedule procedures to run at a particular time

12.1 Creating a procedure

The Procedure window is used to create, open, display, and run procedures. Procedures can be saved in a repository, in the QMF catalog, or to a file. All commands issued through procedures are governed by your resource limits.

Procedures can contain the following items:

- ▶ Any QMF procedure command:

This chapter shows examples of the most commonly used procedure commands. For a full set of commands that can be used in procedures, see *Getting Started with QMF for Workstation and QMF for WebSphere* at the following website:

<http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp>

- ▶ Comment lines
- ▶ Blank lines
- ▶ RUN commands that run other procedures or queries
- ▶ Substitution variables

To create a procedure:

1. From the User perspective, select **File** → **New** → **Procedure**. The Create Procedure wizard opens. (See Chapter 5, “Installing QMF for Workstation, QMF for WebSphere, and touring the interfaces” on page 65 for how to switch perspectives.)

You can also select the **New Procedure** icon from the menu bar.

2. Specify a name for the new procedure in the Name field.
3. Select from the list of available data sources in the Data source field to specify where the new procedure will reside if saved. If you leave this field blank, the new procedure will be saved to the current data source.
4. Click **Finish**. The new procedure object is opened in the editor view of the User perspective.
5. In the Procedure editor, enter the commands that will be included in the procedure.

Examples of commands in linear procedures and procedures with logic are provided in 12.1.1, “A sample linear procedure” on page 267 and 12.1.2, “A sample procedure with logic” on page 271. If you are creating a procedure with logic using REXX statements, you must type a REXX comment line as the first line of the procedure. The REXX comment line must appear as follows:

```
/*REXX*/
```

If you are including QMF procedure commands in addition to REXX statements, the QMF procedure commands must be enclosed in single or double quotes.

Also keep in mind that procedures are often operating-system-specific and therefore require editing before they can be run in environments for which they were not originally designed. For example, suppose that you have a procedure that was originally designed for QMF on TSO and you now want to run it in QMF for Workstation. If you will be storing any object referenced by the procedure in the QMF for Workstation repository instead of the QMF catalog, you will need to uniquely reference the object if there are multiple repository objects with that name. More information about how to do it and why it is necessary is given in 12.2, “How QMF finds objects referenced by the procedure” on page 272 and 12.3, “Uniquely referencing objects in the procedure” on page 273.

6. Save, run, or print the procedure.

After running a procedure, results are returned in the active editor window. If you are running a procedure with logic, communication with the procedure is handled through the REXX Console view.

12.1.1 A sample linear procedure

Figure 12-1 shows an example of a linear procedure. The numbers in the figure correspond to the following explanations. This procedure follows these steps:

1. Sets up two global variables:
 - Global variable `DSQQW_RPT_USE_PS` specifies the page formatting options to use when printing a report with the `PRINT REPORT` command. The value 1 specifies to use the page setup options specified in the form.
 - Global variable `&DEPARTMENT` is a user-defined variable in the `WHERE` clause of the `GETSTAFF` query referenced in the procedure. In this example, data for Department 10 will be returned to the procedure.
2. Connects to a DB2 data source and runs the `GETSTAFF` query.
3. Connects to a different data source (RS25) and saves the data retrieved from the `GETSTAFF` query into a table on RS25 called `TEST`. This `SAVE DATA` command creates the table if it does not exist (or deletes all records from the table if it does exist), then loads the new data. In this particular case, the table is not qualified with a schema, so each user will get a unique table.
4. Runs a query called `TESTJOIN` that joins a pre-existing table at the RS25 data source with the new `TEST` table, which was just loaded. The result set is then exported to a CSV file (Excel and Lotus compatible format) on the local Windows machine.
5. Produces a visual report that uses these tables and exports it in PDF format to a shared file server and also to the local file system.
6. Prints the visual report to a local printer.
7. Launches the CSV and PDF outputs on the local machine by the `EXECUTE` command, which runs local operating system commands. This command can launch any OS command: `FTP`, `*.exe`, `*.bat` (Windows), `*.sh` (UNIX, Linux), and others.
8. Through the `MAIL TO` command, sends the PDF as an email attachment to a particular user as well as others on the cc: list.

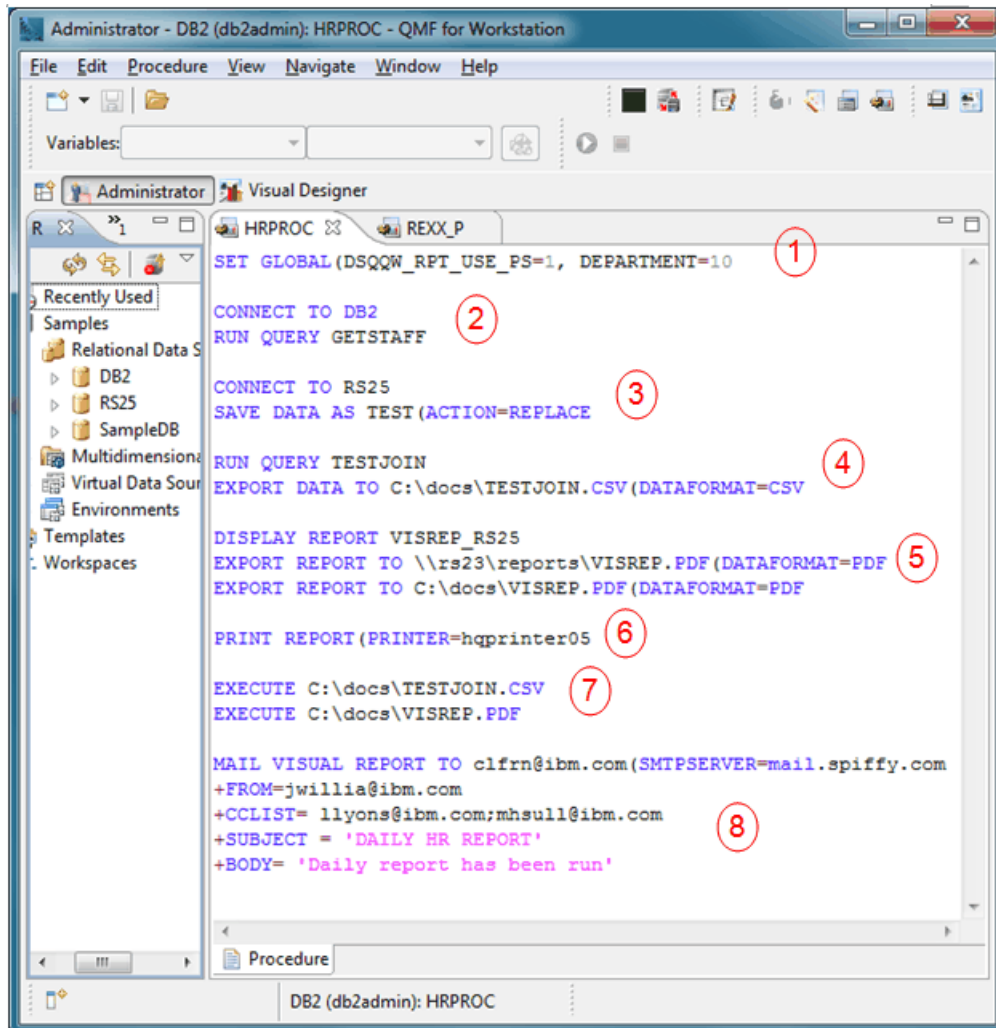


Figure 12-1 A sample linear procedure

Figure 12-2 through Figure 12-5 show the output from the procedure.

Figure 12-2 shows the output files on the local Windows machine residing on the local Windows machine from Step 7 of the linear procedure in Figure 12-1 on page 268.

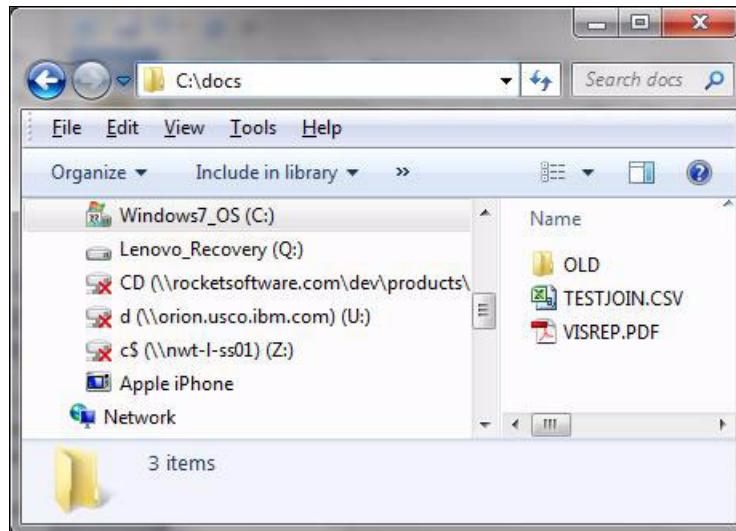


Figure 12-2 Output files residing on the local Windows machine from the linear procedure

Figure 12-3 shows the CSV file output from the linear procedure shown in Figure 12-1 on page 268.

	A	B	C	D	E
1	NAME	DEPT	JOB	TOTAL	DEPTNAME
2	MOLINARI	10	MGR	22959.2	HEAD OFFICE
3	LU	10	MGR	20010	HEAD OFFICE
4	DANIELS	10	MGR	19260.25	HEAD OFFICE
5	JONES	10	MGR	21234	HEAD OFFICE
6					

Figure 12-3 CSV file output from the linear procedure

Figure 12-4 shows the PDF file output from the linear procedure shown in Figure 12-1 on page 268.

The screenshot shows the Adobe Reader interface with a PDF document titled 'VISREP.PDF'. The document contains a report titled 'Spiffy Human Resources Report' with a table of employee data. The table has columns for DEPTNAME, JOB, NAME, DEPT, BONU, and TOTAL. The data is as follows:

DEPTNAME	JOB	NAME	DEPT	BONU	TOTAL
HEAD OFFICE	MGR	MOLINARE	10	\$1,147.96	\$22,959.20
		LU	10	\$1,000.50	\$20,010.00
		DANIELS	10	\$963.01	\$19,260.25
		JONES	10	\$1,061.70	\$21,234.00
				\$4173.17	\$83,463.45
Totals for HEAD OFFICE =				\$4173.17	\$83,463.45
All values				\$4173.17	\$83,463.45

Figure 12-4 PDF file output from the linear procedure

Figure 12-5 shows the email with the PDF attachment from Step 8 of the linear procedure shown in Figure 12-1 on page 268.

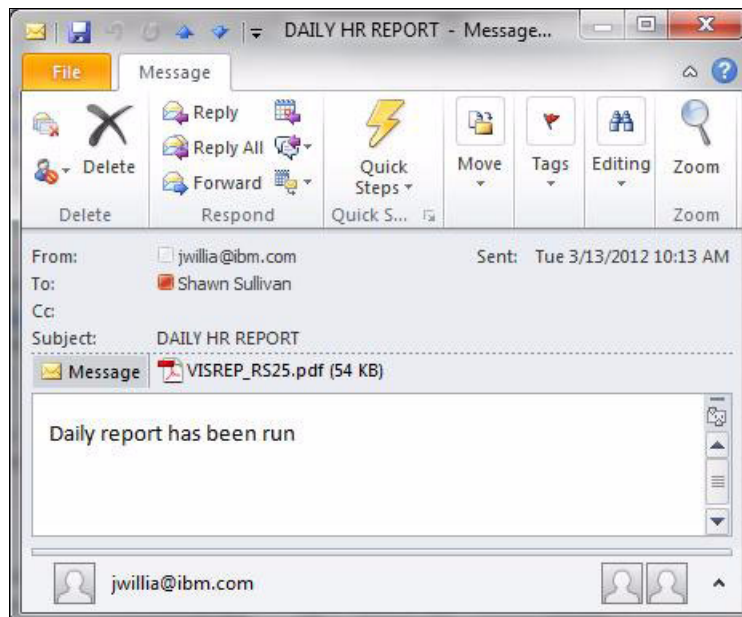


Figure 12-5 Emailed output from the MAIL TO command in the linear procedure

12.1.2 A sample procedure with logic

Procedures with logic can run only in the QMF for Workstation Windows environment and require Open Object REXX. REXX supports logic scripting, such as IF-THEN-ELSE statements, branching CALL statements, iterative DO loops, and more.

Figure 12-6 shows a sample QMF for Workstation procedure with logic:

1. The user is prompted for a number of reports to run and whether employee number 20 should be skipped.
2. Depending on the answers, the same query is run repeatedly, using a different employee ID each time.

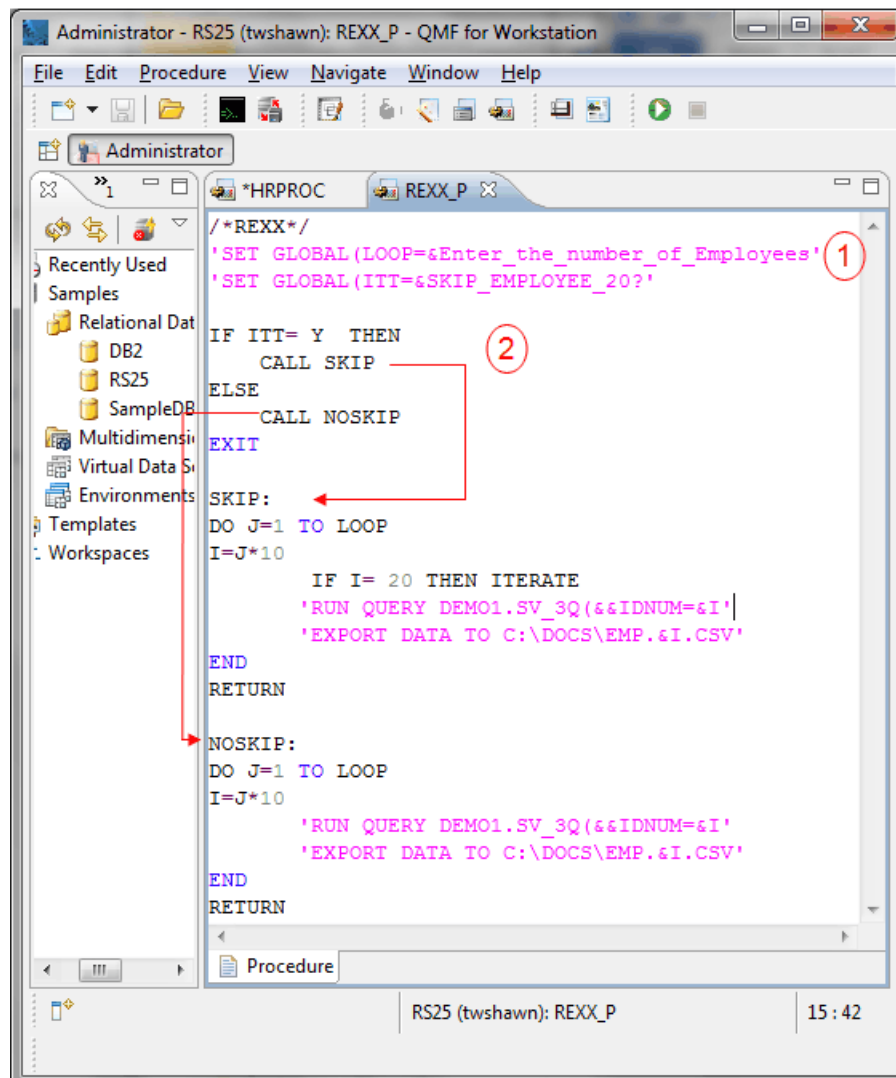


Figure 12-6 A sample procedure with logic

If we input **5** for the number of employees and respond with **Y** to the question about skipping employee number 20, the outputs that we receive are shown in Figure 12-7.

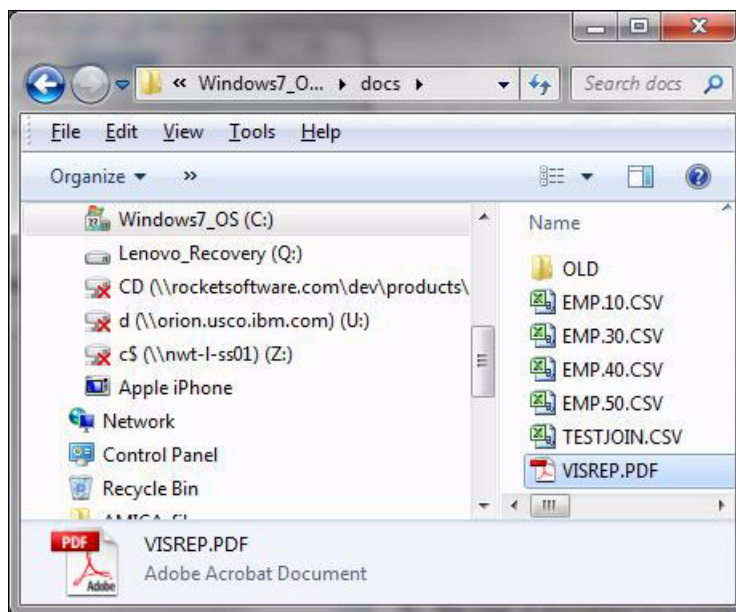


Figure 12-7 Outputs from the sample procedure with logic

For more information about Open Object REXX, see the following website:

<http://www.oorexx.org/>

12.2 How QMF finds objects referenced by the procedure

When a procedure that is stored in a repository workspace references objects by their unqualified names (for example, RUN QUERY Q1), QMF:

1. Searches the QMF catalog:
For example, if the procedure includes the line RUN QUERY Q1 and the current user is DB2ADMIN, QMF searches the QMF catalog for the query DB2ADMIN.Q1.
2. Searches the workspace folder containing the procedure that references the object.
3. Searches recursively in the workspace folders with increasingly widening scope.

In the event that multiple objects with the same name will be created and stored in different folders, it is best practice to uniquely reference procedure objects. The next topic looks at ways to do this.

12.3 Uniquely referencing objects in the procedure

The best way to ensure that QMF locates the correct objects for the procedure is to reference them uniquely. There are two ways you can do this:

- ▶ Use the object's Key property to point to the object.
- ▶ Use the object's relative path to point to the object. The same relative path notation that is used in Windows and UNIX environments is used in QMF.

12.3.1 Using the Key property to point to the object

When an object is selected in the repository tree, the Properties view for that object contains a field called Key. This object key is unique and can be used in a procedure to reference the object.

For example, the key for the report named VISREP_RS25, shown in Figure 12-8, is `qmf:/.workspaces/Business Analyst View/VISREP_RS25`.

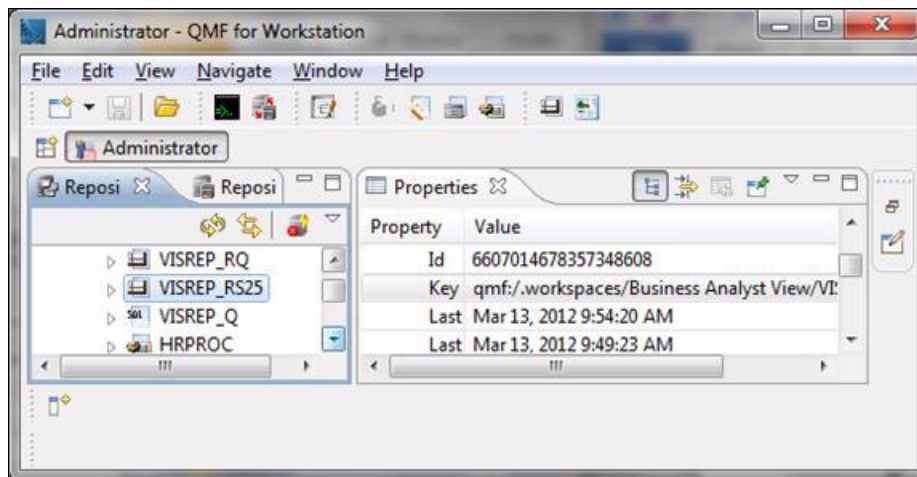


Figure 12-8 The Key property for an object can be used to uniquely reference it

With this object key, the procedure command would be as follows:

```
DISPLAY REPORT 'qmf:/.workspaces/Business Analyst View/VISREP_RS25'
```

The key is delimited with single quotes because it has spaces in the name.

Objects stored in the QMF catalog do not need to have keys because the QMF catalog automatically enforces uniqueness and it is searched first.

12.3.2 Referencing the object by its relative path

If the object's key is too unwieldy to use in the procedure, you can instead reference the object by its relative path.

For example, in Figure 12-9, PROC1 at the right of the figure shows the correct relative path for query Q1 in the folder structure shown at the left of the figure.

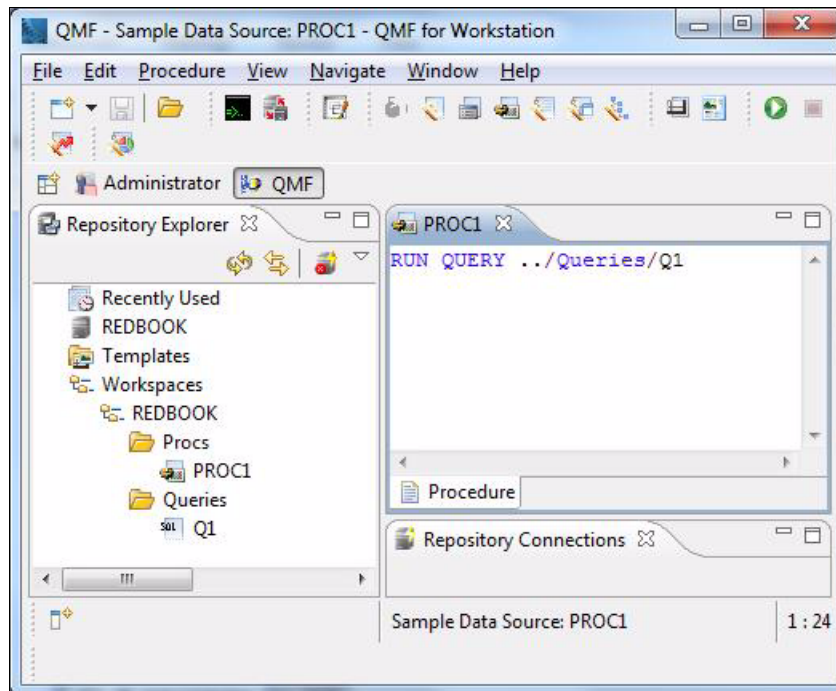


Figure 12-9 Query Q1 located by its relative path

12.4 Scheduling a procedure

For many customers, the ultimate goal of developing procedures is to run them unattended. QMF for Workstation provides a scheduling function that allows you to set up a job to be run by the local operating system scheduler.

To schedule a procedure to be run:

1. Select **View** → **Scheduled Jobs** from the menu.
2. In the Scheduled Jobs window, click the **Add Job** icon. The Schedule New Job window is displayed, as shown in Figure 12-10. This window provides a simple order form dialog to set up the job to be run.

Figure 12-10 shows the "Schedule New Job" dialog box. The "Job name" field is set to "HRREPORT". Under "Login Configuration", the "Repository connection" is set to "Samples". The "Run Procedure" section has "From Repository:" selected, with the path "qmf:/workspaces/Business Analyst View/HRPROC" entered. The "Variables" section is empty. The "Schedule" button is visible at the bottom.

Figure 12-10 Scheduling a procedure to be run

3. Complete the dialog and click **Schedule**. QMF compresses and encrypts the file, then passes the file and its associated Run command to the operating system scheduler. The Windows scheduler is shown in Figure 12-11.

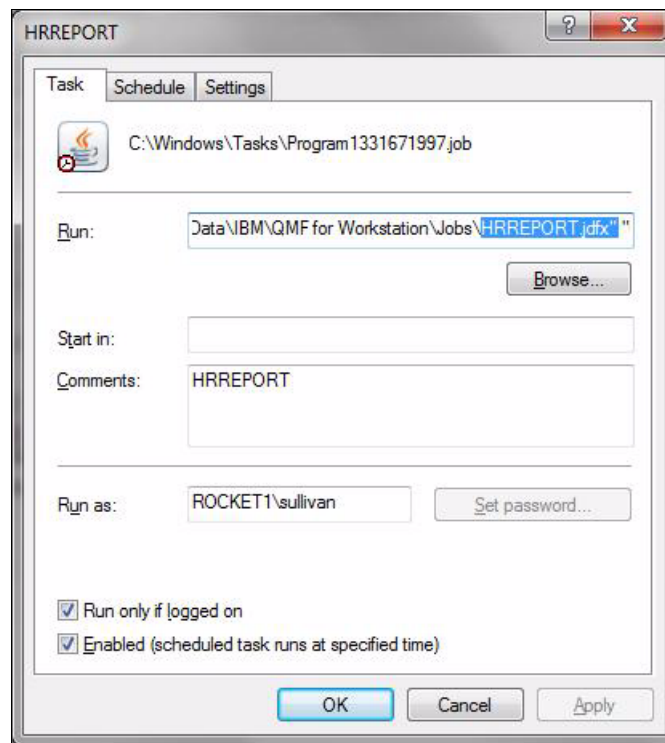


Figure 12-11 The file and associated Run command are passed to the operating system scheduler

Here in the Windows scheduler, you can see the encrypted file and the command-line command that will be used to run the job. The command is valid on this machine (it has some paths to some of the driver files embedded in it) and can be copied from here and used outside of the scheduler if needed.



Analysis and forecasting functions

The competitive market environment today is forcing many enterprise companies to seek a business analytics solution that allows them to easily look forward to predict future business trends. The QMF analysis and forecasting functions, coupled with its ability to directly make use of IBM SPSS® functions, provide a broad range of possibilities in this space.

This chapter explains the following functions:

- ▶ Analysis functions available within QMF for Workstation and WebSphere
- ▶ How to use QMF to forecast future outcomes
- ▶ How to embed SPSS functions in QMF dashboards

13.1 Using the QMF analytical functions

QMF includes a wide variety of analytical functions that can be added to visual reports and dashboards. Each of these functions is made available as an ad-hoc expression, allowing content developers to use them wherever required, from inclusion within charts to determining column values in tables and grid-based layouts.

For example, suppose that Spiffy Insurance Corporation monitors the number of customer inquiry tickets opened and closed each business day. On some days, more tickets are opened than closed, carrying workload over into the next day. On other days, more tickets are closed than opened, helping to clear the ticket backlog. Spiffy's IMS data contains the number of opened and closed tickets on a daily basis, but it does not report the cumulative tickets that remain open on a given day (comprising the day's open tickets plus whatever tickets remain open from prior days).

Customer service is critical in any industry, but in insurance it is extremely important to maintain a high degree of customer satisfaction. The QMF analytical functions provide a simple means of calculating the number of open tickets on a daily basis, something that would otherwise be somewhat difficult to compute using SQL. Using QMF, Spiffy's dashboard authors can develop the dashboard chart shown in Figure 13-1.

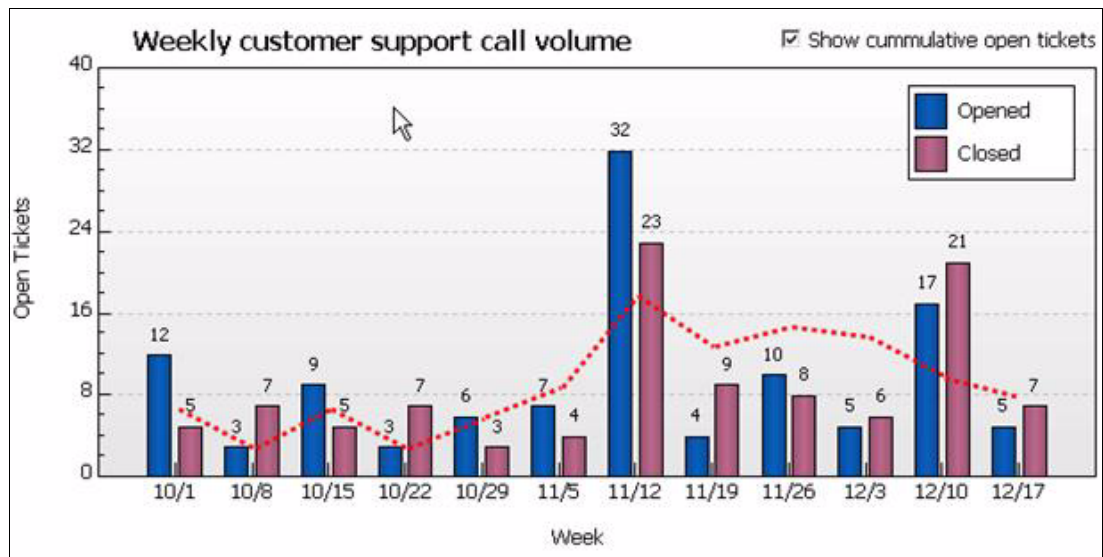
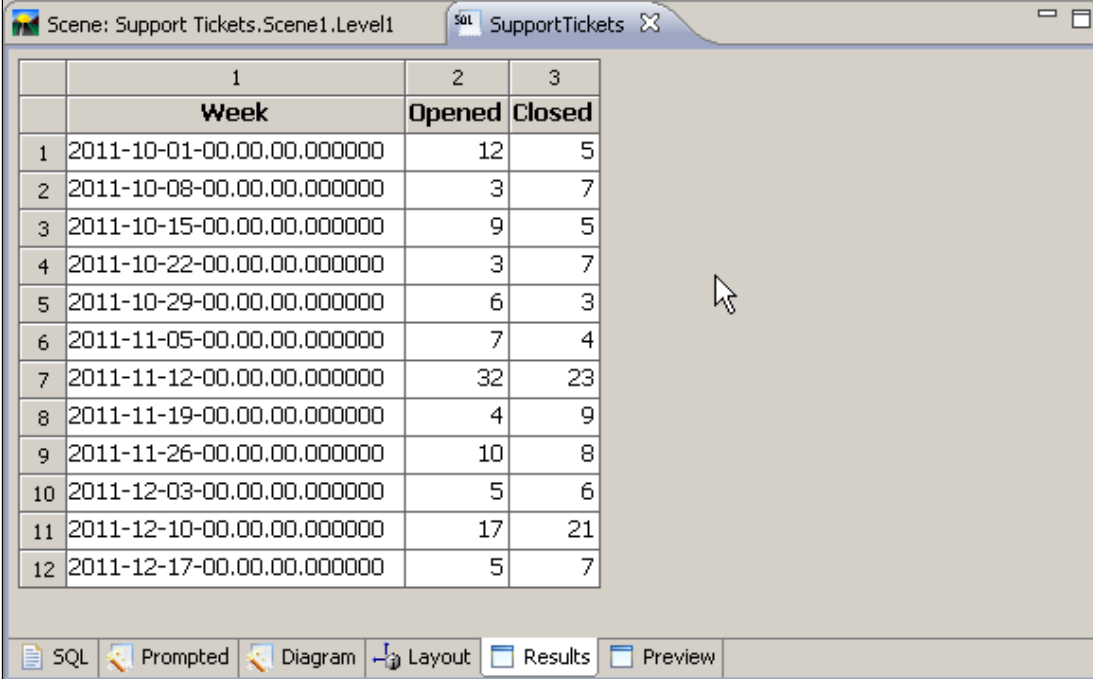


Figure 13-1 Weekly support ticket volume chart

The query behind the chart returns Week, Opened, and Closed columns, as shown in Figure 13-2.



	1	2	3
	Week	Opened	Closed
1	2011-10-01-00.00.00.000000	12	5
2	2011-10-08-00.00.00.000000	3	7
3	2011-10-15-00.00.00.000000	9	5
4	2011-10-22-00.00.00.000000	3	7
5	2011-10-29-00.00.00.000000	6	3
6	2011-11-05-00.00.00.000000	7	4
7	2011-11-12-00.00.00.000000	32	23
8	2011-11-19-00.00.00.000000	4	9
9	2011-11-26-00.00.00.000000	10	8
10	2011-12-03-00.00.00.000000	5	6
11	2011-12-10-00.00.00.000000	17	21
12	2011-12-17-00.00.00.000000	5	7

Figure 13-2 Query results that show numbers of customer support tickets by week

As shown in the results, seven tickets are carried forward in the first week (12 minus 5). In week 2, four more tickets were closed than were opened, dropping the number carried forward to 3, and so on. Although the result set does not contain the rolling tally, it can be easily added using the QMF analytical capabilities. In this case, the `accum()` function is used, one of over 150 analytical functions that can be added to augment result set data within reports and dashboards.

The `accum()` function performs a specified calculation over a specified number of rows, where the calculation can be any mathematical expression. The format for the function is as follows:

```
accum( start row, end row, calculation to perform )
```

In this case, Spiffy's dashboard author uses it as follows:

```
Tally = accum( 0, pointNumber(), Opened - Closed )
```

The foregoing expression accumulates the difference between the opened and closed column values, from the first row to the current row. The current row is determined by calling the `pointNumber()` function. The `accum()` function can be used to perform many different calculations. For example, a 3-point moving average of opened tickets can be returned using the following calculation:

```
MA = accum( pointNumber() - 2, pointNumber(), Opened / 3 )
```

After you understand the syntax and parameters, you can develop some very useful and powerful analyses. The QMF dashboard and report designers allow these functions to be used within any data-driven canvas object. In Spiffy's case, a connecting line is superimposed over the column chart values to provide the cumulative ticket totals, as shown in Figure 13-3. It is done by adding an alignment panel object to the chart to position the connector. The alignment panel's location is then set to the center of the bar slot (0) on the x-axis and the value of the `accum()` function in the y direction, as shown in Figure 13-3.

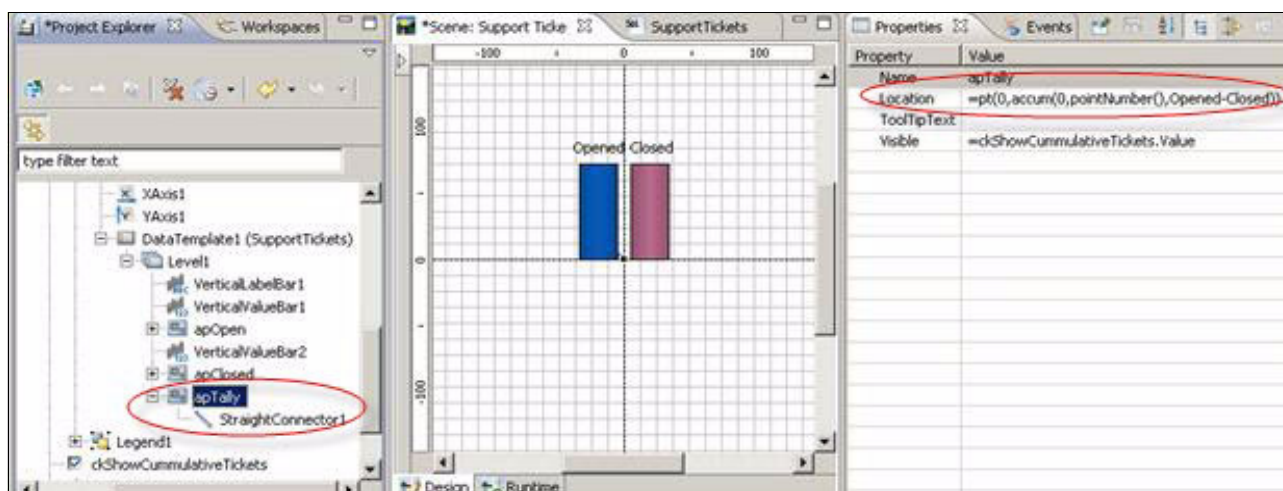


Figure 13-3 Adding the cumulative tally to the column chart

This flexible approach can be used to design custom presentation layouts that provide output based on a broad variety of analytical functions. As outlined in Figure 13-2 on page 279, content developers have the ability to define exactly what a given chart will present on a per-row basis (in this case, two bars and a connector line).

Any chart type can be quickly assembled using the QMF graphical building blocks, providing great flexibility. For example, using the QMF statistical functions, a floating quartile graph can be assembled and added to Spiffy's dashboards and reports to show the range of ages across the company's customer base on the basis of product category. To do so, Spiffy's content developers follow the procedure outlined next:

1. Add a column chart and set the bar value to the maximum age. Set the bar's reference value to the minimum age. It "floats" the bar between two values.
2. Add a solid line to the column chart layout and set it to the mean of the age values.
3. Add two dotted lines to the column chart layout and set them to the 25th and 75th percentile points for the age values.

Using this technique, Spiffy's content authors are able to develop the chart depicted in Figure 13-4.

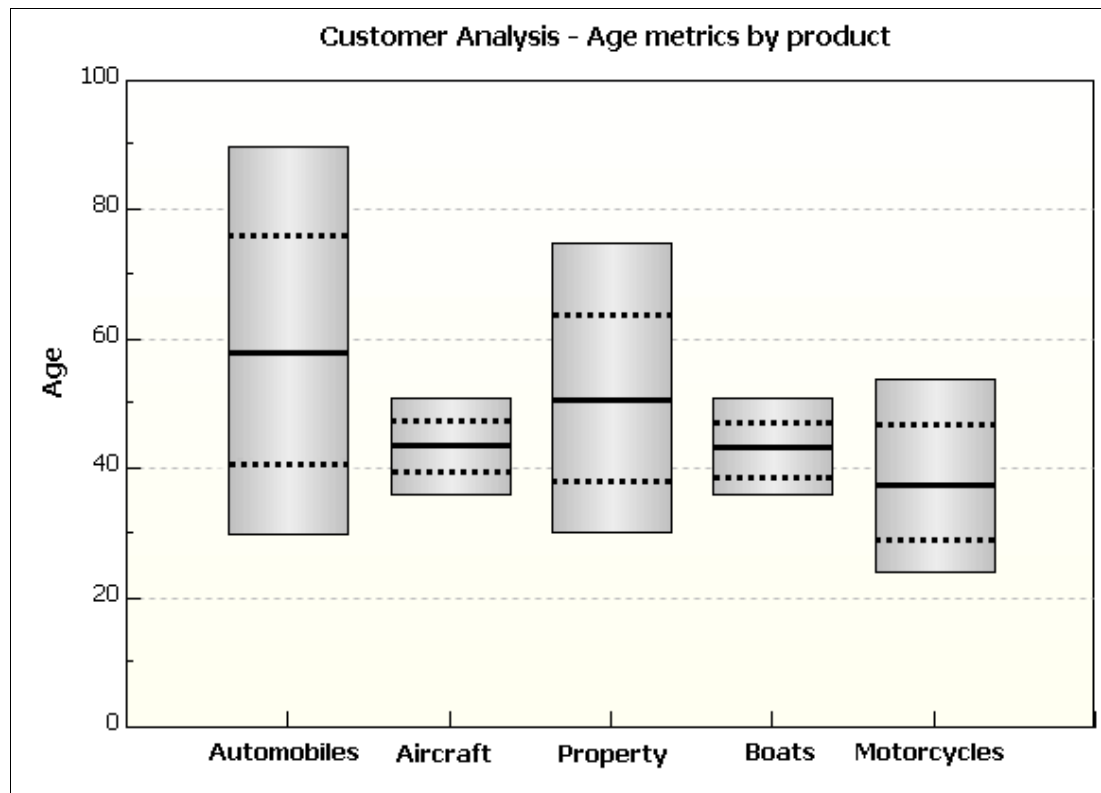


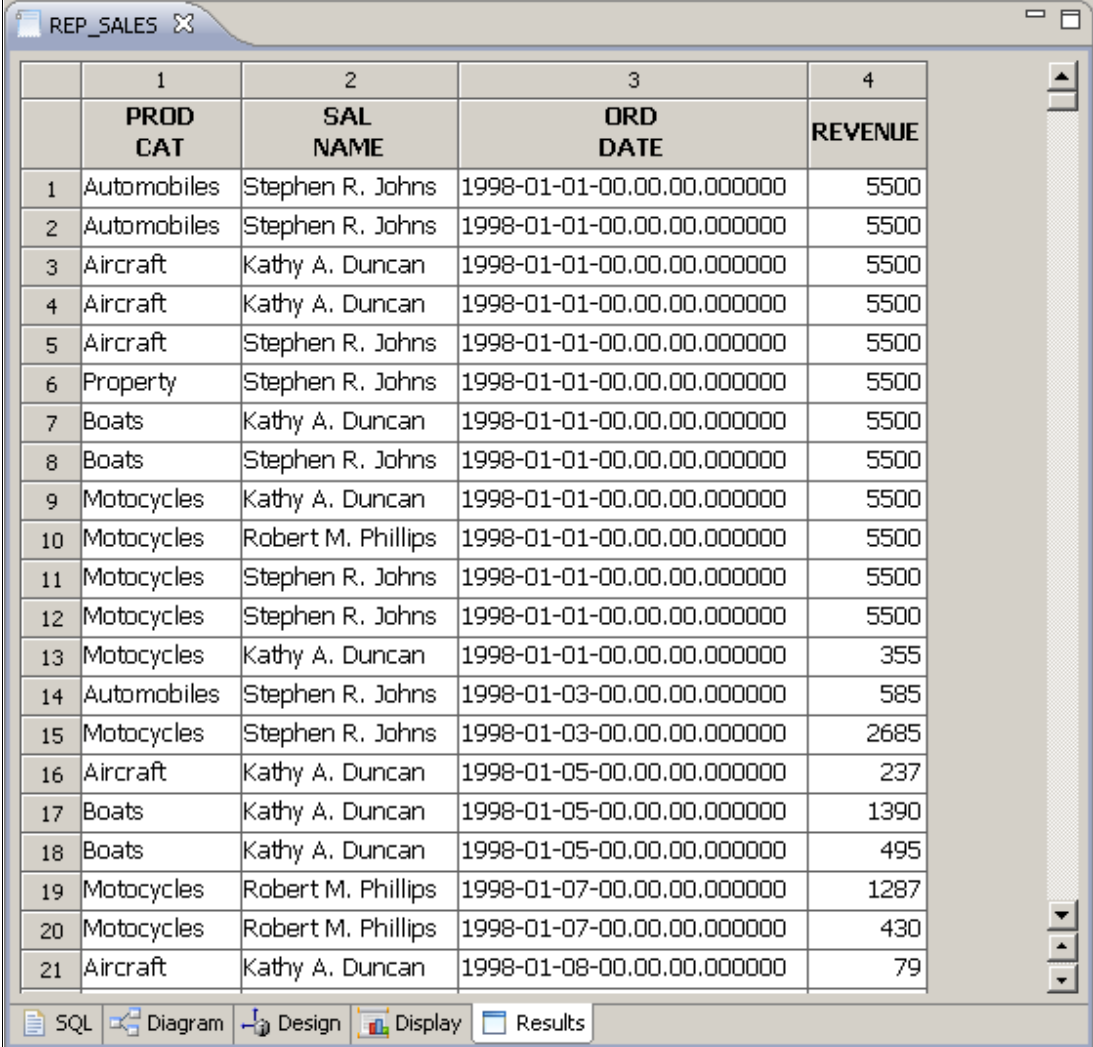
Figure 13-4 Custom QMF floating quartile graph

Appendix A, “Analytical functions available in the Expression Designer” on page 383 shows the broad range of QMF functions available and provides a brief description of each.

13.2 Forecasting outcomes

QMF for Workstation and WebSphere include predictive analytics functions that allow content developers to augment timeline data with forward-looking values. For example, suppose that Spiffy's sales managers are interested in predicting sales over the next three months by analyzing the last 12 months of data from each sales representative. From experience, sales managers know that performance in the past year is a good prediction of future sales, especially when looking out over a relatively near period of time.

The user starts by querying the sales figures, by rep, over the past 12 months. The query results are shown in Figure 13-5.



The screenshot shows a window titled 'REP_SALES' containing a table with the following data:

	1	2	3	4
	PROD CAT	SAL NAME	ORD DATE	REVENUE
1	Automobiles	Stephen R. Johns	1998-01-01-00.00.00.000000	5500
2	Automobiles	Stephen R. Johns	1998-01-01-00.00.00.000000	5500
3	Aircraft	Kathy A. Duncan	1998-01-01-00.00.00.000000	5500
4	Aircraft	Kathy A. Duncan	1998-01-01-00.00.00.000000	5500
5	Aircraft	Stephen R. Johns	1998-01-01-00.00.00.000000	5500
6	Property	Stephen R. Johns	1998-01-01-00.00.00.000000	5500
7	Boats	Kathy A. Duncan	1998-01-01-00.00.00.000000	5500
8	Boats	Stephen R. Johns	1998-01-01-00.00.00.000000	5500
9	Motocycles	Kathy A. Duncan	1998-01-01-00.00.00.000000	5500
10	Motocycles	Robert M. Phillips	1998-01-01-00.00.00.000000	5500
11	Motocycles	Stephen R. Johns	1998-01-01-00.00.00.000000	5500
12	Motocycles	Stephen R. Johns	1998-01-01-00.00.00.000000	5500
13	Motocycles	Kathy A. Duncan	1998-01-01-00.00.00.000000	355
14	Automobiles	Stephen R. Johns	1998-01-03-00.00.00.000000	585
15	Motocycles	Stephen R. Johns	1998-01-03-00.00.00.000000	2685
16	Aircraft	Kathy A. Duncan	1998-01-05-00.00.00.000000	237
17	Boats	Kathy A. Duncan	1998-01-05-00.00.00.000000	1390
18	Boats	Kathy A. Duncan	1998-01-05-00.00.00.000000	495
19	Motocycles	Robert M. Phillips	1998-01-07-00.00.00.000000	1287
20	Motocycles	Robert M. Phillips	1998-01-07-00.00.00.000000	430
21	Aircraft	Kathy A. Duncan	1998-01-08-00.00.00.000000	79

At the bottom of the window is a toolbar with buttons for SQL, Diagram, Design, Display, and Results.

Figure 13-5 Field sales representative performance by month

The user can invoke predictive analytics against any open query result set by selecting the “Transfer To” button on the application toolbar, as shown in Figure 13-6.

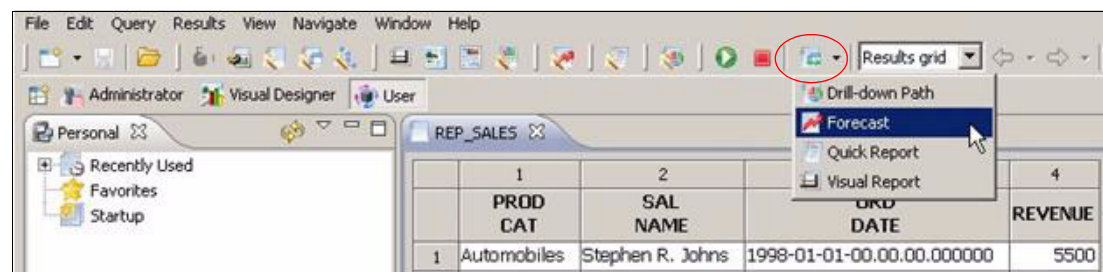


Figure 13-6 Transferring data to the QMF predictive analytics engine

This action creates a new forecast, as shown in Figure 13-7. Notice that the forecast automatically detects the data column and reporting period (monthly, in this case). The user has elected to forecast the next three sales periods and has also selected forecasts on a per-product basis; this was done by moving the product category (PROD_CAT) column over to the right in the group hierarchy section.

The screenshot shows a software window titled "REP_SALES" with a sub-tab "*Forecast1". The window is divided into three main sections:

- Data Source:**
 - ☒ **Embedded**: Includes buttons for "Import", "New", and "Edit".
 - ☐ **Linked**: Includes a "Path:" text field and a browse button "...".
- Date Parameters:**
 - Date column:** A dropdown menu showing "ORD_DATE".
 - From:** A date field showing "1/ 1/1998".
 - To:** A date field showing "12/23/1998".
 - Period:** A dropdown menu showing "Month".
 - Number of future periods:** A numeric field showing "3".
- Grouping Hierarchy:**
 - Available columns:** A list box containing "SAL_NAME".
 - Selected columns:** A list box containing "PROD_CAT".
 - Between the two list boxes are four arrow buttons: a single right arrow, a double right arrow, a single left arrow, and a double left arrow.

At the bottom of the window, there are two tabs: "Group" (selected) and "Model".

Figure 13-7 Reviewing the forecast options

[illegible]

284 Complete Analytics with IBM DB2 Query Management Facility

When executed, the forecast examines patterns in the historical data and computes the likely sales values for the next three months, on a per-sales rep and per-product basis. The result of the analysis is shown in Figure 13-9. Notice that the last three rows of the grid are the forward predictions. The grid and chart show the performance of the three predictive algorithms proactively selected by the user.

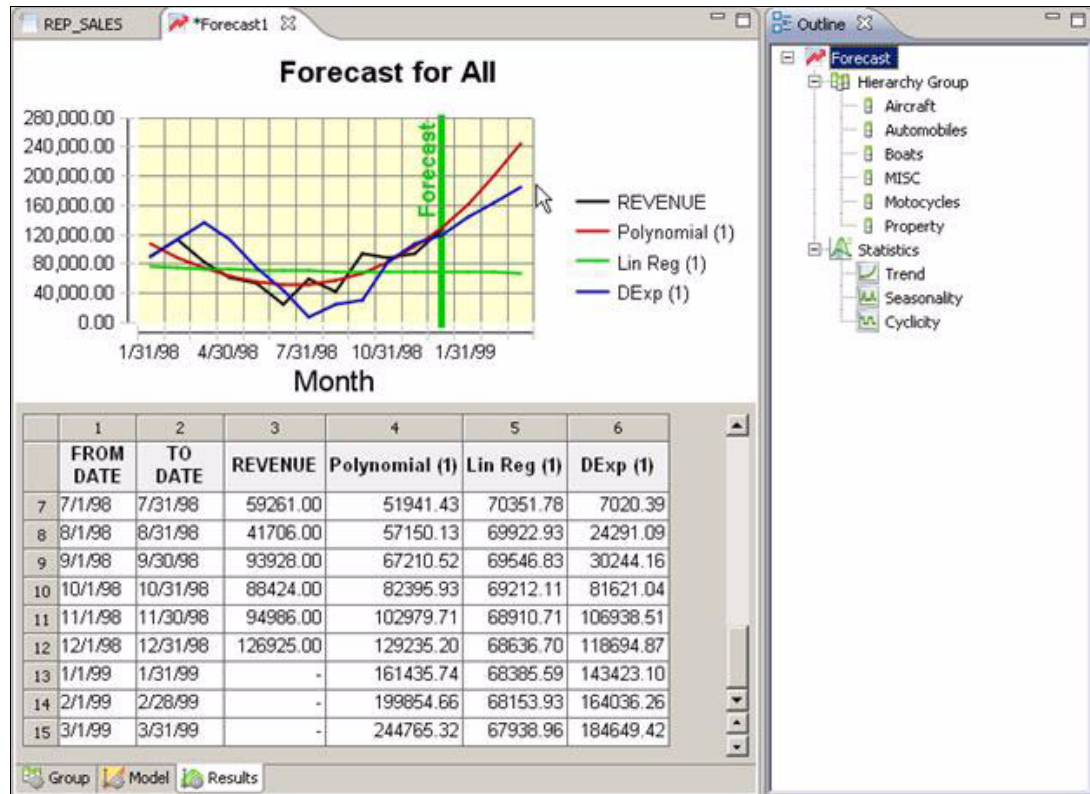


Figure 13-9 Reviewing historical and predicted data values generated by QMF

When defined, the forecast acts much the same as a query, reading the historical data from Spiffy's data sources and adding predictive values for the upcoming three months, based on measured trends in the historical data. This data set can then be incorporated directly within the QMF page-based reports or interactive dashboard solutions, perhaps populating data in a sales revenue chart or table.

The QMF analytical functions, coupled with its flexible dashboard and report design environment, allow content developers to assemble customized data visualizations that offer a broad range of possibilities. These analytical capabilities are focused on performing calculations on result sets that are derived from one or more data sources in the enterprise. QMF can also interoperate with the comprehensive and highly specialized analytical capabilities offered by the IBM SPSS software suite.

13.3 Embedding SPSS functions in QMF dashboards

SPSS provides the ability to directly access DB2 to model various scenarios to reveal specific business insights. SPSS is available on System z running under Linux.

Ideally, Spiffy wants to be able to produce an accurate model that the company can use to analyze customer purchasing habits to tailor products, services, and campaigns to increase sales. Spiffy's SPSS modeling process is outlined in Figure 13-10.

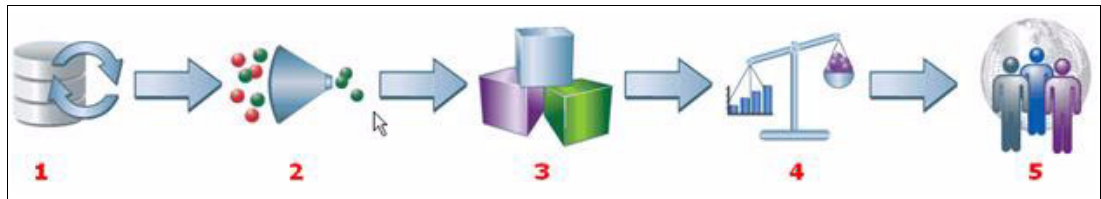


Figure 13-10 Spiffy's SPSS modeling process

- Step 1: Select the data to be processed. In Spiffy's case, it is information held in DB2, IMS, and SQL Server.
- Step 2: Select customers for analysis. For example, Spiffy might choose to exclude customers with bad credit or those who otherwise should not be considered for campaigns and offers.
- Step 3: Define campaigns and offers, specifying how customers are to be allocated.
- Step 4: Specify how revenue, cost, and prioritization values are combined to balance corporate objectives and results.
- Step 5: Review the modeling/analysis configuration and prepare it for deployment.

After these steps have been completed, the resulting model can be applied to incoming data. SPSS functionality is delivered by a web-based interface that provides direct interoperability with QMF for Workstation and QMF for WebSphere.

Spiffy uses the QMF embedded content dashboard object to directly embed SPSS functionality within QMF dashboards, as shown in Figure 13-11. The embedded content object allows dashboard developers to incorporate an external web page directly onto a dashboard canvas. Spiffy merely places the object on the canvas and sets the URL to that of the SPSS server in the company's environment.

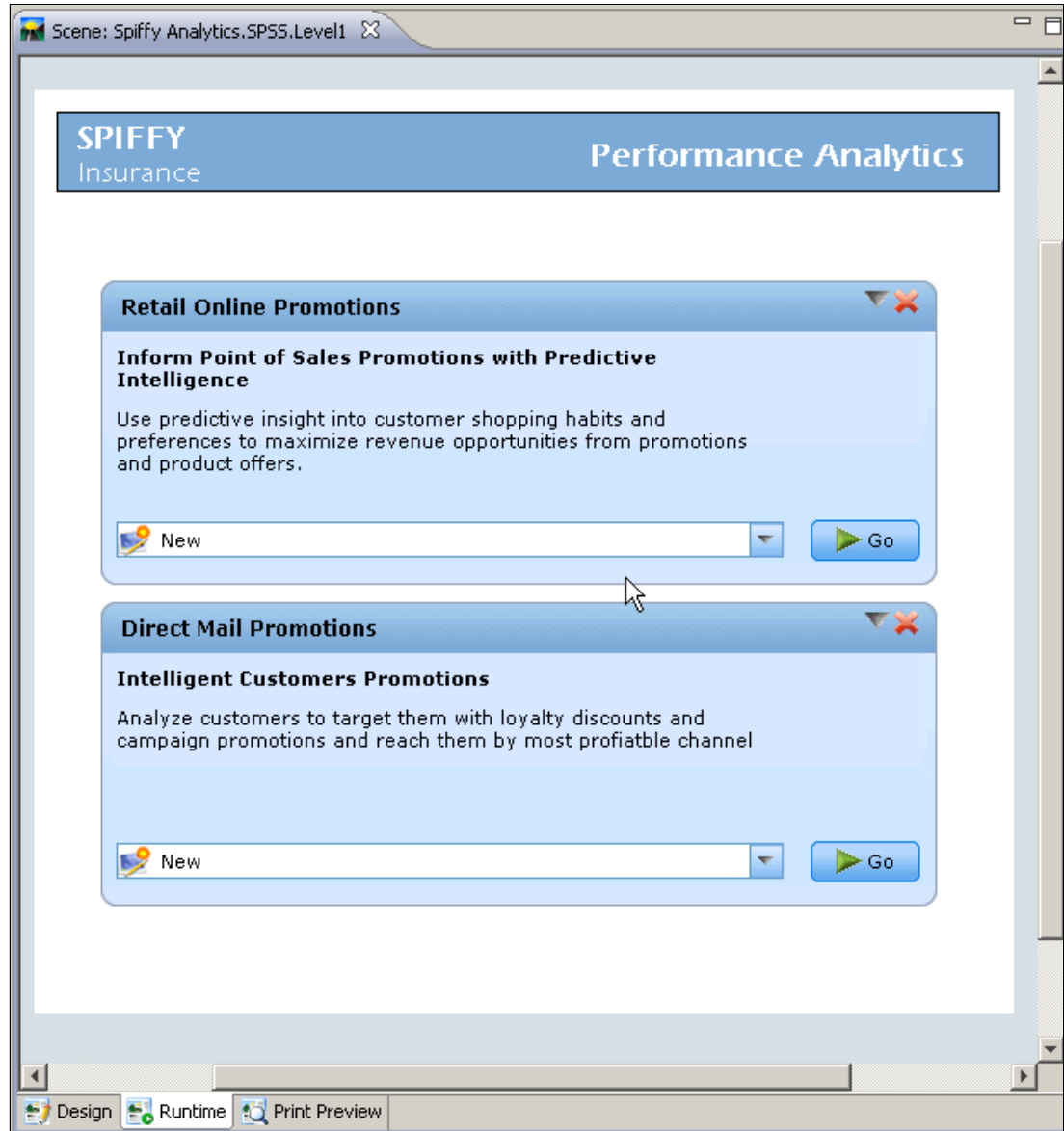


Figure 13-11 SPSS components running within a QMF dashboard

By integrating SPSS web components within QMF, Spiffy is able to deploy solutions that blend real-time business performance reporting with advanced predictive analytics and business data modeling. When clicked, the dashboard content launches the respective data modeling panels in an independent browser window, as shown in Figure 13-12.

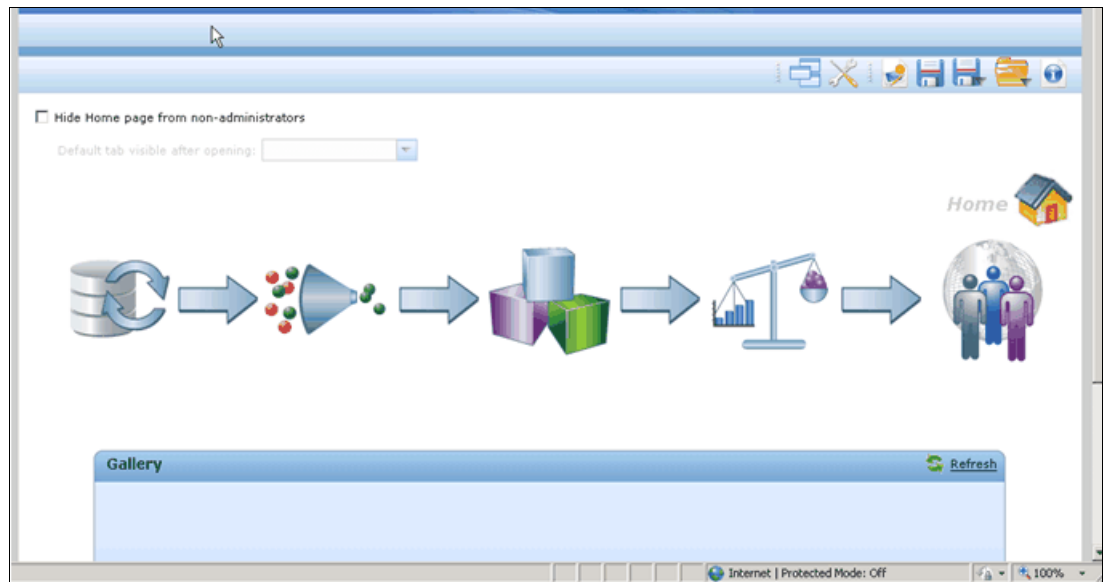


Figure 13-12 SPSS Decision Management window, launched from a QMF dashboard

In Chapter 14, “Putting it all together: Developing dashboards” on page 289, we examine step-by-step how to create dashboards such as the one used to launch the SPSS content shown before.



Putting it all together: Developing dashboards

A dashboard is a type of visual project that can be used to present interactive or persistent data to a wide range of users. Like applications, dashboards are usually designed by content developers with specific end user needs for data access and presentation in mind. QMF dashboards can simultaneously draw data from multiple, heterogeneous data sources and present this data using a wide variety of charts, graphs, maps, and user interface widgets.

This chapter walks you through the steps required to create a dashboard and provides two in-depth scenarios that allow you to get a closer look at some key dashboard elements in action.

14.1 Basic elements of a QMF dashboard

A QMF dashboard is made up of four elements:

- **Connections:**

Dashboards are intended to be shared across the enterprise. To facilitate sharing and distribution, QMF for Workstation does not tie the queries that are included in a dashboard to a specific data source. Instead, it ties the query to a connection information alias that you define before you begin creating the dashboard.

- **Queries:**

The data that appears in the dashboard is driven by underlying queries that are defined and tied to specific components of the dashboard.

- **Scenes:**

A scene is the main element of a dashboard. Each dashboard is made up of one or more scenes. Each scene of a dashboard graphically represents information retrieved from a database. A scene can also include other graphic elements, such as static text and images, and navigational tools such as embedded scenes and jumps to other scenes.

- **Globals:**

Resources that can be used across the entire dashboard project, in all dashboard scenes, are called globals.

The Project Explorer tree in the Visual Designer perspective, showing the four main elements of a dashboard, can be seen at the left of Figure 14-1.

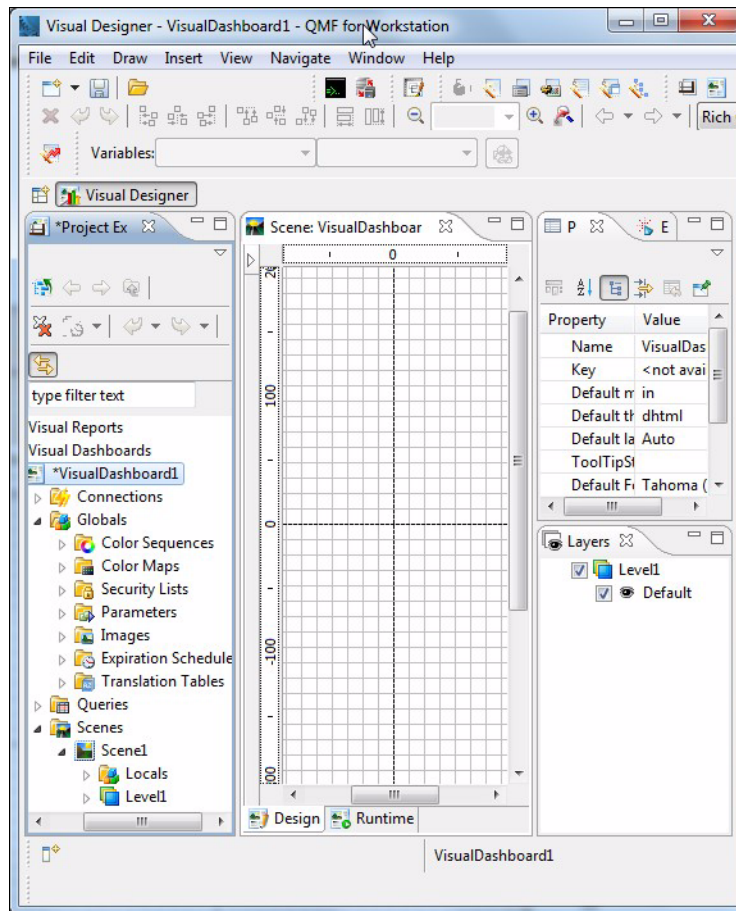


Figure 14-1 Project Explorer tree, Visual Designer perspective, with four main elements of a dashboard

14.2 How to create a dashboard

These steps outline the main tasks that you need to perform to create a dashboard in QMF:

1. Plan the dashboard.

Up-front planning facilitates developing a dashboard. As a content developer, you need to decide what to present, how much initial detail to provide, how to access more detailed information, and how to navigate to and from the scenes that make up your dashboard.

A storyboard is a plan for what your dashboard will include. You use a storyboard to design the basic features, functionality and presentation flow of the scenes and objects that will be included in your dashboard. A storyboard for a dashboard must specify the following items:

- Data for each scene
- Scenes, drawn roughly
- Layouts for each scene
- Sequence of presentation
- Points where users can access further information
- Actions that result from user events
- Jumps between scenes

There are various tools that you can use to create storyboards. You can sketch one using pencil and paper. You can use a presentation tool, such as Microsoft PowerPoint, to create sample layouts, with supporting detail on notes pages. You can also put together web pages to show actual jumps and navigation flow. The following list of general topics is a guide to some of the design elements you need to include in your plan:

- Displaying query results data:

You use layout objects or the List and Combo controls to display query results data. Explore the different display options and decide which ones best suit the kind of analysis you want to show.

- Capturing user input:

You can capture user preferences with standard user interface controls. Look at the Controls palette on the Palette view to explore your options.

- Passing information:

Using parameters, you can pass information acquired from user actions to affect the display of a dashboard, the contents of a scene, or the execution of a query.

- Navigating around dashboards:

You can use navigation features to allow users to move around dashboards to get to new information.

- Locating your data:

Your storyboard should specify the database and include a list of the tables that you plan to use. Writing this list will help you organize your work. You can then spend a session creating all the queries you need prior to creating your first scene. Alternatively, you might work with others who extract the data that you need from the database tables and consolidate it in summary tables.

2. Use the Create New Visual Dashboard wizard to create a new, empty dashboard.

3. Open the **Visual Designer** perspective. The Visual Designer perspective presents an editor window and several views and menus that you will use when you work with dashboards. You can open the Visual Designer perspective by either clicking the Open perspective icon in the toolbar (shown in Chapter 6, “Configuring access to data sources and populating user workspaces” on page 89, where perspectives were initially explained) or selecting **Window** → **Open Perspective** → **Visual Designer** from the menu.

Each dashboard that you create is listed under the Dashboards node in the Project Explorer tree. In the tree, each dashboard will have folders for connections, globals, queries, and scenes.

4. Create connections to the data sources that will be used to populate the various areas of the dashboard.

The connection information includes a name for the connection and the name of the data source to which it will point. Before a connection to a data source can be created, you must:

- Specify the location of the JDBC driver files for each type of database that will be used as a data source in the dashboard.
- Add the data source so that it is defined and available for you to configure a connection to it.

These tasks are covered in Chapter 6, “Configuring access to data sources and populating user workspaces” on page 89.

5. Specify the first query that will supply data for the dashboard.

You can specify an existing query or create a new query. When you specify the query, QMF prompts you to associate the query with a connection information alias. The connection information alias identifies the data source against which the query runs.

6. Design the first scene for the dashboard.

A scene is the container (similar to a presentation slide) that will hold all the elements used to display your data. A new, empty dashboard has one default scene. You can choose to have only one scene in your dashboard or you can add additional scenes. The examples in this chapter show dashboards with multiple scenes so that you can see the interaction between them.

At the bottom of the scene canvas are two tabs: Design and Runtime. The design tab allows you to insert objects into the scene, while the Runtime tab shows what the dashboard user will see, providing you with a view of how the dashboard is progressing as you go through the design process.

7. Insert objects into the scene.

The objects that you can insert are displayed in the Palette view and are organized on individual palettes based on object type. From the Palette view, select the objects that you want to insert into your dashboard scene. Figure 14-2 shows how to display the Palette view.

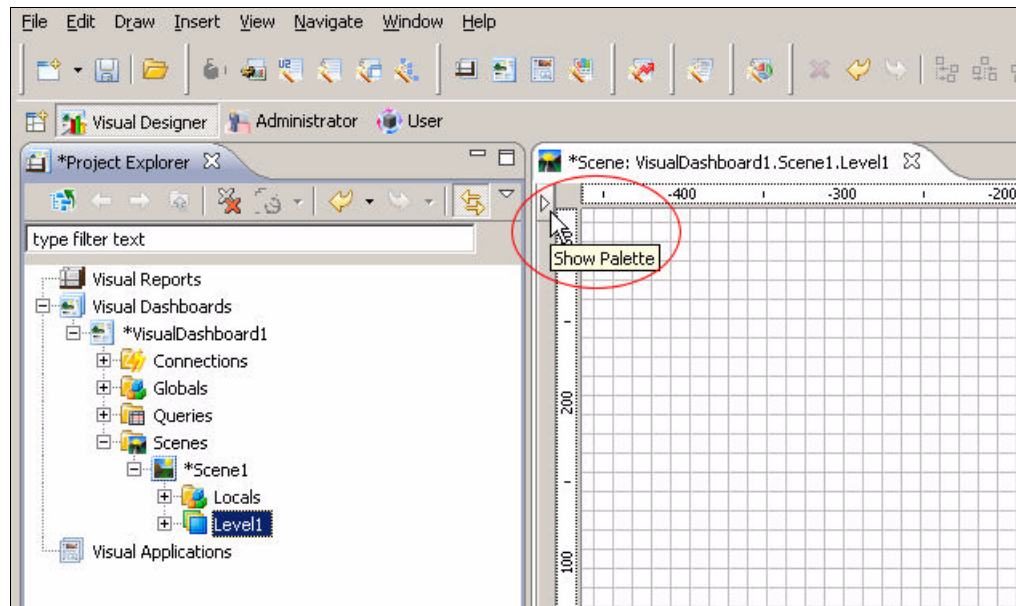


Figure 14-2 Displaying the Palette view by clicking the right arrow at the top left of the scene canvas

These three types of palette objects are most commonly used:

- Primitives:

Primitive objects are graphic objects such as text, lines, shapes, alignment panels, and pictures.

- Controls:

Control objects are graphical objects that are used to display information or accept user input, such as text boxes, list boxes, buttons, and date/time selectors.

Each type of control has its own properties that make it suitable for a particular purpose. Using controls, you can request input from a user and, based on the response to the control, you can trigger subsequent events.

- Layouts:

The Layouts palette within the Palette view contains graphical objects that you can use to display the data that has been obtained as a result of a query. Layout objects include a wide assortment of charts, graphs, maps, tables, and grids.

When you insert a layout object, the system prompts you to associate the object with a query that is contained in the Queries folder. You can add queries to your Queries folder at any time during the process of creating your dashboard.

Layout objects can present data in many ways. For most layout objects, you display the results of multiple queries in a single layout. For example, you might create a single XY chart that displays sales figures derived from one query and spending figures derived from another query.

You can add multiple layout objects to each scene in your dashboard. When you place layout objects, you can pass query result information from a higher-level layout object to a lower-level layout object. Because you have the ability to pass this information, you can use the placed layout objects to display more detailed information that relates to a specific data value.

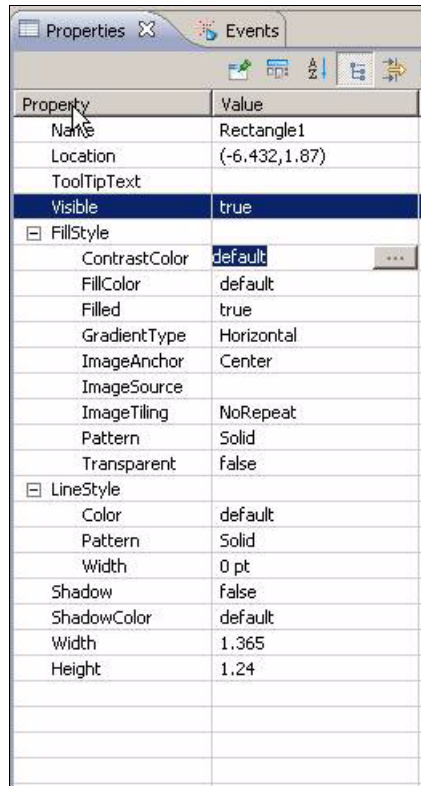
Figure 14-3 shows the Palette view:



Figure 14-3 The Palette view, which displays objects that can be placed in a dashboard

8. Modify each object's properties in the Properties view.

Each object added to the dashboard has its own properties. The available properties that can be assigned to an object vary depending on object type. For example, Example 14-4 shows properties available for a rectangle object.



Property	Value
Name	Rectangle1
Location	(-6.432,1.87)
ToolTipText	
Visible	true
<input checked="" type="checkbox"/> FillStyle	
ContrastColor	default
FillColor	default
Filled	true
GradientType	Horizontal
ImageAnchor	Center
ImageSource	
ImageTiling	NoRepeat
Pattern	Solid
Transparent	false
<input checked="" type="checkbox"/> LineStyle	
Color	default
Pattern	Solid
Width	0 pt
Shadow	false
ShadowColor	default
Width	1.365
Height	1.24

Figure 14-4 Object properties for a rectangle

By modifying an object's properties, you set the values that determine how the object looks and behaves. Properties can be set in design mode, and also set at runtime, in response to dashboard events.

Properties can be set by simple conditions, or by more complex criteria that can be created using the array of built in functions available in the Expression Designer, which is shown in Figure 14-5.

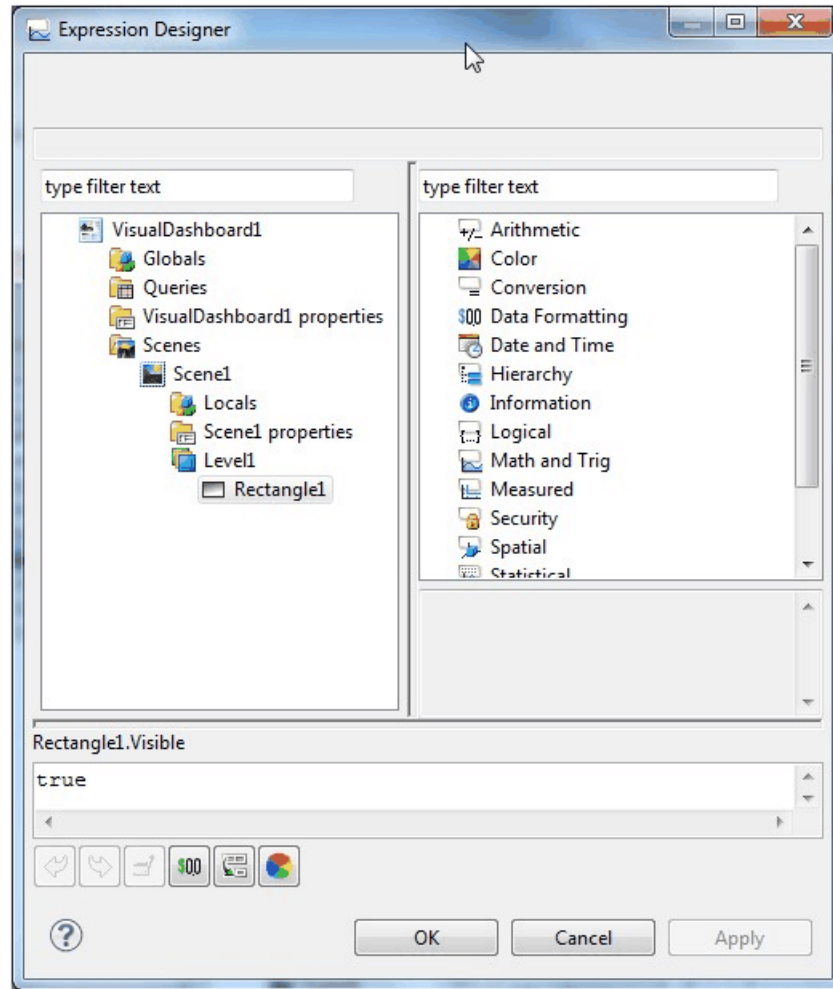


Figure 14-5 Expression Designer

9. Use the Events view to assign the different navigation options.

Events are actions that are performed by the dashboard user. As a user navigates the dashboard, the user's choices and selections trigger predefined events, such as jumps between dashboard scenes, execution of SQL statements, or the launching of other applications. For example, using a button's click event, you could set a property (or multiple properties) for other controls on the dashboard. Figure 14-6 shows a button event currently in design.

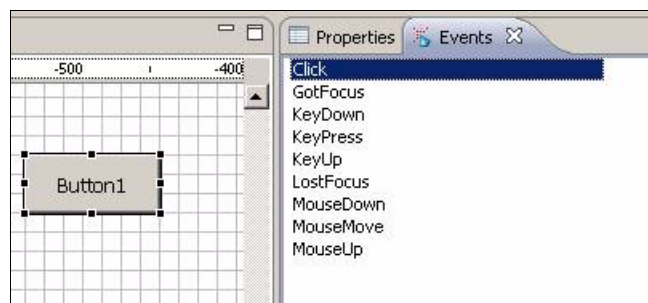


Figure 14-6 Button event

Figure 14-7 shows actions that can be associated with an event. Notice that multiple actions can be associated with the same event.

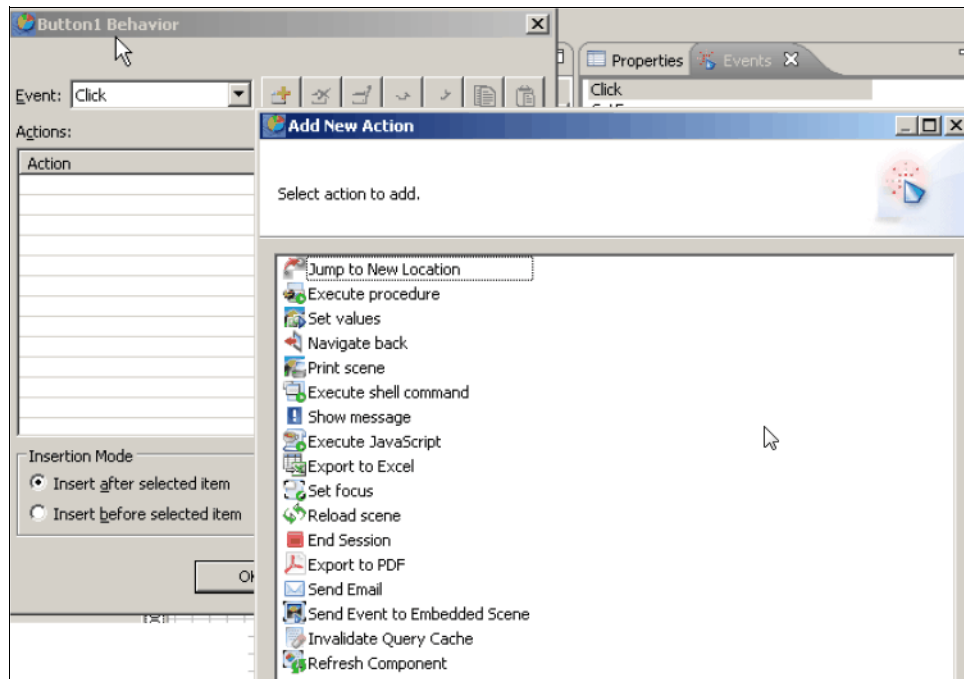


Figure 14-7 Actions that can be associated with an event

By assigning navigation options to scenes and objects, you set a path through the dashboard. One form of navigation is to define dependencies between dashboard control objects (such as combo or list box objects) and data-driven layout objects (such as a grid or bar chart object) using the Connectivity tool from the Palette view.

10. Run the dashboard to test its functions.

You run the dashboard by clicking the **Runtime** tab. By performing this step, you can review how the results will display at run time. Think of this step as a test-run of your dashboard that allows you to improve upon the design before saving and deploying the dashboard.

As you design your dashboard, adding multiple objects, and perhaps multiple scenes, you will likely use the Design and Runtime tabs frequently to toggle between designing elements and checking to see how they will appear to the dashboard user.

11. Save and deploy the finished dashboard.

When you have finished designing and testing the dashboard, you can save it and copy it to one or more workspaces for distribution to other users. To save a dashboard, click it in the Project Explorer tree and select **File** → **Save** from the menu.

The rest of this chapter provides two detailed scenarios that show dashboard creation in action following the foregoing process:

- ▶ The first scenario is for a dashboard that has three scenes and draws customer-related data from both analytical and operational systems to present it to Spiffy Insurance field agents. Read through this scenario to gain a deeper understanding of the power of the QMF dashboarding functions and to understand the dashboard creation process within the context of a specific example.
- ▶ The second scenario is for a dashboard that has one scene that draws sales data from OLAP and relational data sources to present it to Spiffy market analysts. This scene provides even more detail than the first scenario and is ideal to follow as a live tutorial.

14.3 Scenario 1: Field agent dashboard (relational and hierarchical data) with three scenes (three data sources)

Like many organizations today, Spiffy Insurance has evolved a data infrastructure that includes several databases from different vendors. These databases are installed on a variety of platforms and, in the short term, the company has a business requirement for its field agents to be able to access data from all of these sources.

In the last topic, we covered the basic process for creating a dashboard. This topic shows how Spiffy's IT department would go through the dashboard creation process to create a dashboard customized for its field agents.

To provide the information its field agents require, Spiffy must create a dashboard to present the necessary information from three distinct data sources in a transparent manner:

- ▶ Customer information from the data warehouse, sourced in DB2 for z/OS
- ▶ Product information, sourced in Oracle
- ▶ Real-time customer activity, sourced in IMS

Each data source has its own unique characteristics and part to play in the composite dashboard, which will appear to field agents as an easy-to-use application. The dashboard must meet two main objectives for Spiffy's field agents:

- ▶ Provide insurance business overview information by geography:

Field agents need summaries of customers and products by geography. This information must be federated from both DB2 for z/OS and Oracle and presented in maps and charts by geography. Furthermore, the charts, maps, and other dashboard controls must be linked, such that drilling down into one area automatically updates the data shown in another.

- ▶ Show both historic and real-time purchasing activity for individual customers within selected geographies:

Field agents require customer detail from their analytical systems showing historic purchasing activity, as well as operational data from IMS showing the most recent 30 days of purchasing activity, for a specific customer. Although IMS data does get loaded into the warehouse driven by DB2 for z/OS, the lag time and level of detail are less than what the agents require and so real-time analytics are required for the dashboard.

These main dashboard components will be presented as three distinct scenes in the dashboard.

14.3.1 Creating a virtual data source

Chapter 7, “Defining virtual data sources to reduce complexity for business users” on page 133 covered in detail how to create a virtual data source, as well as the benefits that virtual data sources provide to end users, who then do not have to try to understand the complexities of the underlying database schemas in order to access the data they need.

With its customer information sourced in DB2 for z/OS and its product information sourced in Oracle, Spiffy needs to create a virtual data source for field agents that federates the data from these two sources behind the scenes and presents the information in the business overview component of the dashboard. To create this virtual data source, we follow these steps:

1. In the Administrator perspective, right-click **Virtual Data Sources** and select **New** → **Virtual Data Source**, as shown in Figure 14-8. For a review of perspectives, see Chapter 5, “Installing QMF for Workstation, QMF for WebSphere, and touring the interfaces” on page 65.

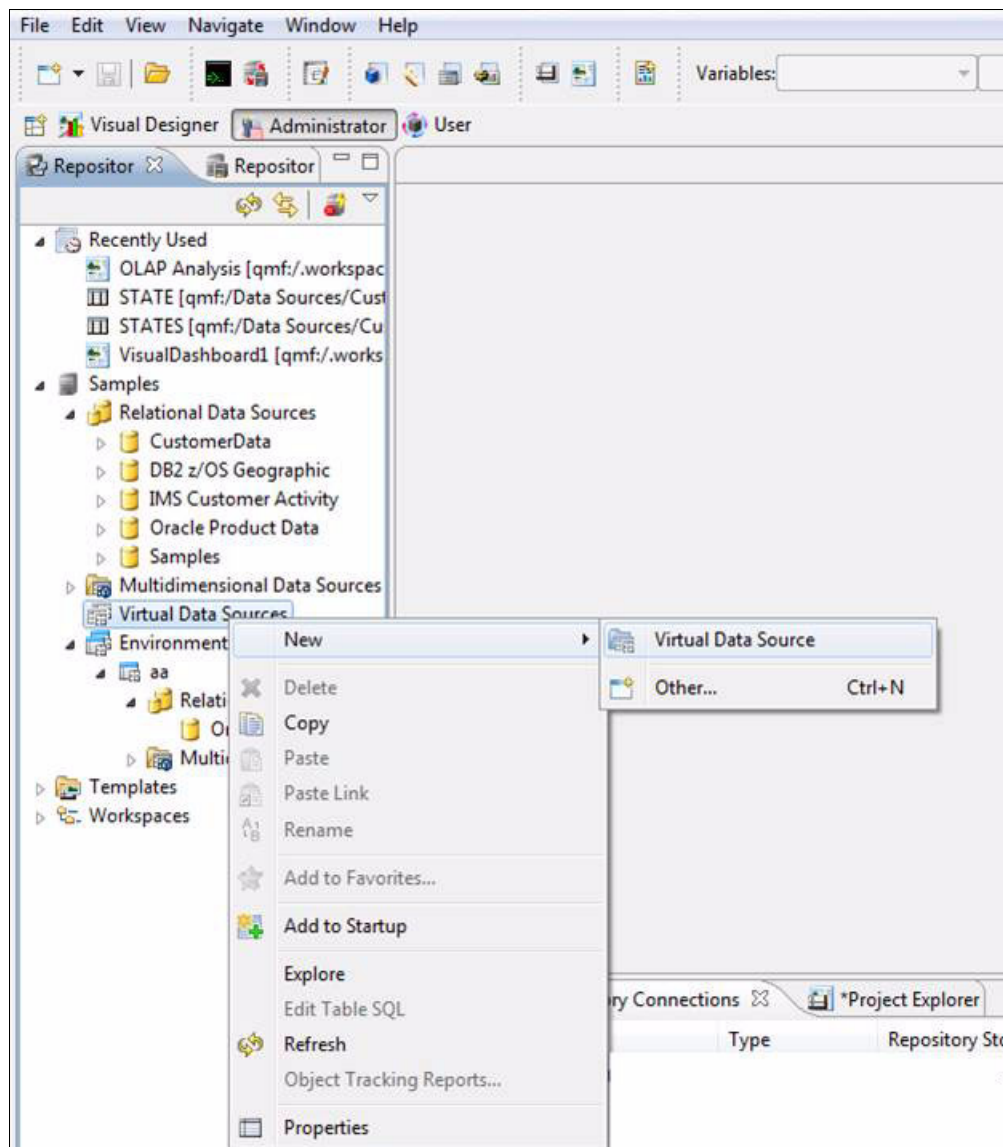


Figure 14-8 Creating a new virtual data source

2. Enter a meaningful name in the resulting dialog and click **Finish**.

For this exercise, we are going to use Business Overview as the name, because this federated data source will drive the business overview component of the field agent dashboard.

3. Select the tables to include in the virtual data source.

Because the data that will drive the business overview component of the dashboard resides in both Oracle and DB2 for z/OS, Spiffy's IT staff must select tables from both databases for the virtual data source. Customer summary tables selected from DB2 for z/OS are shown in Figure 14-9.

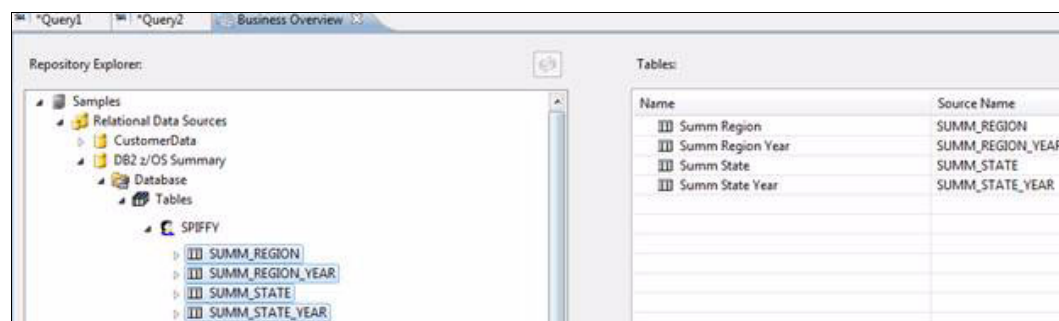


Figure 14-9 Adding DB2 for z/OS customer summary tables

From Oracle, Spiffy selects product information tables, as shown in Figure 14-10.

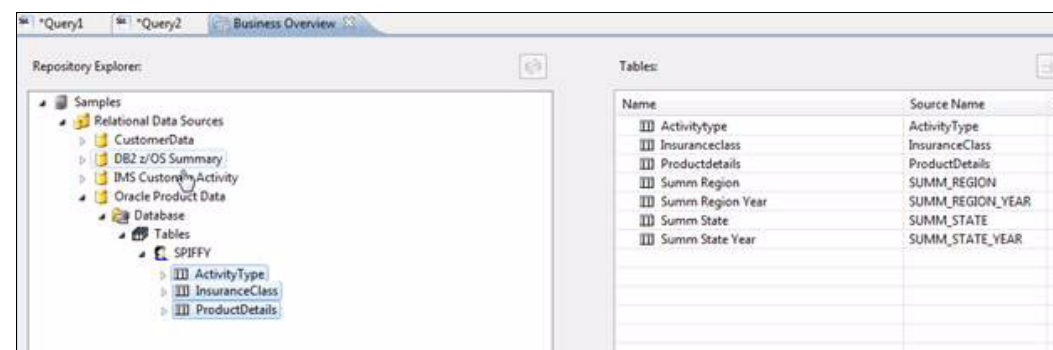


Figure 14-10 Adding Oracle product Information tables

QMF allows you to select specific columns for the virtual data source, should that be required, as well as rename the columns. Chapter 7, “Defining virtual data sources to reduce complexity for business users” on page 133 provides a comprehensive view of the powerful features of the QMF virtual data sources function if you want to explore them in more detail.

After creating the virtual data source that will populate the Business Overview scene of the dashboard, Spiffy is ready to begin creating the dashboard, as explained in the next topic.

14.3.2 Creating a new dashboard

To begin creating a dashboard:

1. Open the **Visual Designer** perspective and select **File** → **New** → **Visual Dashboard**, as shown in Figure 14-11. For more information about perspectives, see Chapter 5, “Installing QMF for Workstation, QMF for WebSphere, and touring the interfaces” on page 65.

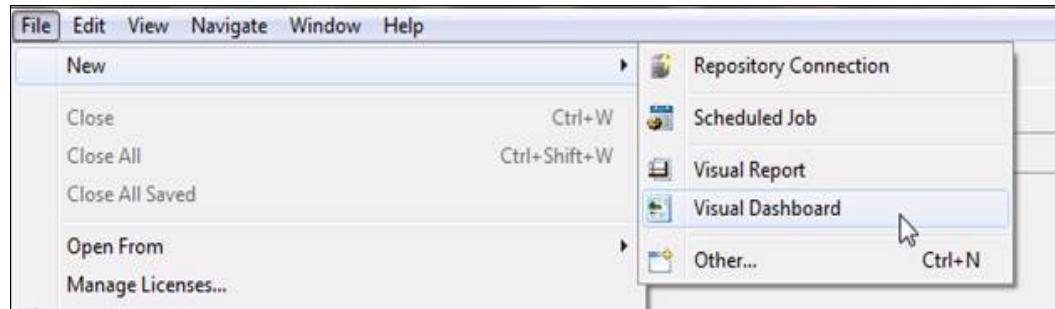


Figure 14-11 Creating a new, empty dashboard

The Create New Visual Dashboard wizard opens.

2. Type a unique name for your dashboard in the Dashboard name field.
3. Click **Finish**.

14.3.3 Creating data source connections

Before you begin designing scenes in a dashboard, you must set up connections to each data source that will be used to populate these scenes.

To specify connection information for a data source:

1. In the tree on the left side of the Project Explorer view, right-click **Connections** and select **Insert Connection** from the context menu.
2. From the list of available data sources, select the data source that will be associated with this connection information alias.

For this example, we select the **Business Overview** virtual data source defined in 14.3.1, “Creating a virtual data source” on page 300, as shown in Figure 14-12.

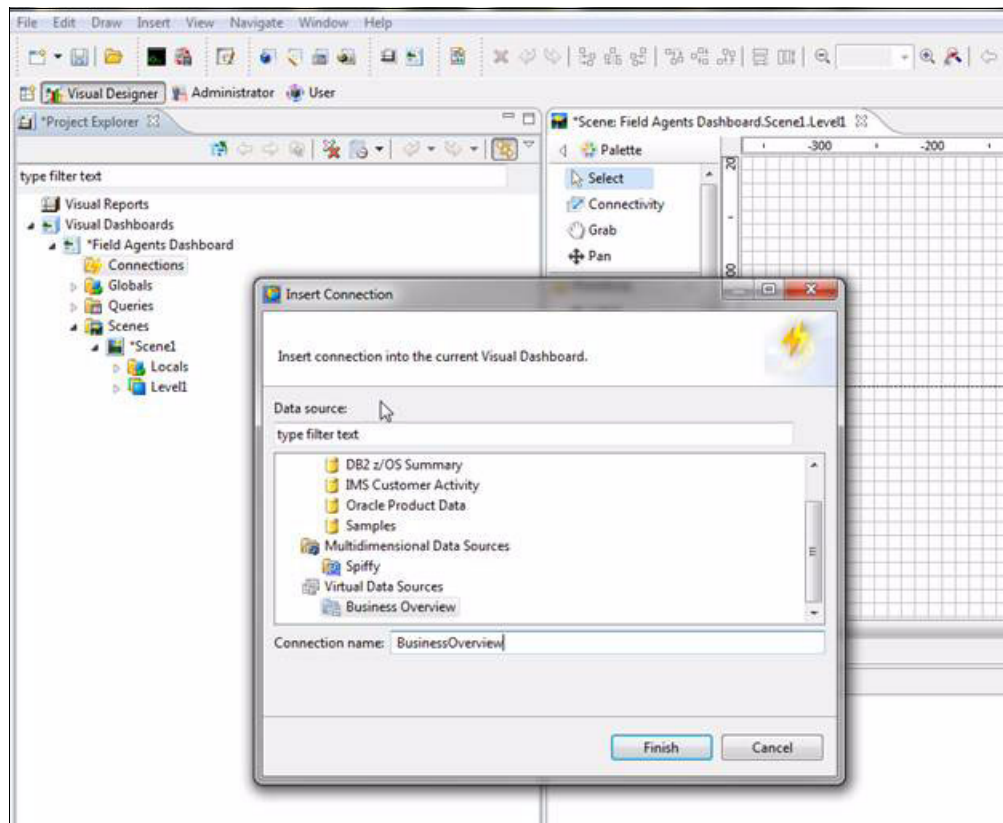


Figure 14-12 Defining a data source connection for the dashboard

3. Specify a unique name for this connection information in the Connection name field. For this example, we define BusinessOverview as the connection name.
4. Click **Finish**.

14.3.4 Specifying the first query that will supply data to the dashboard

The Business Overview scene in the field agent dashboard will require a map to be displayed. A new query is required to populate this map. Figure 14-13 shows the list of tables available in the Business Overview virtual data source.

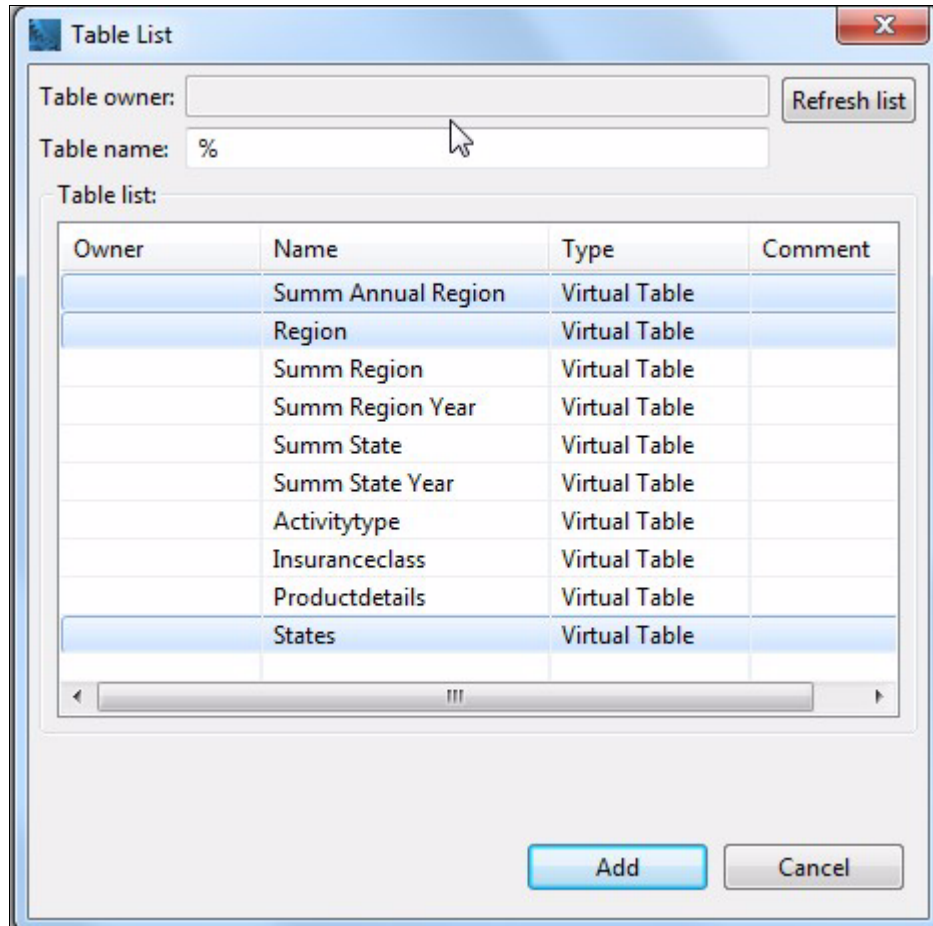


Figure 14-13 Tables available in Spiffy's Business Overview virtual data source

Three virtual tables are used to create the query required, which is called qryMap and is shown in the query diagram view in Figure 14-14. The "Summ Annual Region" virtual table holds summary data for multiple years, so we add a row condition to only show data for a specific year, whose value can then be dynamically set from the dashboard. For more information about creating queries with the query diagram view, see "Developing queries using the query diagram view" in Chapter 9, "Getting to the data you need: Query methods" on page 191.

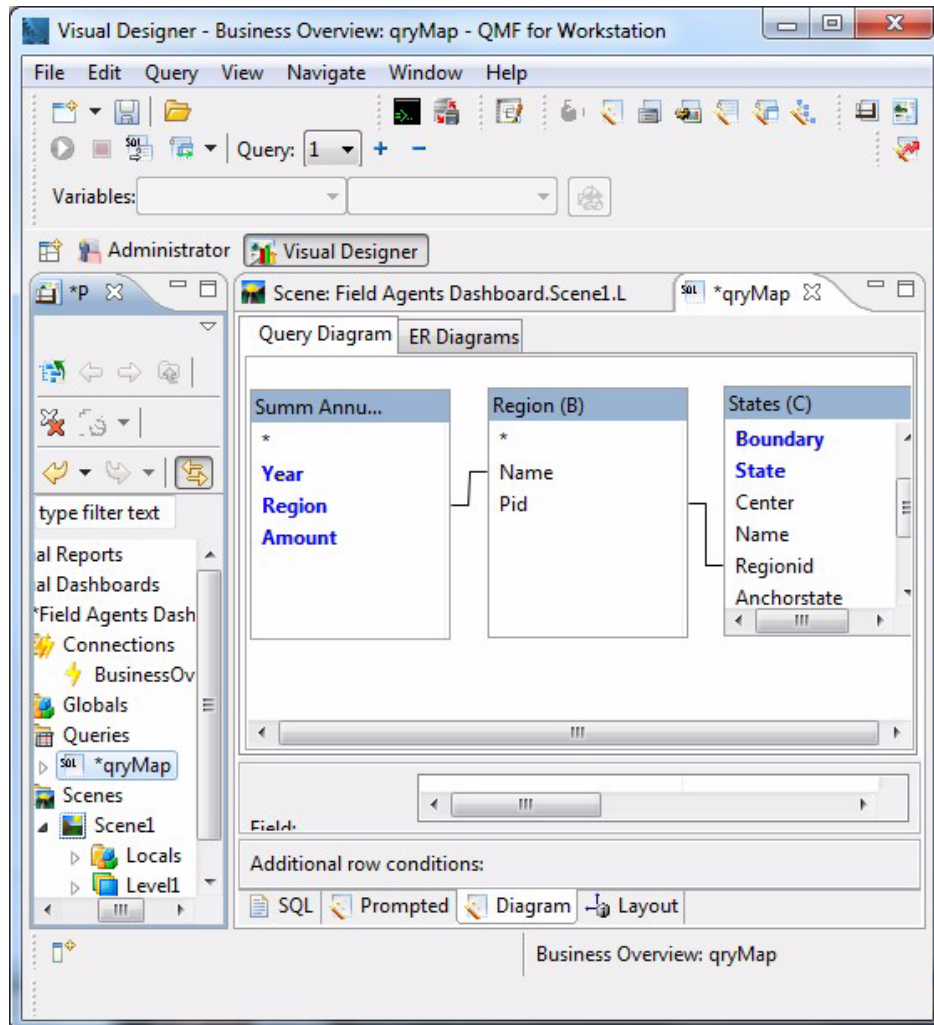


Figure 14-14 Joining virtual tables in the query

After the first query that will supply data for the dashboard is complete, you can begin designing the first dashboard scene. The field agent dashboard will have three scenes:

- ▶ Scene 1 provides the insurance business overview by geography and allows agents to drill into specific geographies.
- ▶ Scene 2 displays historic purchasing activity for a specific customer selected in scene 1.
- ▶ Scene 3 displays real-time purchasing activity in the last 30 days for the customer selected in scene 2.

14.3.5 Designing scene 1: Insurance business overview

As explained previously, Spiffy's field agents require a business overview component in their dashboard, which will provide summary data of customers and products by geography and will serve as a jumping off point for them to drill down into additional data for specific customers. To display the necessary information in this Business Overview scene, Spiffy's IT staff will add a map, two charts, and a list.

Adding a map

To display overview information geographically to the field agents, we must add a linear map, which displays spatial data in a geographic context. The map will be colorized to show three different geographic regions. It will be populated by data returned by the qryMap query we defined earlier in 14.3.4, “Specifying the first query that will supply data to the dashboard” on page 304.

For instructions on how to insert a linear map object, see scenario 2 in 14.4.5, “Designing the scene: Adding a geospatial map driven by selections in the column chart” on page 338.

After the map has been added to the scene, the next step is to colorize each of three regions on the map using the Expression Designer, shown in Figure 14-15.

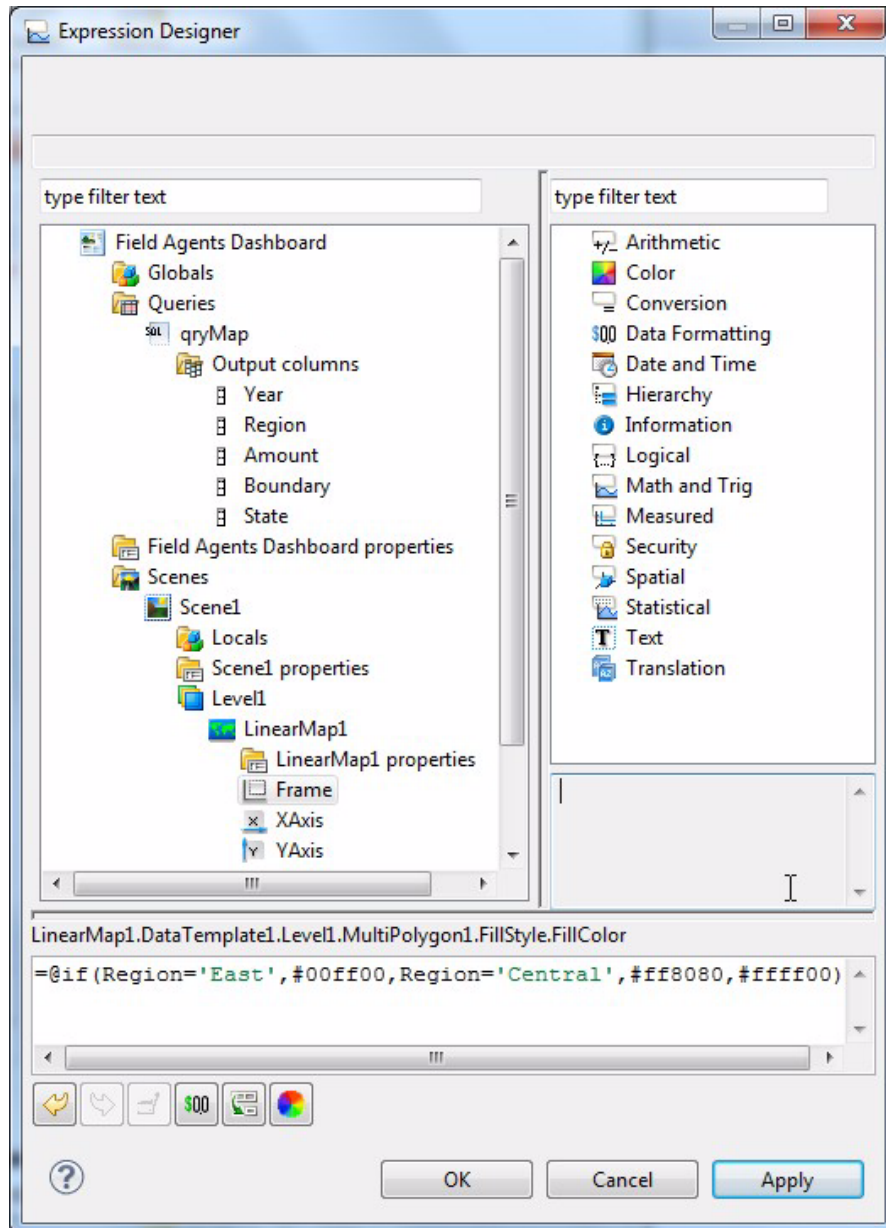


Figure 14-15 Setting color for each region on the map

Figure 14-16 shows how to set the tooltip for each state, so that it shows the selected year, the region the state is in, and the annual sales for that region. (Scenario 2, later in this chapter, walks you through using the Expression Designer and defining tooltips in more detail.)

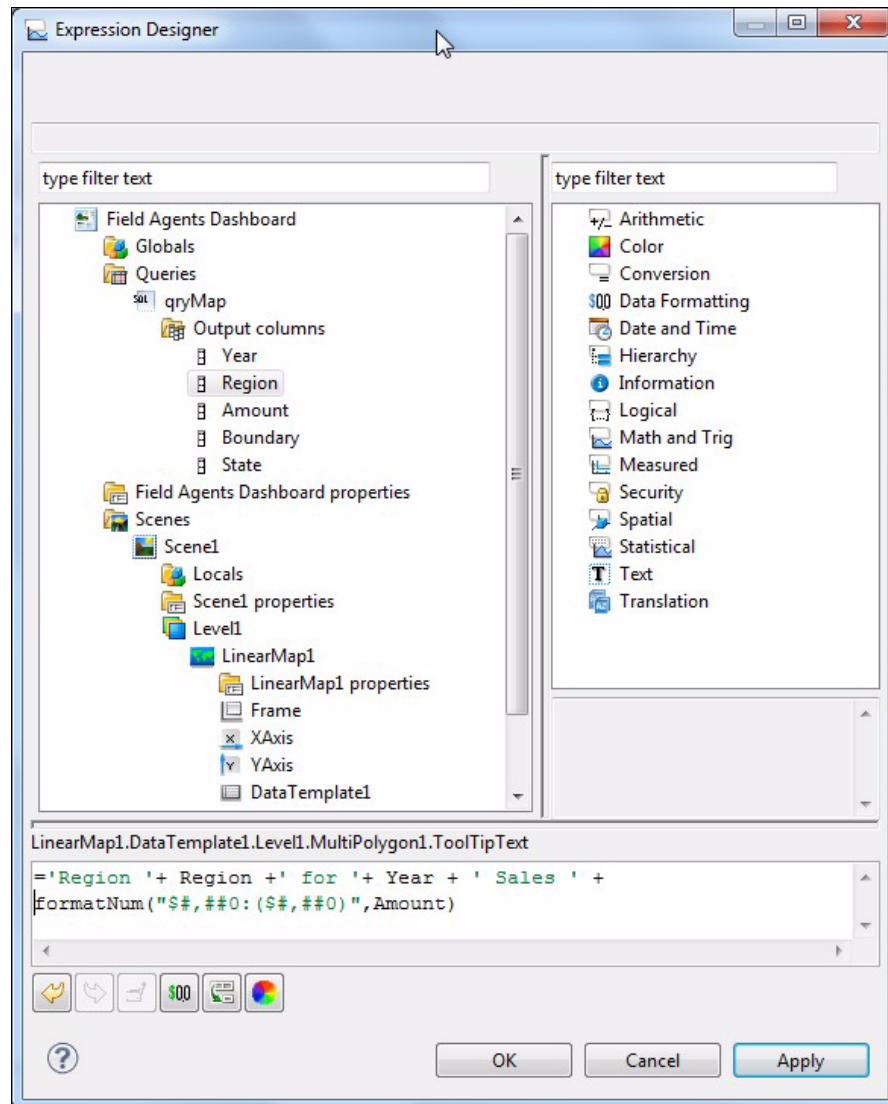


Figure 14-16 Setting the tooltip for each state

After the colors are set and the tooltip is complete, we can click the **Runtime** tab to see how the map will appear to dashboard users. The finished map is shown in Figure 14-17.

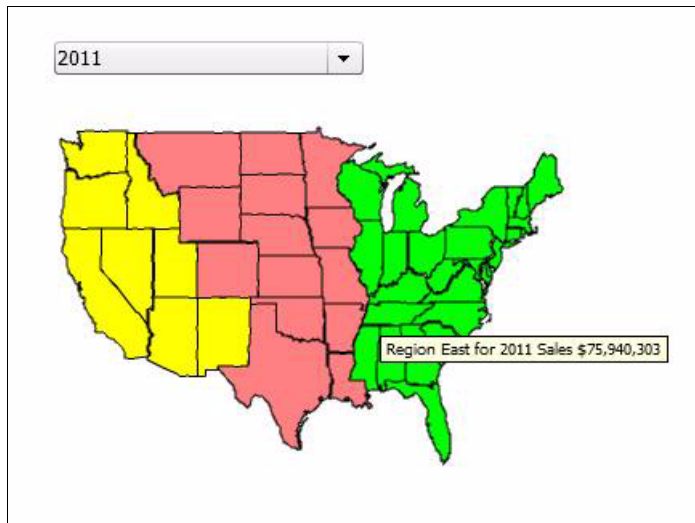


Figure 14-17 The finished map and tooltip

Adding charts

Spiffy's annual sales for all three regions shown in the map in Figure 14-17 must also be able to be displayed in a column chart in the Business Overview scene. The query to do so is shown in Figure 14-18.

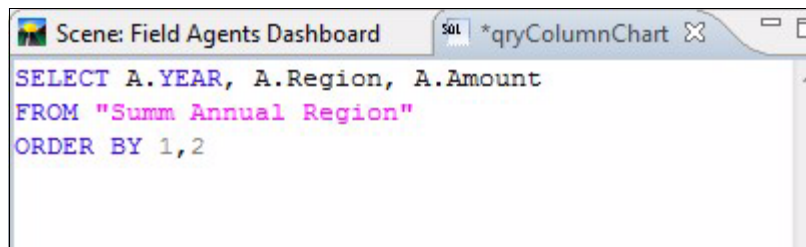


Figure 14-18 The qryColumnChart query populates the column chart in the Business Overview scene

After creating the query that will populate the column chart, we can add the column chart object to the scene. For details on how to add the column chart, see “Adding the column chart object to the canvas” on page 330.

The results are displayed in Figure 14-19, showing both the finished map and column chart as they will appear in the dashboard.

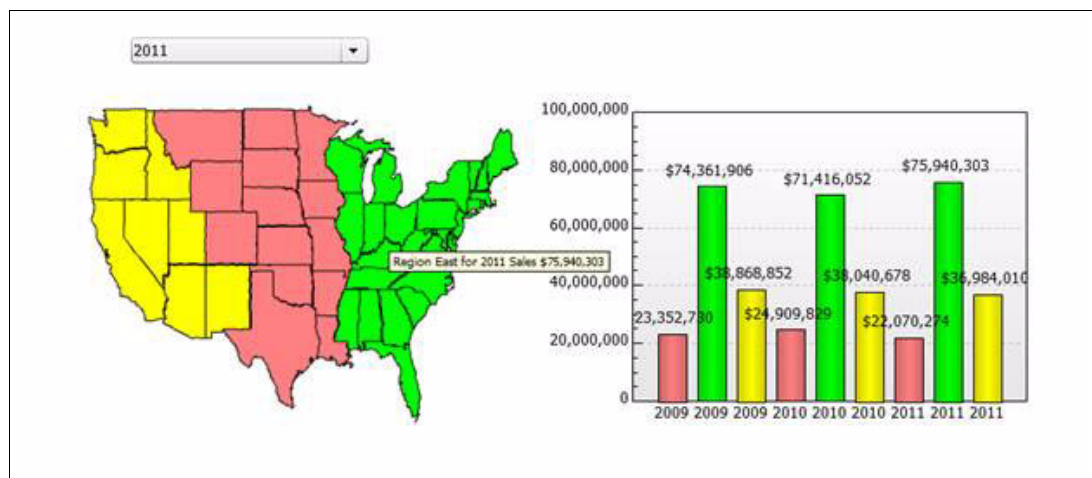


Figure 14-19 Map and column chart showing annual sales by region

Field agents want an additional chart that shows the sales by product for the selected region. The region needs to be selectable from both the map and the column chart. This selection, called a click event, will drive what is displayed in the sales-by-product chart, allowing the agents to see more detail with a single mouse click. You will read more about how to create a click event in scenario 2, later in this chapter.

In this example, the dashboard user clicks on the column chart or the multipolygon map and the agent's choices for region and year become parameters that then drive values in the sales-by-product chart. The query that will drive this sales-by-product chart is shown in Figure 14-20.

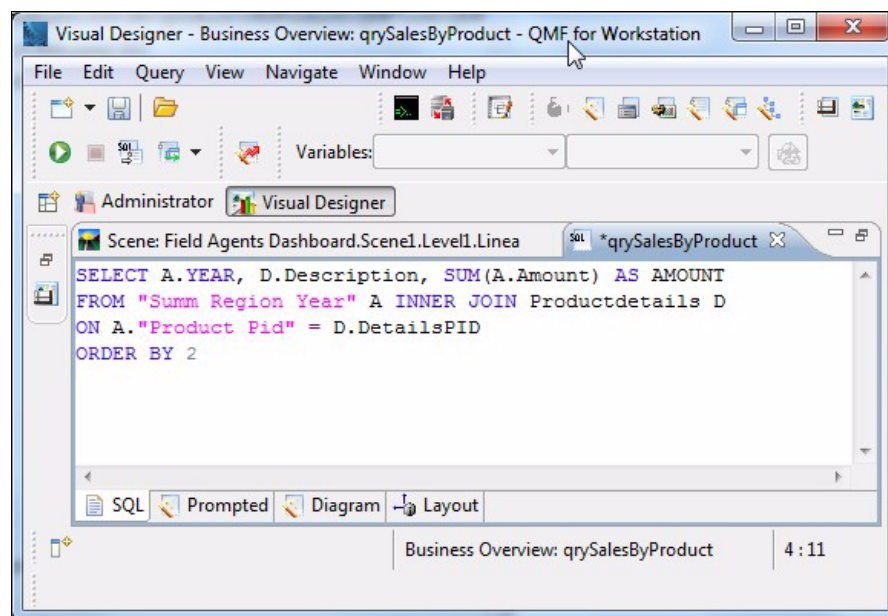


Figure 14-20 qrySalesByProduct, which has two parameter values, year and region

Figure 14-21 shows the properties sheet for this query, where the query parameters are set. To the right, you can see the column chart that is associated with qrySalesByProduct.

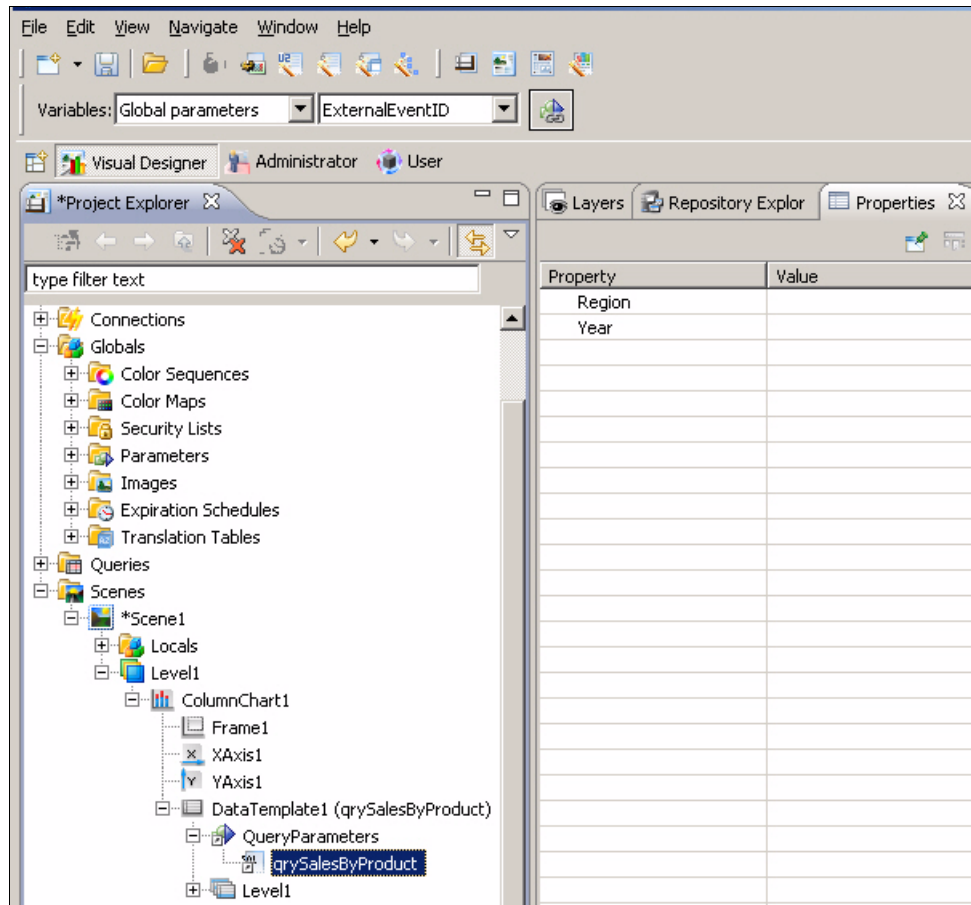


Figure 14-21 Column chart associated with qrySalesByProduct, where query parameter values are set

As shown in Figure 14-22, we now have a composite dashboard with a regional map and two linked column charts that allow Spiffy's field agents to look at the data for the three regions and drill down to see sales by product for whichever year and region are selected.

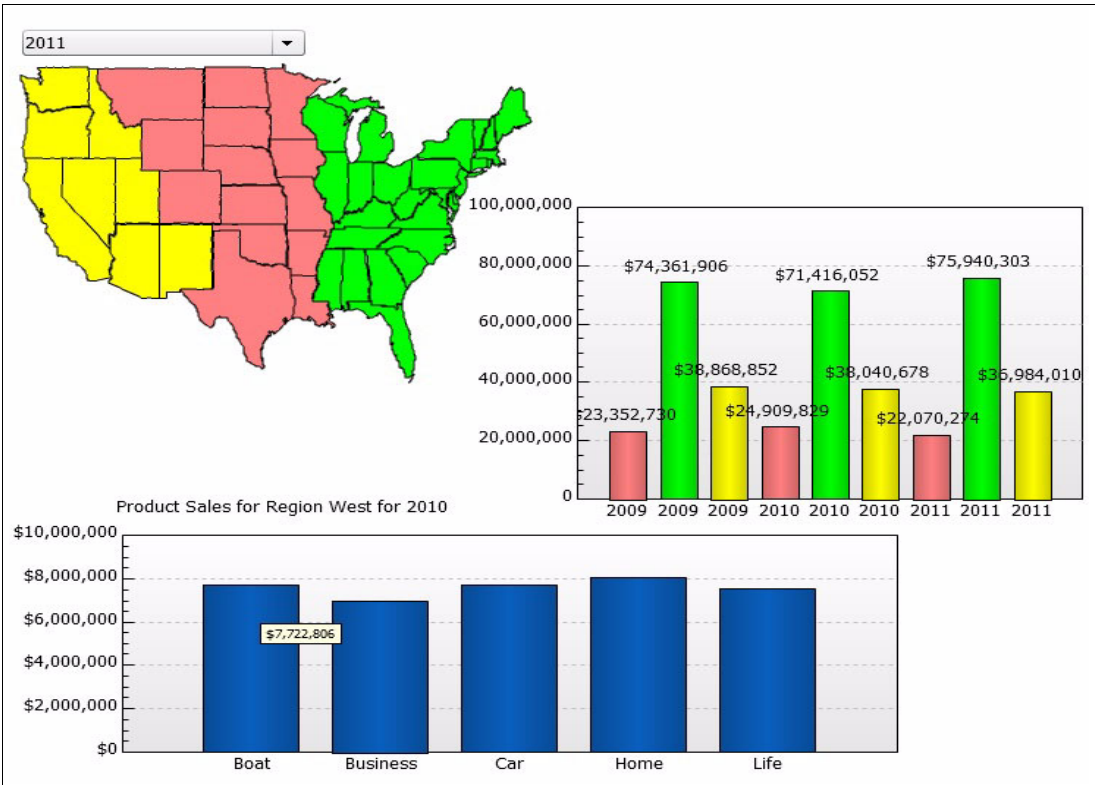


Figure 14-22 Second column chart, which shows insurance product sales for selected region and year

Adding a table

The last control Spiffy requires for its Business Overview scene is a list of customers who have purchased the type of insurance selected in the sales-by-product chart shown at the bottom of Figure 14-22. It typically results in a long list of names, so in the dashboard we will display them as a table that shows only 15 rows at a time. Navigation buttons (created automatically) will provide a means for field agents to scroll the data.

To populate the customer table control with data, Spiffy creates a new query, called qryCustomerList, which has three parameters: region, year, and selected product (product ID, or “PID”). This query will populate the list of customers in the Business Overview scene.

Two different views are of this query shown in Figure 14-23 and Figure 14-24.

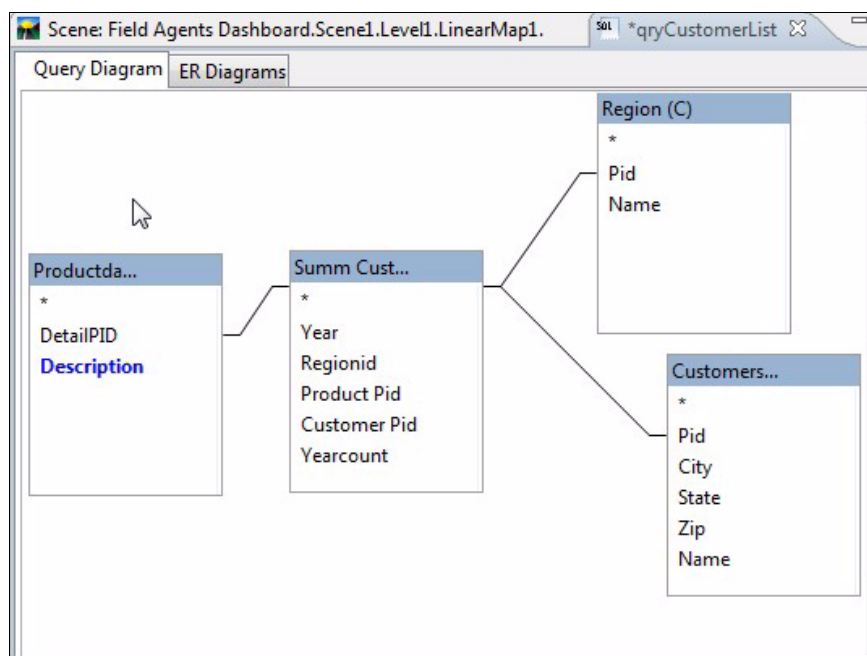


Figure 14-23 Query diagram view of the qryCustomerList query, which populates the list of customers

The Description column has been selected. If other columns are selected as appropriate, and a parameterized WHERE clause is added, the query shown in Figure 14-24 is generated. (See Chapter 9, “Getting to the data you need: Query methods” on page 191 for details on how to create queries.)

```
SELECT DISTINCT D.Name, C.Name AS CUSTOMER_NAME, B.Description, A."Year"
FROM "Summ Customer Annual Activity" A INNER JOIN Productdetails B ON A.
"Product Pid" = B.DetailsPID INNER JOIN Customers C ON A."Customer Pid" =
C.Pid INNER JOIN Region D ON A.Regionid = D.Pid
WHERE (D.Name = &Region)
AND (A."Year" = &YEAR)
AND B.Description = &Product
ORDER BY 1 ASC, 2 ASC, 4 ASC
```

Figure 14-24 SQL editor view of the same query

When the query that will populate the customer list has been created, we can add the table control to the dashboard. Scenario 2, later in this chapter, shows how to add a table control step by step. For details, see 14.4.6, “Designing the scene: Adding table driven by state selected on the map” on page 351.

Figure 14-25 shows the finished customer list table at the bottom right of the Business Overview scene.

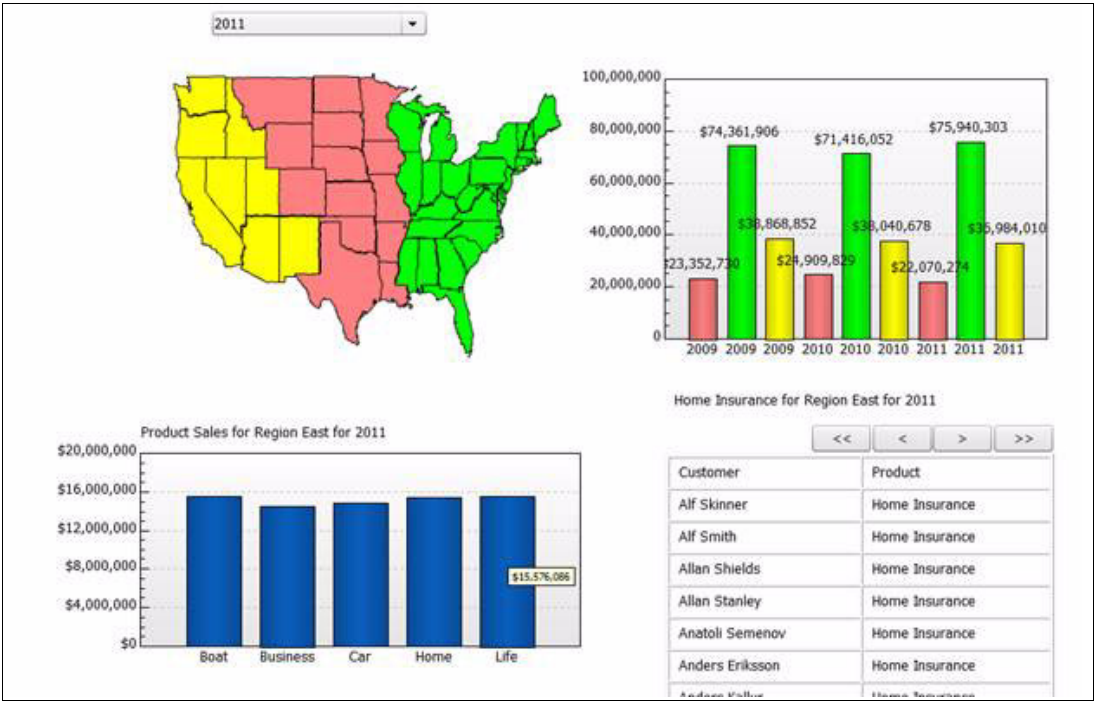


Figure 14-25 Scrollable table that lists customers for selected year, region, and product

Spiffy’s field agents want to see both historic and real-time purchasing activity by customer when a specific customer is selected from the list in the scene shown in Figure 14-25. To accomplish this result, two additional scenes are added, which we show next.

14.3.6 Designing scene 2: Historic customer activity

For agents to be able to see a history of purchasing activity for a particular customer selected in the table shown in the first scene, a click event must be added to the label control that displays the customer name in the table. This click event, shown in Figure 14-26, provides a link or “jump” to the second scene, which Spiffy names CustomerSummary. (We cover how to create a click event in more detail later in this chapter, in scenario 2.)

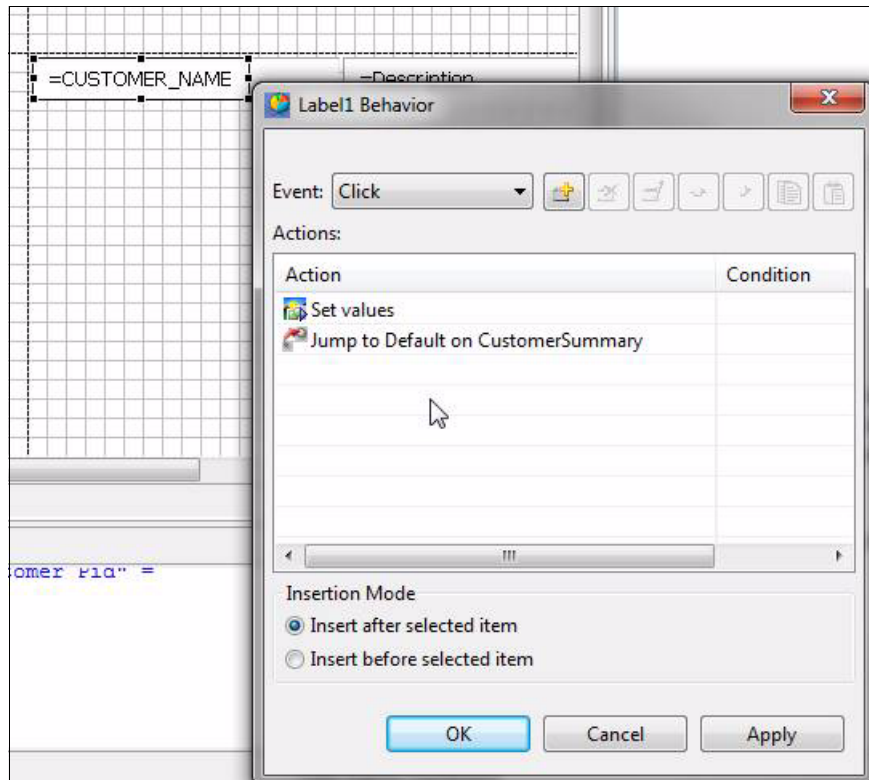


Figure 14-26 A click event on customer label in scene 1 that jumps to scene 2 (CustomerSummary)

Scene 2, called CustomerSummary, has three main controls:

- ▶ A combo box that lists the states
- ▶ A second combo box that is populated with a list of customer names for the state selected from the first combo box.
- ▶ A matrix (which essentially looks like a table) that is populated with historic purchasing activity for the customer selected from the second combo box

We begin by adding a scene to the dashboard. To add a scene, from the Project Explorer tree right-click the **Scenes** folder for the dashboard to which you want to add the scene, and select **New Scene** from the context menu.

When we have a new, empty scene, we begin design of the scene by adding the combo box that will list the states. Before we add the control, we first must specify the query that will provide the data for the combo box. This query is shown in Figure 14-27.

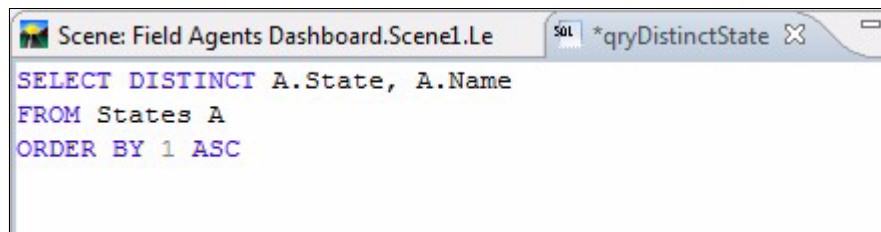


Figure 14-27 The *qryDistinctStates* query, which populates the first combo box in the *CustomerSummary* scene with a list of states

The combo box displays the state name. Spiffy uses state abbreviations (“A.State” column, as shown in Figure 14-27) to filter the list of customer names. The combo boxes must be linked, so an SQL statement is used to populate the list of customer names for a state as shown in Figure 14-28.

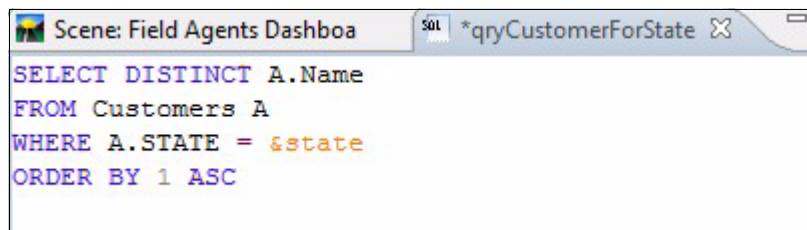


Figure 14-28 *qryCustomerForStates* query, which uses a parameter for the dashboard user's state selection

With the two boxes associated with each of the queries on the dashboard, we use the Connectivity tool to define the dependency between them, so that a change in the selected state will update the list of customer names:

1. In the Palette view, click the Connectivity tool.
2. Click the control that drives that will be displayed in the other control.
3. Click the second control to link the two events. We also need to specify the values to be passed. This sequence is shown in Figure 14-29 to Figure 14-32 and described here:
 - Figure 14-29: Linking two controls together using the Connectivity tool
 - Figure 14-30: Using the Connectivity tool to select the control whose value will drive what appears in the second control
 - Figure 14-31: Using the Connectivity tool to set how the passed-in value is going to be received (as a parameter value specified in the query in Figure 14-28 on page 315)
 - Figure 14-32: Using the Connectivity tool to pass the state selected by the user to the query parameter called “state”

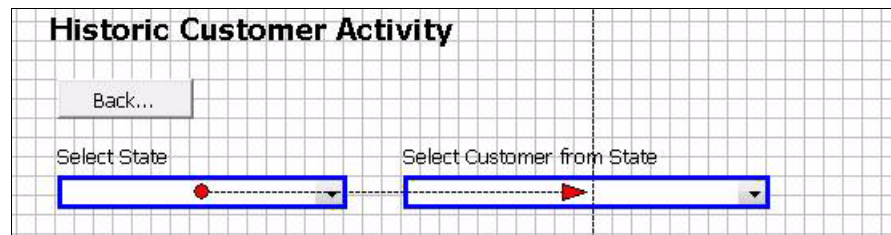


Figure 14-29 Linking two controls together using the Connectivity tool

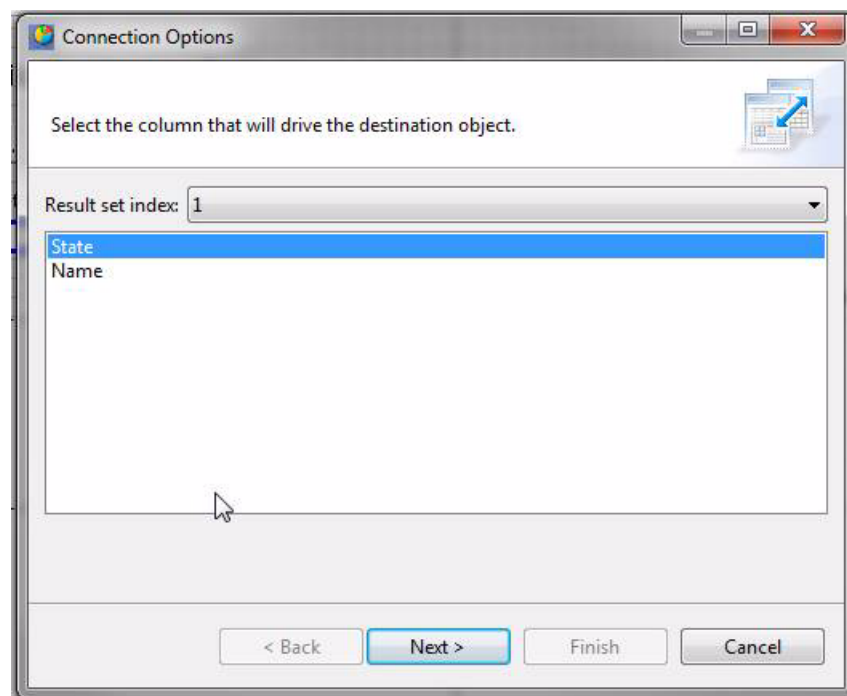


Figure 14-30 Selecting the control whose value will drive what appears in the second control

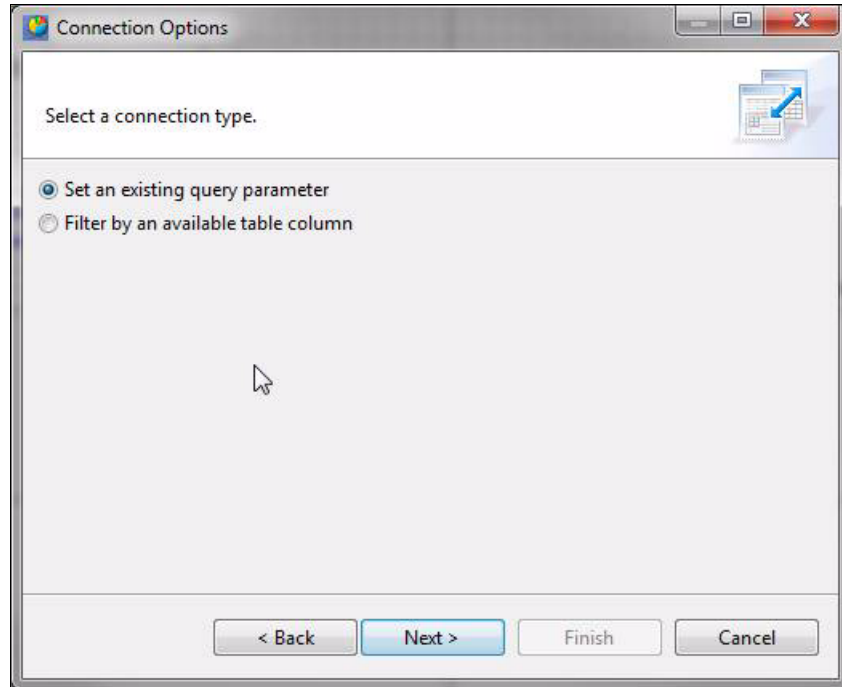


Figure 14-31 Setting how the passed-in value is going to be received (as a parameter value)

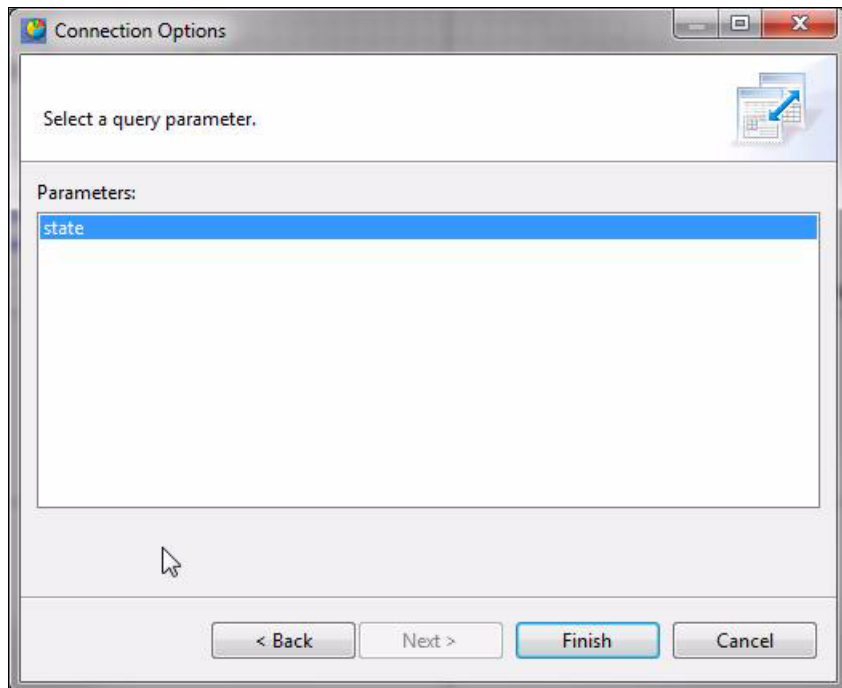


Figure 14-32 Passing the state selected by the user to the query parameter called “state”

Now that we have created the queries that will drive the combo boxes, we can add the combo box controls to the canvas.

To add a combo box:

1. Locate the Controls section of the **Palette** view.
2. Scroll through this section of the palette until you see the Combo object. Double-click it, which launches the ComboBox wizard.
3. In the first dialog, select **Yes** for the question “Do you want to populate the list with data from a query?”
4. In the next dialog, select the query that will drive the data in the combo box.
5. In the final dialog, choose the field containing the list of options to be displayed, then click **Finish**.

Selecting a state drives what customer names are available in the customer combo box, shown in Figure 14-33. Selecting a customer name will populate the matrix control with historic sales for the selected customer and state.

Historic Customer Activity

Back...

Select State: Colorado

Select Customer from State: John Doe

Date	State	City	Product	Policy #	Amount
Jan-09	CO	FORT COLLINS	Boat Insurance	#AC_F101946	\$30,688
Sep-09	CO	FORT COLLINS	Business Insurance	#AC_F109829	\$441
Jun-10	CO	FORT COLLINS	Home Insurance	#AC_F110326	\$33,689
Oct-10	CO	FORT COLLINS	Boat Insurance	#AC_F107051	\$1,562
May-11	CO	FORT COLLINS	Boat Insurance	#AC_F113177	\$18,552

Figure 14-33 Customer Summary scene after historic sales by customer has been added

14.3.7 Designing scene 3: Customer activity in the last 30 days

In the scene in Figure 14-33, which is the second scene of the field agent dashboard, there is a **View Most Recent Activity** button. It jumps to a third dashboard scene, called Customer Activity, where Spiffy agents can view IMS data for the most recent transactions for this customer. Agents can then click the **Back** button to navigate back to prior scenes.

For the Customer Activity scene, we need to retrieve data from IMS. Though IMS tables could be added to the Business Overview virtual data source already in place, we define a new virtual data source, called Business Activity, for the IMS data, following the process outlined in Chapter 7, “Defining virtual data sources to reduce complexity for business users” on page 133. After the new virtual data source has been defined, a connection to it can be configured, as shown in Figure 14-34.

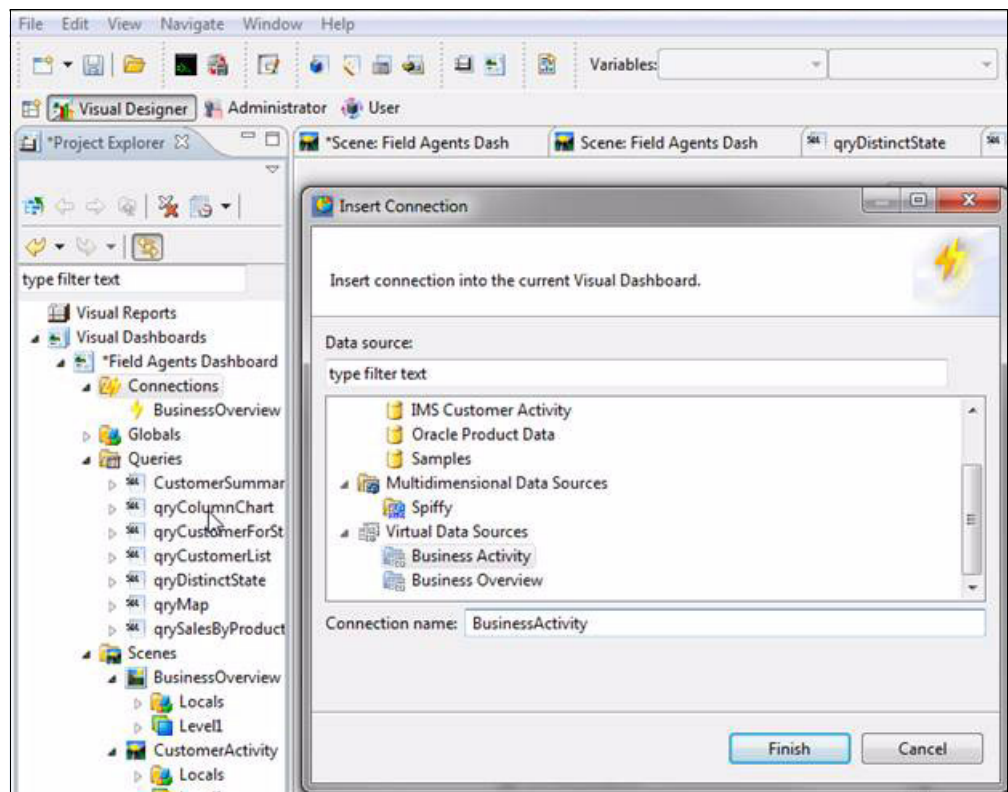


Figure 14-34 Defining a dashboard connection to the new IMS virtual data source

At this point, Spiffy's dashboard designers can follow the same process as they did for prior scenes, resulting in the completed third dashboard scene, shown in Figure 14-35.



Figure 14-35 Current Customer Activity scene in Spiffy's field agents dashboard, which pulls data directly from IMS

14.4 Scenario 2: Market analysis dashboard (OLAP and relational data) with one scene

Due to some recent acquisitions, Spiffy is in the process of expanding into new markets, and therefore the company is actively looking at enhancing existing insurance products as well as rolling out new ones. Such evaluation requires that Spiffy's market analysts have access to the company's OLAP data in a format that they can readily consume. Sales and customer data must be able to be sliced and diced at many different dimensional levels, and so Spiffy's IT department has decided to create a market analysis dashboard for this user population at the company. Though, over time, the company will be consolidating its business analytics data sources, in the near term the data sources that provide data for this dashboard will be SQL Server (Analysis Services) as well as DB2 for Linux, UNIX, and Windows (DB2 LUW).

The market analysis dashboard allows Spiffy's market analysts to examine business across all territories and drill down into product revenue for a given territory and year, then further drill into individual customer data in each territory. Many business analytics offerings provide drill-down capabilities by navigating the user away from the dashboard. As seen earlier in the field agent dashboard scenario and, as we show you in this scenario, QMF dashboards provide drill-down capability directly within the dashboard, such that the controls placed on the dashboard provide a data-driven view of business metrics all in one place.

The market analysis dashboard provides the following features:

- ▶ A grid that displays sales by region and allows dashboard users to slice the data by product type (type of insurance).
- ▶ A column chart that displays sales by time and allows dashboard users to slice the data by year, then quarter within year, then month within quarter.
- ▶ A geospatial map that displays the volume of sales by state, with colors to indicate sales from high to low. The data displayed in the map is driven by the date range selected in the column chart.
- ▶ A table that displays the underlying customer records for the state selected in the map and the date range selected in the column chart.

To create a new, empty dashboard for this scenario, from the Visual Designer perspective, we select **File** → **New** → **Visual Dashboard** from the menu, as shown previously for scenario 1 in Figure 14-11 on page 302. We name this dashboard “OLAP Analysis”.

14.4.1 Creating data source connections

The data sources for this dashboard are both relational (DB2 for Linux, UNIX, and Windows) and multidimensional (MS Analysis Services OLAP data held in SQL Server). Before connections to these data sources can be created, the data sources must be defined to QMF. The following topics cover how to add the data sources:

- ▶ 6.3.1, “Adding a relational data source” on page 106
- ▶ 6.3.2, “Adding a multidimensional data source” on page 114

For this example, we call our OLAP data source “Spiffy”.

After the data sources have been added as explained in Chapter 6, “Configuring access to data sources and populating user workspaces” on page 89, we can define connections to them. These connections are defined in the same manner as previously shown for scenario 1, by right-clicking **Connections** under the new dashboard name in the Project Explorer tree and selecting **Insert Connection**, as shown in Figure 14-36 for our OLAP data source.

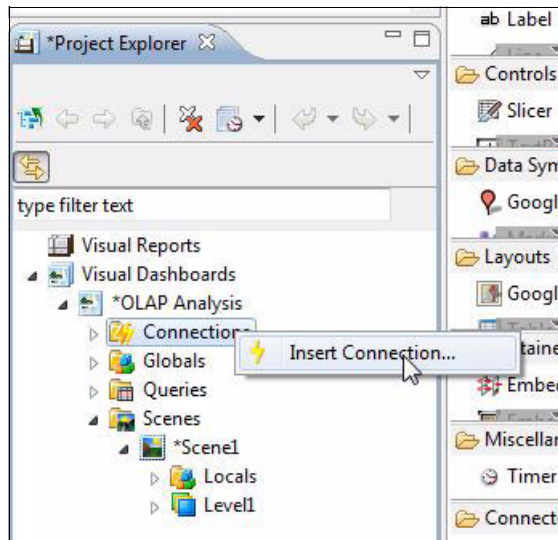


Figure 14-36 Creating a connection to the OLAP data source

In the Insert Connection window, we expand the multidimensional data sources and select **Spiffy**, then change the connection name to SpiffyOLAP, as shown in Figure 14-37.

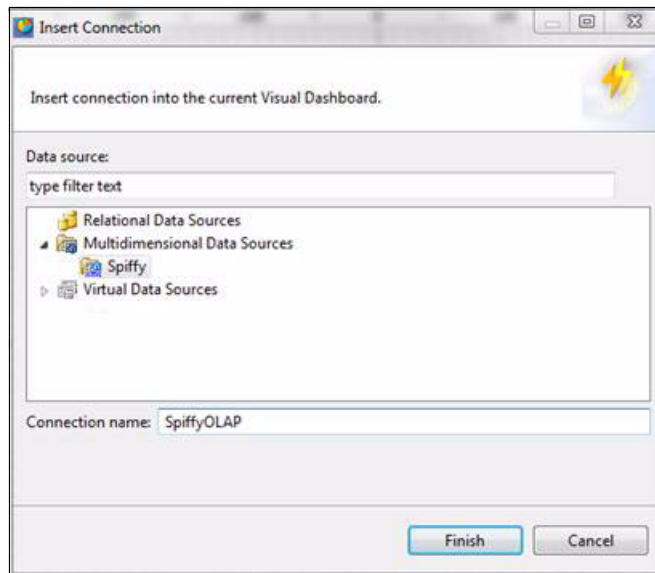


Figure 14-37 Naming the connection to the OLAP data source

Click **Finish** to close the Insert Connection window.

We repeat the foregoing steps to define a connection to Spiffy's relational data source under DB2 LUW, which we call CustomerData.

14.4.2 Specifying the first query that will supply data to the dashboard

As in scenario 1 earlier in this chapter, after connections to the data sources are defined, we begin by creating the first query that will supply data to the dashboard. You need not create all the queries for the dashboard at this time; additional queries can be added at any time.

The first object we add to the dashboard is a grid that displays Spiffy's OLAP-based sales data by year and region, so the first query we create will populate this grid. (The query shown in this example is the same OLAP query presented in Chapter 9, “Getting to the data you need: Query methods” on page 191, but we reproduce it here so that you can see it in the context of the market analysis dashboard.)

To create the query:

1. In the Project Explorer tree, right-click queries and select **Insert Query**, as shown in Figure 14-38.

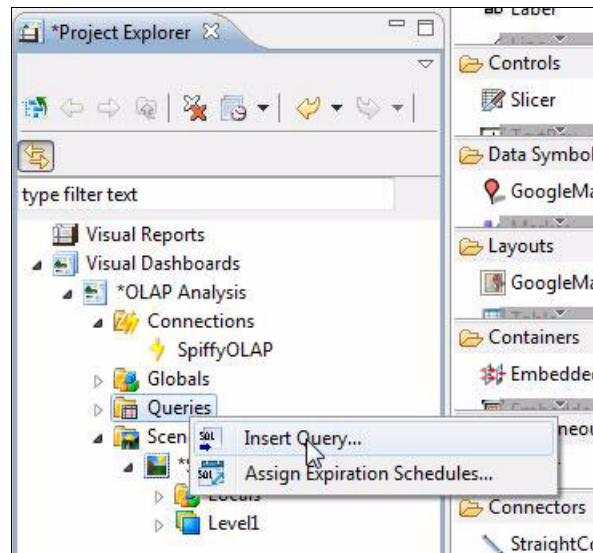


Figure 14-38 Starting the OLAP query

2. Change the query name to qryGrid and click **Finish**. We are now in the QMF OLAP (cube structure) query editor.
3. Select the cube to use by right-clicking **Cube** and selecting **Set Cube**. Select **Spiffy** and click **Finish**, as shown in Figure 14-39.

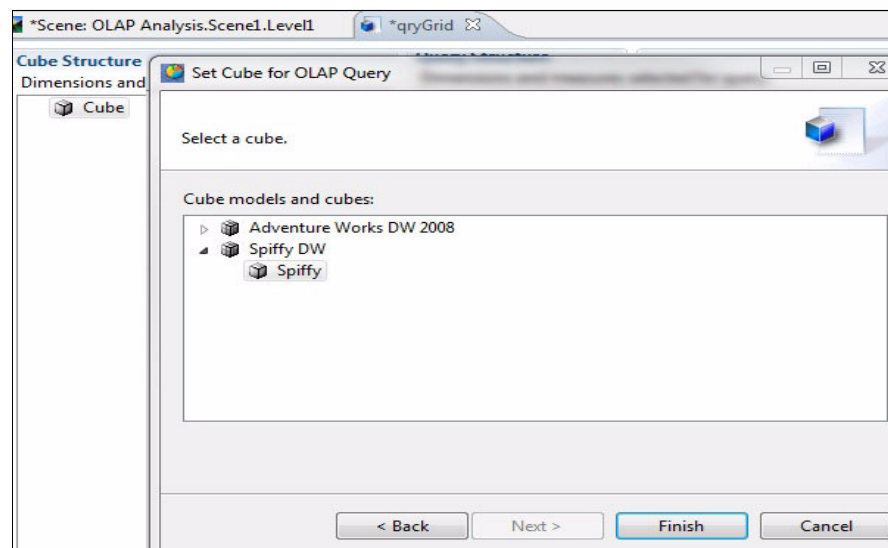


Figure 14-39 Setting the cube for the OLAP query

We now see a list of dimensions and measures on the left and the query structure on the right.

4. Define the dimensions and measures to use for the query. As shown in Figure 14-40:
 - a. Drag **Time** to be the top dimension.
 - b. Drag **Market** to be the side dimension.
 - c. Drag **Sales** to be the measure.

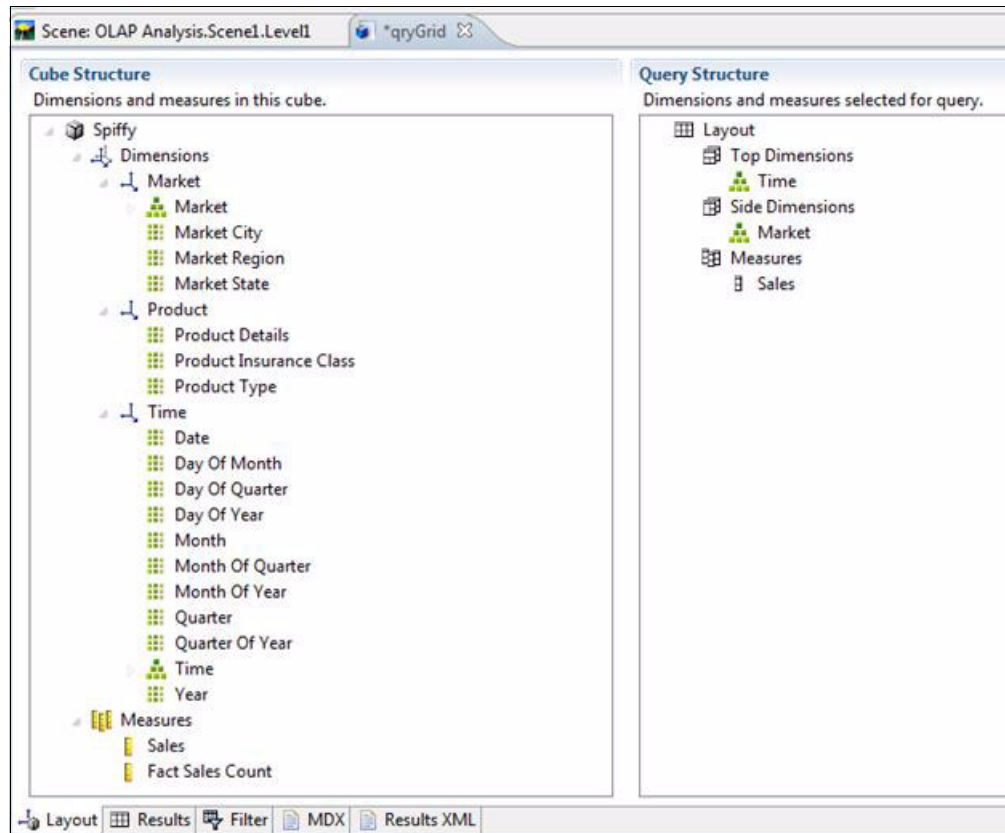


Figure 14-40 Creating the OLAP query structure

- Click the Results tab to see the data, as shown in Figure 14-41.

	1	2	3	4
Time	+ Calendar 2009	+ Calendar 2010	+ Calendar 2011	
Market	Sales	Sales	Sales	
1 + Central	26159488	27103385	24598108	
2 + East	82514186	79672850	82208937	
3 + West	43897568	42813736	41829842	

Figure 14-41 OLAP query results, a cross-tab of sales by year and market (region)

The query shows a cross-tab of Spiffy's sales by year and market (region). Clicking the + signs will cause the grid to expand and show an additional level of data, as seen in Figure 14-42.

	1	2	3	4	5	6	7
Time	- Calendar 2009	+ Quarter 1, 2009	+ Quarter 2, 2009	+ Quarter 3, 2009	+ Quarter 4, 2009	+ Calendar 2010	
Market	Sales	Sales	Sales	Sales	Sales	Sales	
1 - Central	26159488	7419072	5994881	5670053	7075482	27103385	
2 + AR	1262012	473265	237961	338586	212200	1215495	
3 + CO	2450654	758151	444503	457304	790696	3053515	
4 + IA	8957830	2497563	2195278	1829423	2435566	9228614	
5 + KS	1999124	376307	420663	493859	708295	2405487	
6 + LA	1397663	346930	361251	296246	393236	875857	
7 + MN	430991	87882	57704	103215	182190	728519	
8 + MO	700148	309473	91676	181242	117757	908637	
9 + MT	12844		12844			75500	
10 + ND	1111				1111	11699	
11 + NE	290650	139627		58471	92552	274427	
12 + OK	457163	49396	175034	63678	169055	220526	
13 + SD	134561	44148		67265	23148	127853	
14 + TX	7836235	2209541	1917495	1771410	1937789	7649368	
15 + WY	228502	126789	80472	9354	11887	327888	
16 + East	82514186	21006903	20096992	20684247	20726044	79672850	
17 + West	43897568	10236956	10237074	11759529	11664009	42813736	

Figure 14-42 Expanded OLAP data, showing sales for calendar year 2009 in the central region

14.4.3 Designing the scene: Adding grid for sales by region with product type slicer

The next step in designing this dashboard is to place a control on the dashboard scene that will display the data from query qryGrid. We use a grid to display the data.

Adding the grid object to the canvas

To place the grid:

1. From the Project Explorer tree, select Level1 in Scene1, as shown in Figure 14-43.

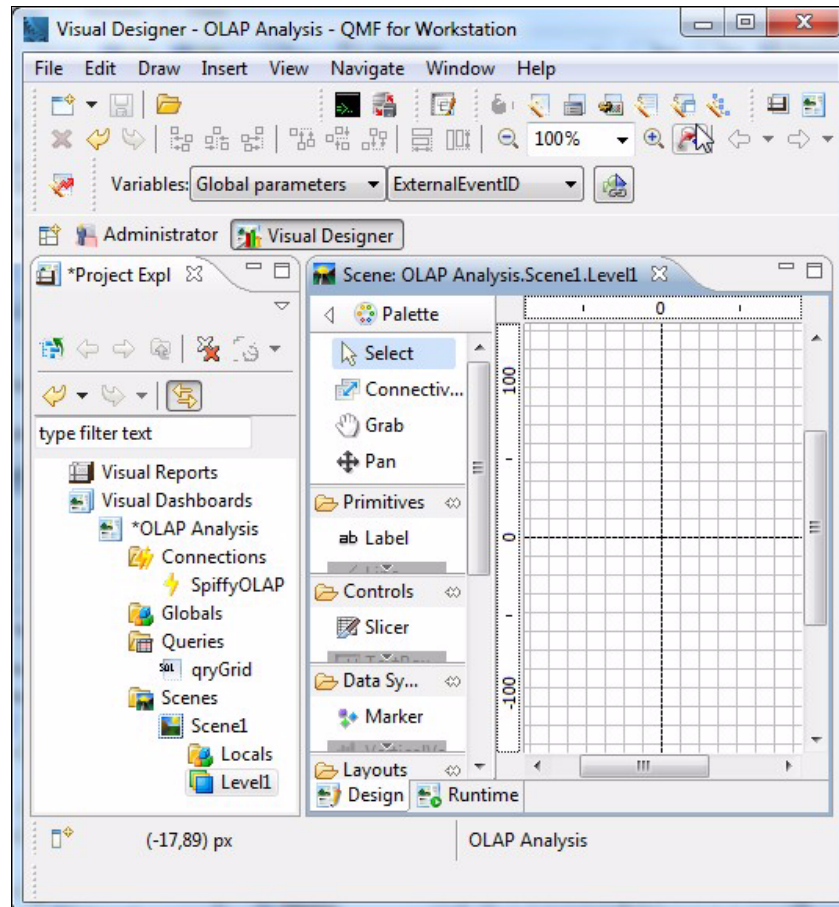


Figure 14-43 Selecting Level1 in Project Explorer tree and then clicking inside the scene canvas

2. Click inside the scene canvas and select **Insert** → **Layout** from the menu. The Layout wizard appears.

- From the Layout wizard, select **Pattern** from the Category list and select Grid in the Style pane of the wizard, as shown in Figure 14-44.

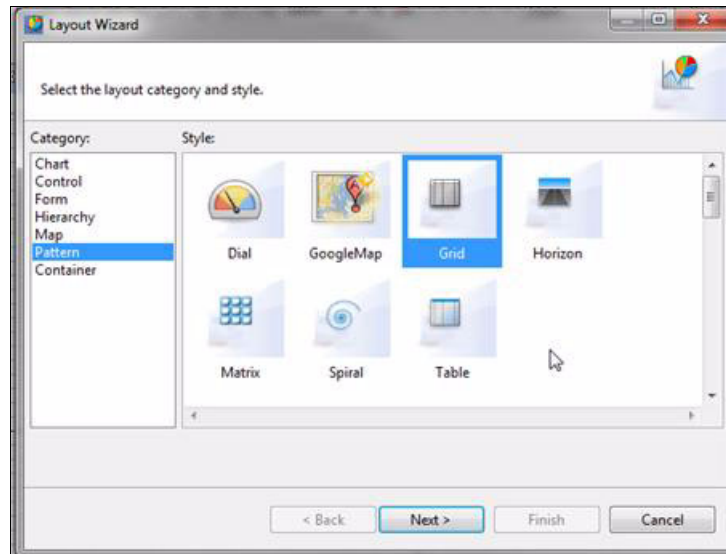


Figure 14-44 Selecting a grid object from the Layout wizard

- Click **Next** and select the query named **qryGrid**, which was just created, then click **Finish**.
The grid object, named Grid1 by default, now appears in the dashboard Design tab. You can resize the grid by clicking it and dragging either the side or corner.
- Click the **Runtime** tab to see how the grid will appear to dashboard users (Figure 14-45).

	1	2	3	4
	Time	Calendar 2009	Calendar 2010	Calendar 2011
	Market	Sales	Sales	Sales
1	Central	26159488	27103385	24598108
2	East	82514186	79672850	82208937
3	West	43897568	42813736	41829842
4	AZ	3979189	4667129	3697780
5	BENSON	79400	88555	62595
6	BUCKEYE		96567	73287
7	CAMP VERDE	35796	14752	
8	CAVE CREEK	35957	45713	6526
9	CHANDLER	74845	42479	31795
10	DOUGLAS	33045	57758	142774
11	FLAGSTAFF	79022	30082	49726
12	GILBERT	31463	63321	118096
13	GLENDALE	310424	451531	268875
14	GLOBE		78239	57413
15	GREEN VALLEY	34536	92807	18200
16	HOLBROOK	21285	33462	4151
17	KINGMAN	99468	141100	80813

Figure 14-45 Grid in market analysis dashboard; western region and state of Arizona expanded

The grid is currently showing all the data. To provide market analysts with the capability to further filter the data, we add slicers.

Adding a product type slicer to filter grid data by type of insurance

Slicers can be based on any defined dimension and multiple dimensions can be added. In this case, we add a slicer that allows market analysts to filter the data by product type (type of insurance).

To add the slicer:

1. Click the **Design** tab at the bottom of the scene canvas to ensure that the dashboard is in design mode.
2. Click the scene canvas and select **Insert** → **Control** → **Slicer** from the menu. The Slicer wizard is displayed.
3. Select **qryGrid** and click **Next**.
4. Select the dimension to use. For this example, we select **Product Details**, as shown in Figure 14-46, which will allow filtering of the data by product type.

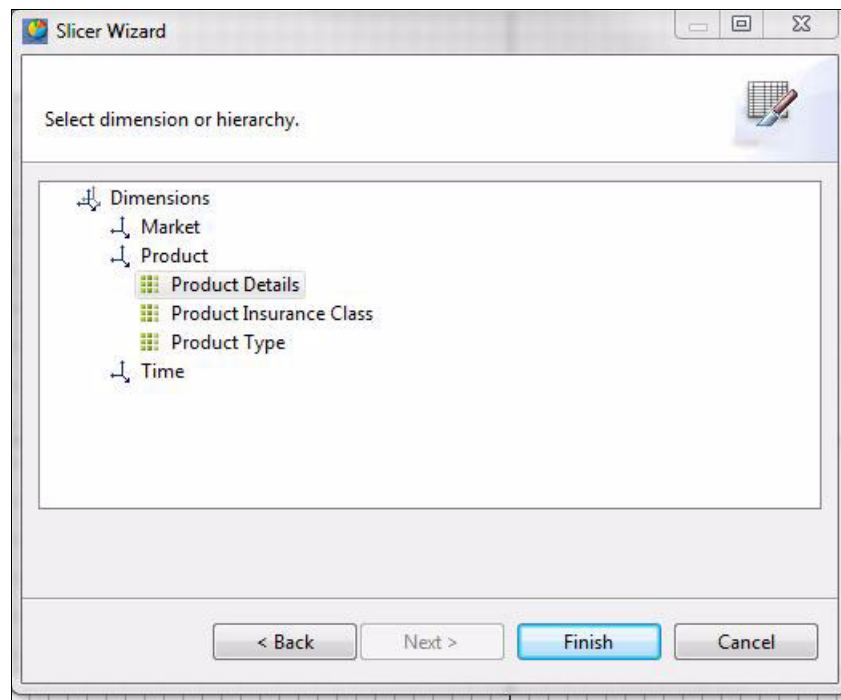


Figure 14-46 Slicer wizard — Selecting dimension

5. Select the resulting slicer control and drag and drop it to the desired location.
6. In the Properties view, change the name of the slicer to something meaningful. We use **Slicer_ProductDetails** for this example, as shown in Figure 14-47.

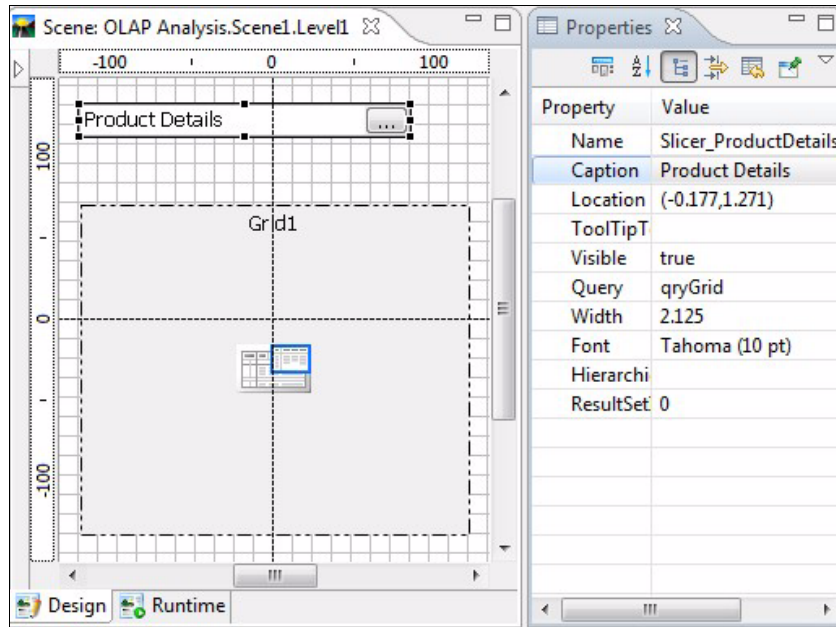


Figure 14-47 Changing the name of the slicer control in the Properties view

7. Associate the newly created slicer control with the grid, so that changes made in the slicer filter the data displayed in the grid:
 - a. While still in design mode, select **Grid1**.
 - b. In the Properties view, click the **Value** field for the Slicers property.
 - c. In the Slicers window shown in Figure 14-48, check the slicer just created, called Slicer_ProductDetails, then click **OK**.

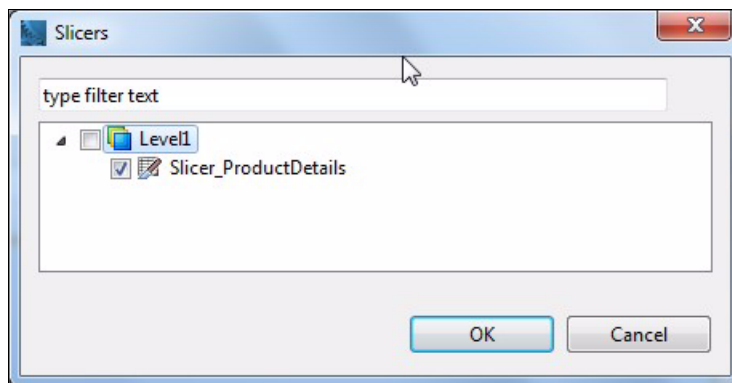


Figure 14-48 Associating the Slicer_ProductDetails slicer with the grid

The slicer has now been added to the dashboard and associated with the grid that we added earlier.

To see the dashboard design so far, as a dashboard user will see it, click the **Runtime** tab at the bottom of the scene canvas. Click the new **Product Details** slicer in the dashboard, and a list of the available filter values appears, as shown in Figure 14-49.

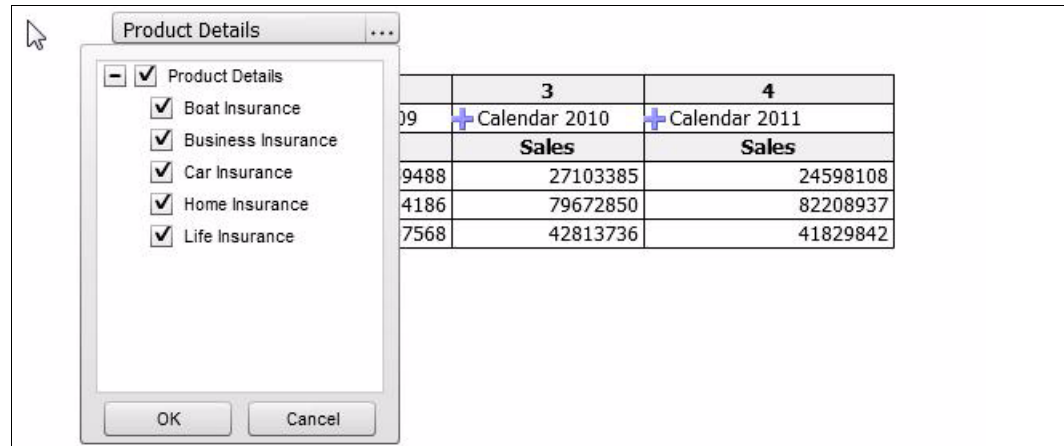


Figure 14-49 Product slicer for type of insurance

If we now deselect all products except car insurance and click **OK**, the grid shows sales data by region for car insurance only, as seen in Figure 14-50.

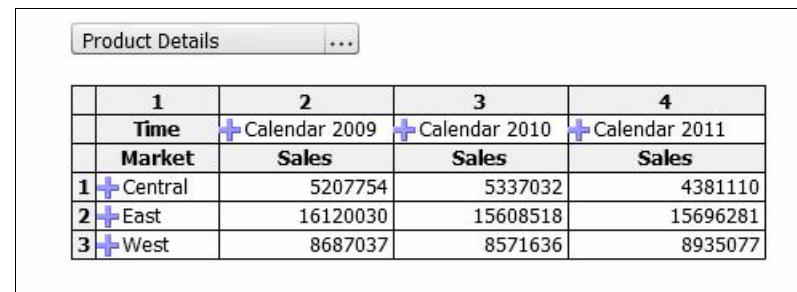


Figure 14-50 Only car insurance is selected in the slicer; grid values are updated to reflect this filter

Tip: You can associate multiple slicers with the same control (grid or chart). Also, the same slicer can be used by more than one control at the same time.

14.4.4 Designing the scene: Adding column chart for sales by time with a time slicer

We now add a column chart to the market analysis dashboard that displays Spiffy's sales by time.

Creating the query that will populate the column chart

Before we can add the control for the column chart, we need to create the query that will populate the chart with data. This query will be called qryChart, and we create it in the same manner that we created the query for the grid that we have already added. You can refer to the earlier discussion for the steps to follow.

A column chart only uses the side dimensions of an OLAP query, so the qryChart query will look like the one shown in Figure 14-51.

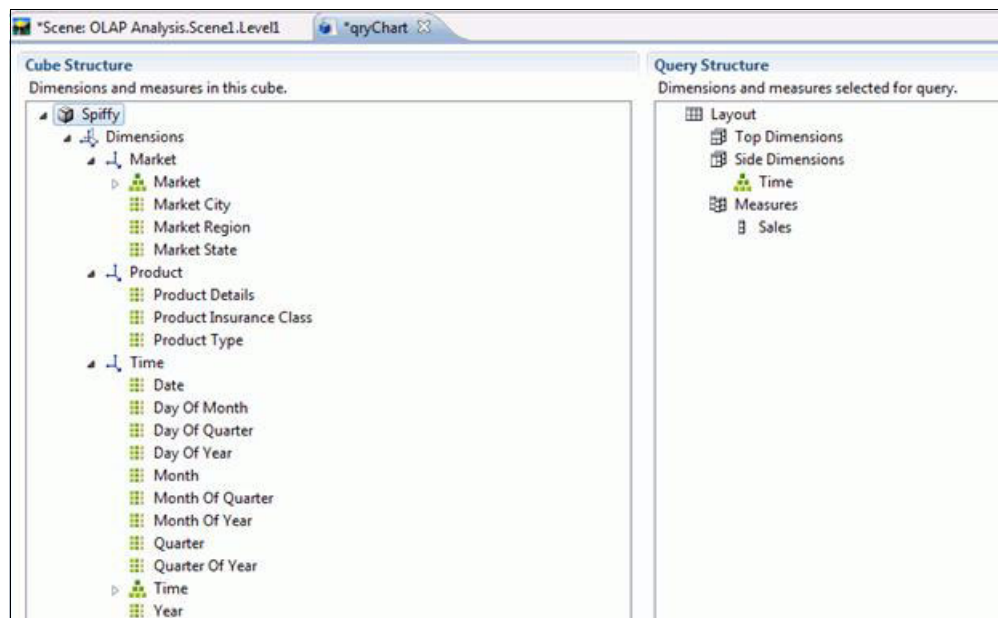


Figure 14-51 qryChart query; no top dimension because a column chart uses only side dimensions

Adding the column chart object to the canvas

After the query that will populate the column chart is defined, we can add the column chart object to the scene canvas as follows:

1. Click the **Design** tab to ensure that you are in design mode.
2. Select **Insert** → **Layout** from the menu. The Layout wizard is displayed.

3. Select **Chart** from the Category List and select ColumnChart from the Style pane, as shown in Figure 14-52. Click **Next**.

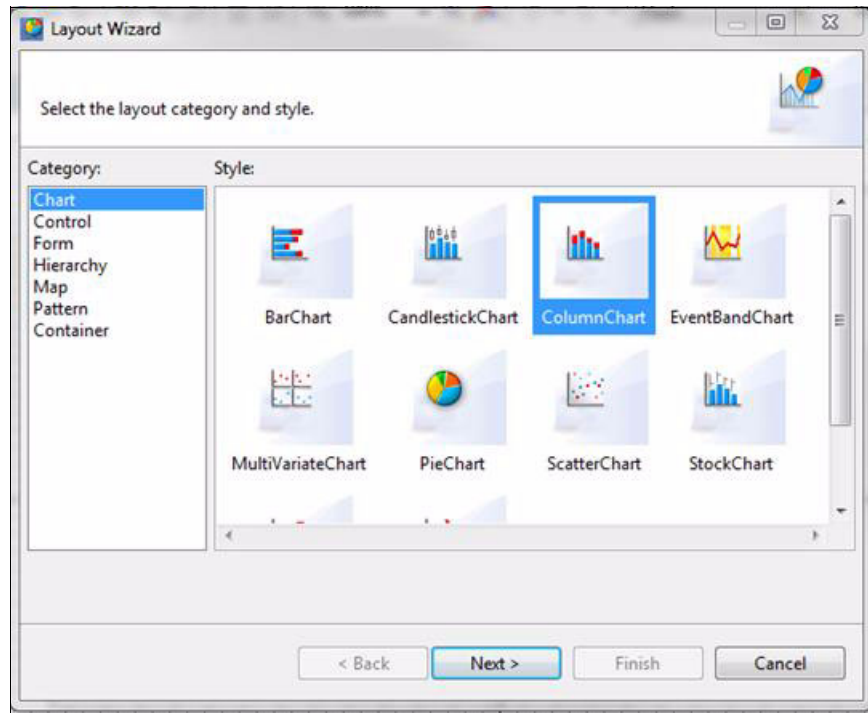


Figure 14-52 Inserting a column chart

4. Select the **qryChart** query to drive the data in the column chart, and click **Next**.
5. Select the **Sales** field, shown in Figure 14-53, as the field to be displayed, and click **Next**.

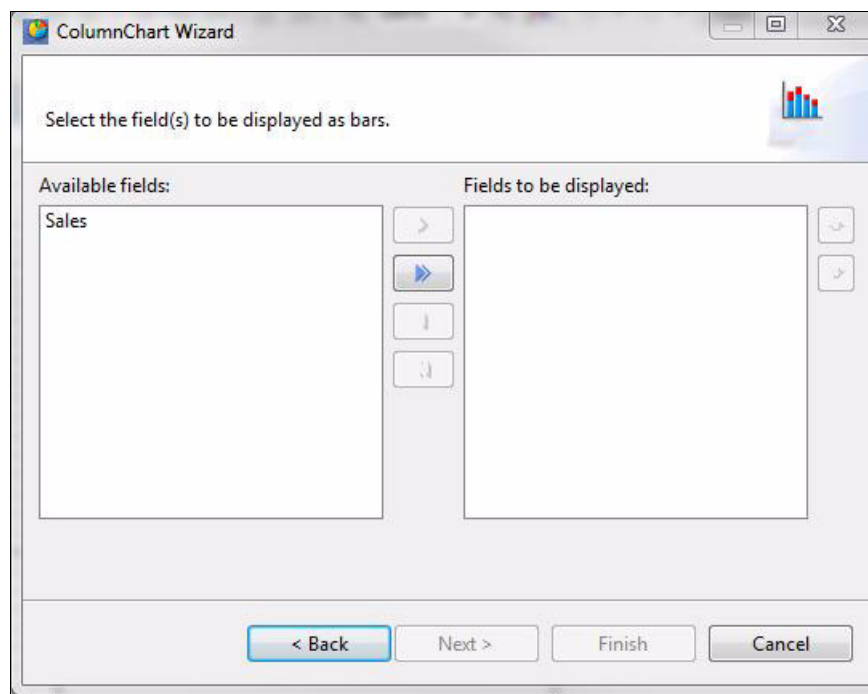


Figure 14-53 Selecting the fields to be displayed in the column chart

6. Click **Finish**.

The column chart has now been added to the dashboard. To see how the chart will appear to dashboard users, click the **Runtime** tab (Figure 14-54).

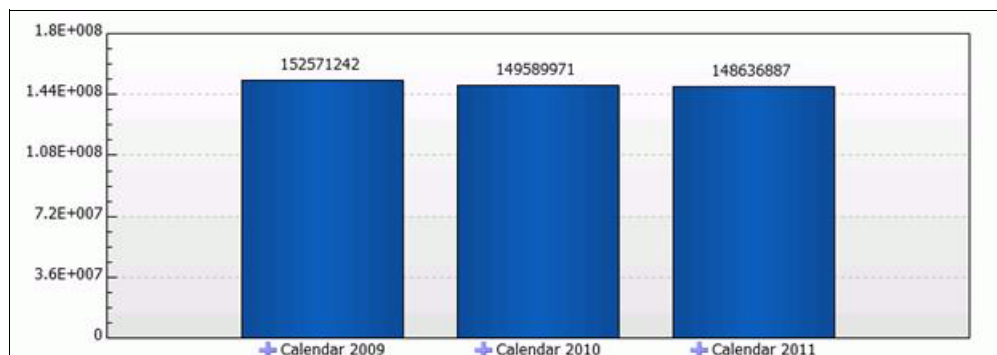


Figure 14-54 Column chart at run time

You might notice that the default formatting for numeric values on the chart needs some adjustment. In the next topic, we look at how to format these values as currency.

Formatting the numeric values on the chart as currency

Both the numeric values on the Y axis of the chart and the numeric values at the top of each column bar need to have currency formatting.

Formatting the Y-axis values as currency

To provide currency formatting for the values on the Y axis of the chart, follow these steps:

1. Click the **Design** tab at the bottom of the scene canvas to be sure you are in design mode.
2. From the Project Explorer tree, select **yAxis1** under ColumnChart1, as shown in Figure 14-55.

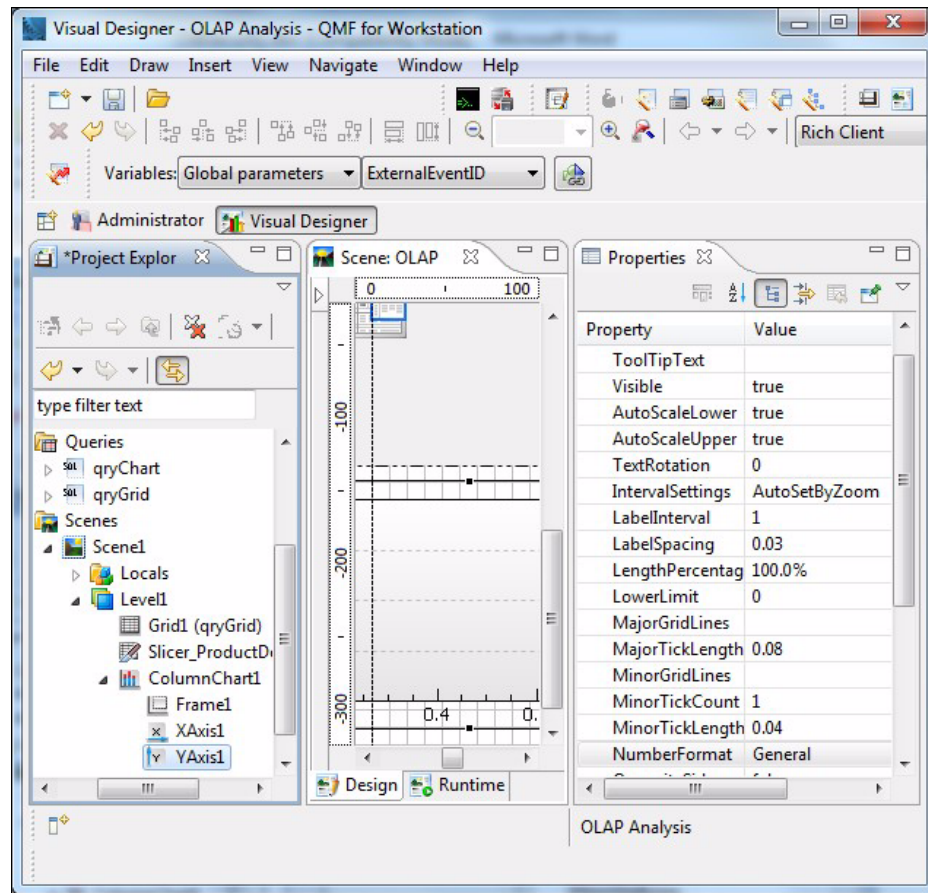


Figure 14-55 Selecting Y axis values for formatting

3. In the Properties view, change NumberFormat for yAxis1 from General to the following expression, as shown in Figure 14-56:

= "\$#,##0;(\$#,##0)"

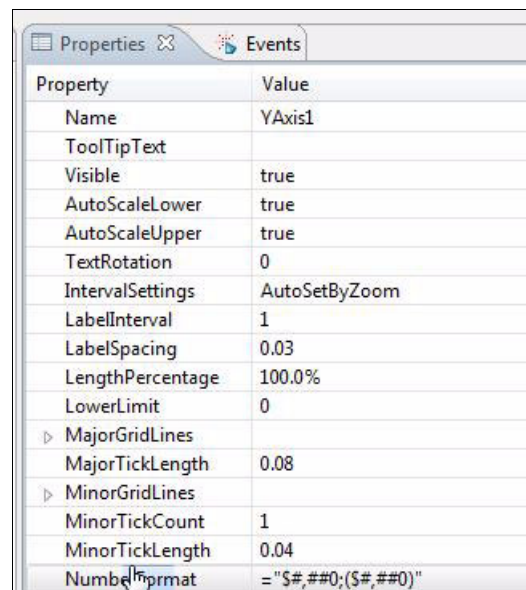


Figure 14-56 NumberFormat value for YAxis1, changed to allow currency formatting

Formatting the column bar labels as currency

To format the numeric values at the top of each column bar as currency, follow these steps:

1. Click the **Design** tab at the bottom of the scene canvas to be sure you are in design mode.
2. From the Project Explorer tree, select **Label1** at the very bottom of the tree, as shown in Figure 14-57.

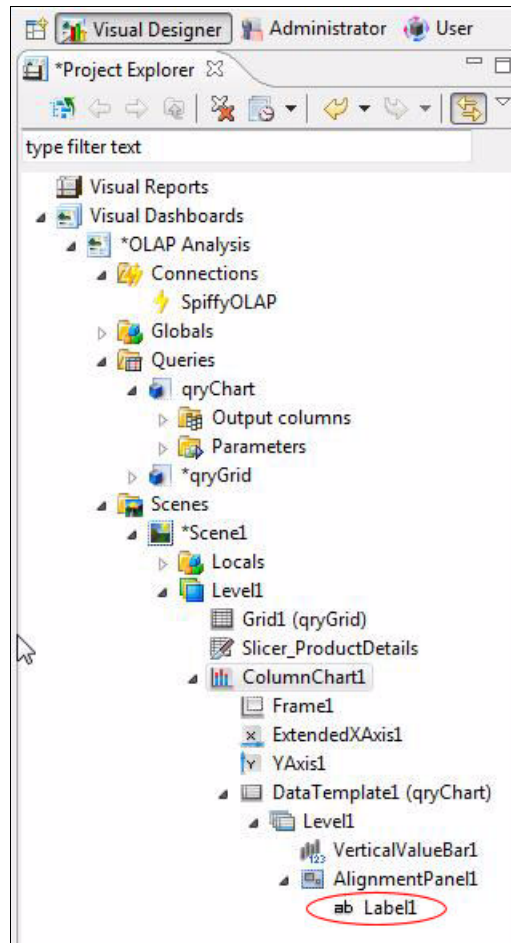


Figure 14-57 Selecting the column bar label control to format the labels as currency

3. In the Properties view for Label1, click the **Value** field for the Text property, then click the **Edit with Expression Designer** icon, highlighted in Figure 14-58.

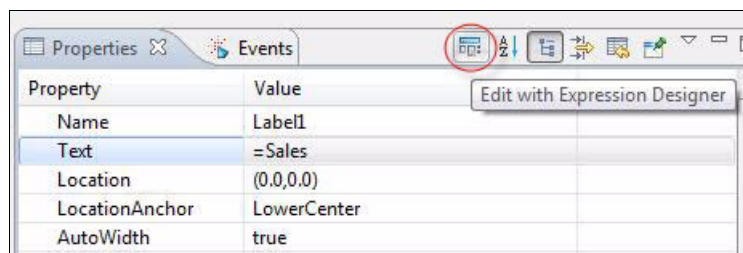


Figure 14-58 Invoking the Expression Designer

4. Highlight **Sales** and click the **Open FormatNumber Dialog** icon, as shown in Figure 14-59.

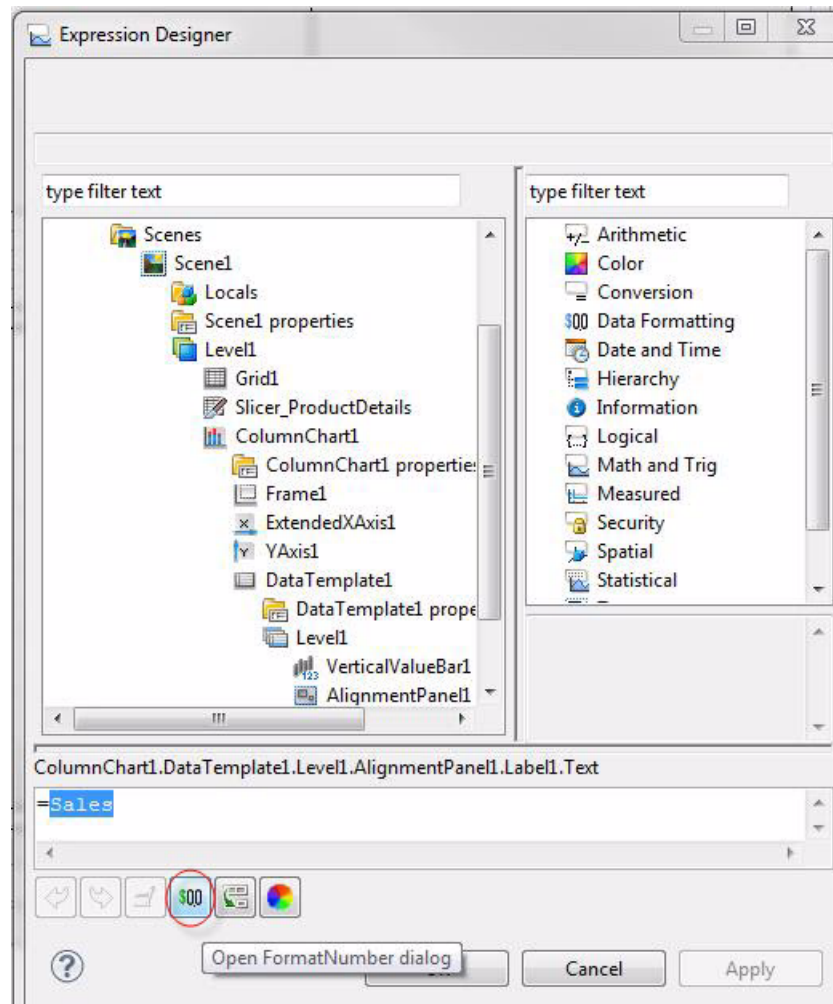


Figure 14-59 Opening the FormatNumber dialog

5. Select **Currency** from the Category list and double-click the desired currency template, as shown in Figure 14-60 and Figure 14-61.

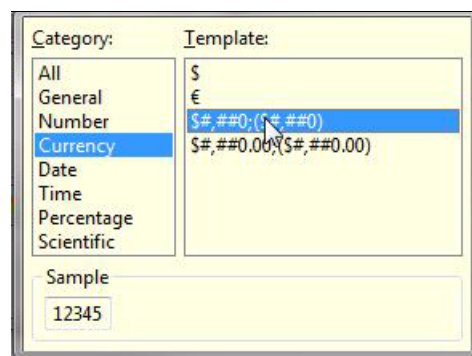


Figure 14-60 Selecting the currency template

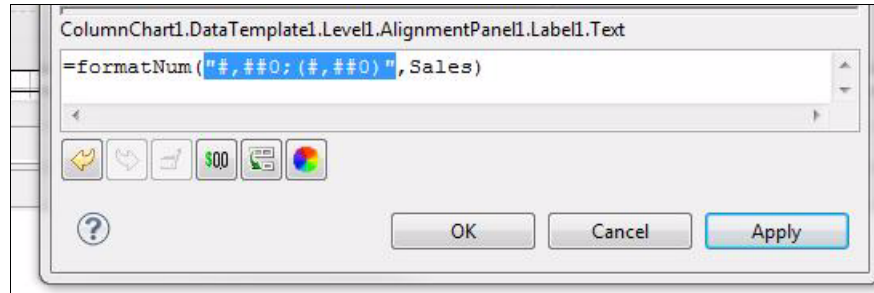


Figure 14-61 Format expression for column bar labels in the column chart

After clicking **OK**, we now see that, when the chart is displayed (as shown in Figure 14-62), the new currency formatting has been applied along the Y axis and on the column bar labels.

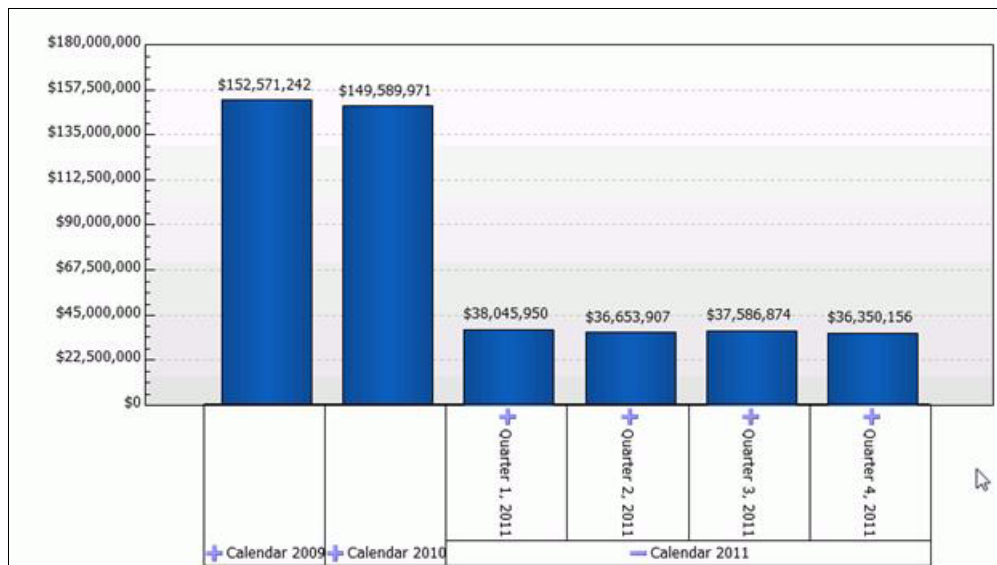


Figure 14-62 Column chart with formatted Y axis and column bar totals

Adding slicers to the column chart

The column chart needs to have two slicers associated with it:

- The first slicer allows dashboard users to filter the data in the column chart by product type (type of insurance).

This slicer, called Slicer_ProductDetails, already exists, as we created it previously for the grid. We show you in this topic how to associate this existing slicer with the column chart.

Tip: Because this slicer is also associated with the grid, any changes in the slicer will apply a filter to the data shown in both controls (the grid and the column chart).

- A second slicer, called Slicer_time, allows dashboard users to filter the data by quarter within year, and further by month within quarter.

This slicer can be created using the same steps as shown for the Slicer_ProductDetails slicer earlier in the scenario. (The ability for users to filter the data down to the month level in the dashboard is shown in Figure 14-62, where users can click the + sign for the quarter in question and drill down to the month level.)

To associate the existing Slicer_ProductDetails slicer with the column chart:

1. Click the **Design** tab at the bottom of the scene canvas to be sure you are in design mode.
2. From the Project Explorer tree, select **ExtendedXAxis1**.
3. In the Properties view, click the **Value** field for the Slicers property.
4. In the Slicers window shown in Figure 14-63, check the slicer named **Slicer_ProductDetails** and click **OK**.

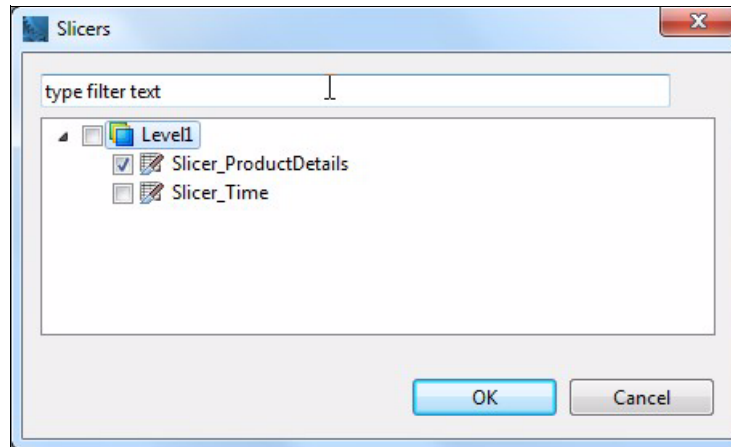


Figure 14-63 Associating the Slicer_ProductDetails slicer with the column chart

Now when we click the **Runtime** tab, we see both slicers for the column chart (Figure 14-64).

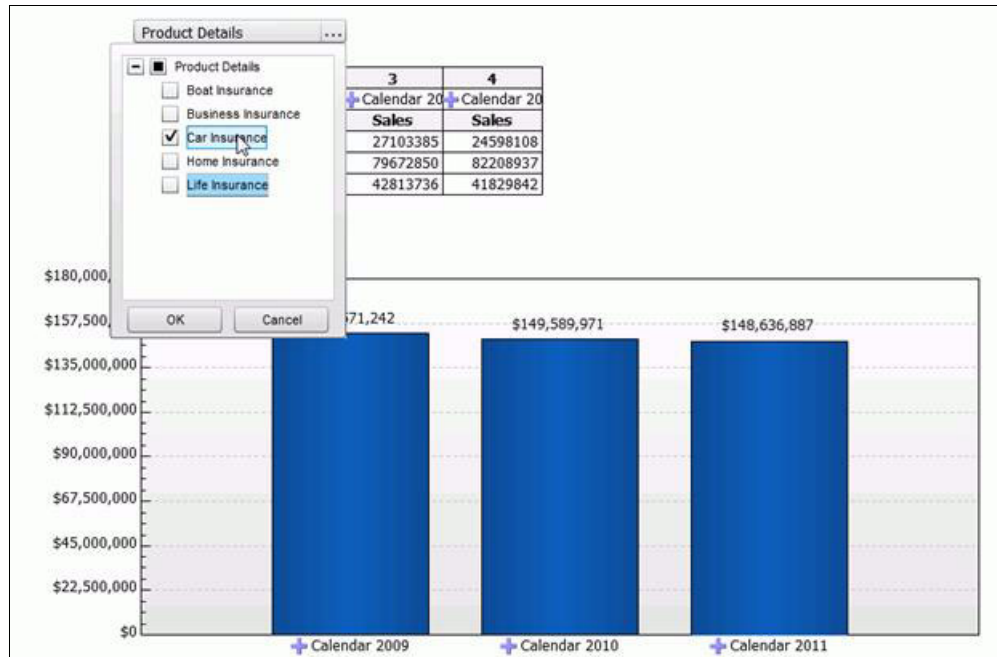


Figure 14-64 Filtering column chart details by time and product type

14.4.5 Designing the scene: Adding a geospatial map driven by selections in the column chart

The next step in designing the market analysis dashboard is to add a geospatial map such that, when dashboard users select a **Time Period** in the column chart, the map reflects the volume of sales by state, with colors used to indicate sales from high to low.

Creating the query that will populate the map

Before we can add the map object, we need to create the query that will be used to populate it. Our query for this part of the dashboard will be called `qryMap`, and it includes parameters for the date range currently selected in the column chart. Query parameters allow you to build SQL that can dynamically be changed at runtime, depending on the values passed. Using an ampersand (&) in a query denotes that the string following it is a parameter.

To create the `qryMap` query, from the Project Explorer tree, right-click **Queries** and select **Insert Query**, making sure to select **CustomerData** (our DB2 LUW data source) from the list of available connections. It is a query against the DB2 LUW data source (CustomerData), with parameters that will pass to the map the date range that is currently selected in the column chart. The finished query is shown in Figure 14-65.

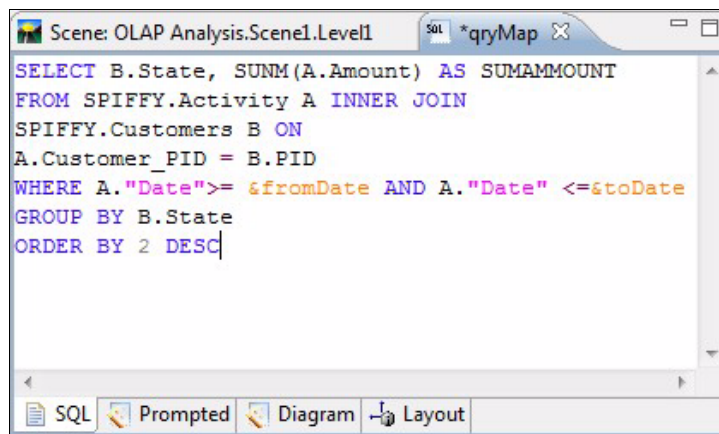


Figure 14-65 Query against the DB2 LUW data source (CustomerData)

We now have our basic aggregation of sales by state, and we join this to a table that holds state boundary data (coordinates) for each of the states. The resulting query is shown in Figure 14-66.

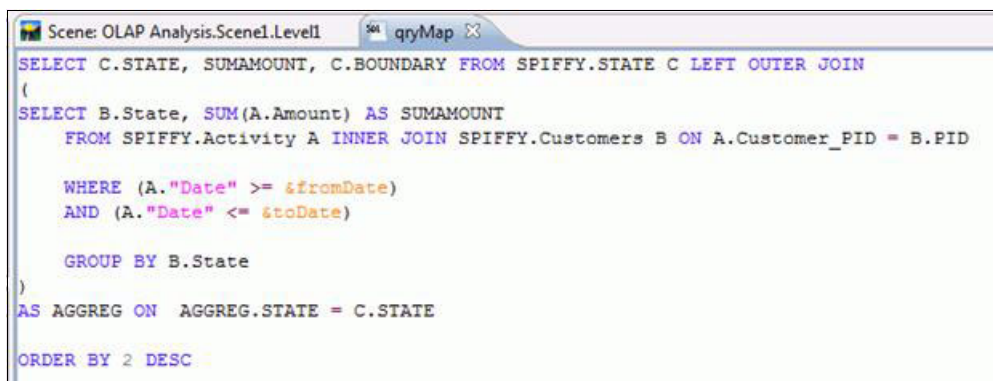


Figure 14-66 Left outer join of map boundary data with aggregate sales data

Tip: The ORDER BY clause in the query in Figure 14-66 is important, as the data returned for the state with the highest sales will be the first row and the data returned for the state with the lowest sales will be the last row. When working with query result sets in a dashboard, each row is matched with its ordinal position in the returned set. It allows us to colorize the USA map showing highest to lowest sales, because each state (row) will have its own ranking in the result set.

Adding the map object to the canvas

Now that we have the query that will populate the map, we add a linear map object to the dashboard, associating it with the qryMap query.

To insert a linear map:

1. Click the **Design** tab at the bottom of the scene canvas to ensure that the dashboard is in design mode.
2. Click inside the scene canvas and select **Insert** → **Layout** from the menu. The Layout wizard opens, as shown in Figure 14-67.

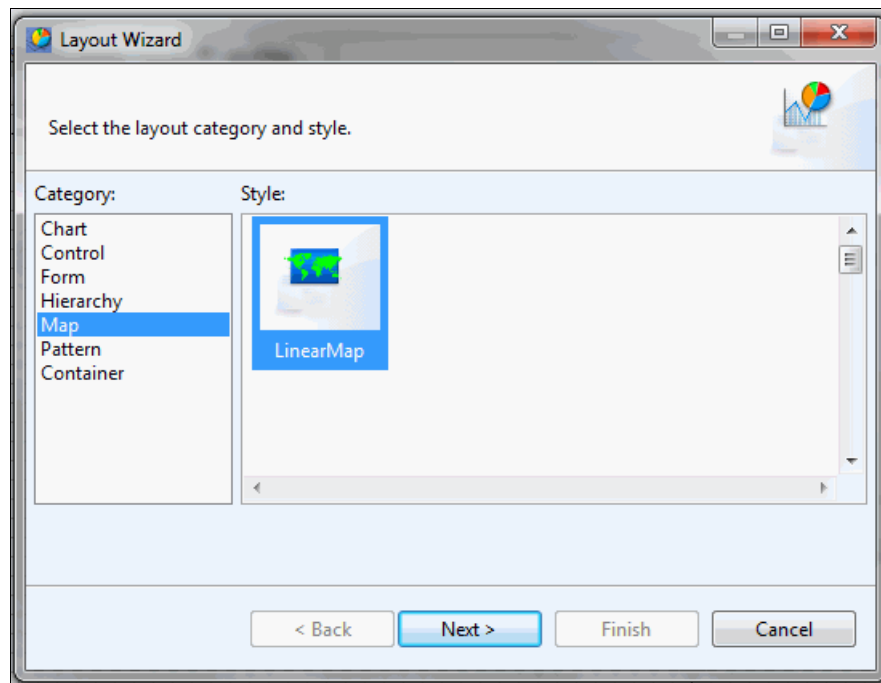


Figure 14-67 Inserting a LinearMap object

3. From the Layout wizard, select **Map** from the Category list and select **LinearMap** in the Style pane of the wizard, as shown in Figure 14-67. Click **Next**.
4. Select the query **qryMap** and click **Next**.

5. Select **Data contains closed boundaries. Plot as polygons**, as shown in Figure 14-68 and click **Next**.

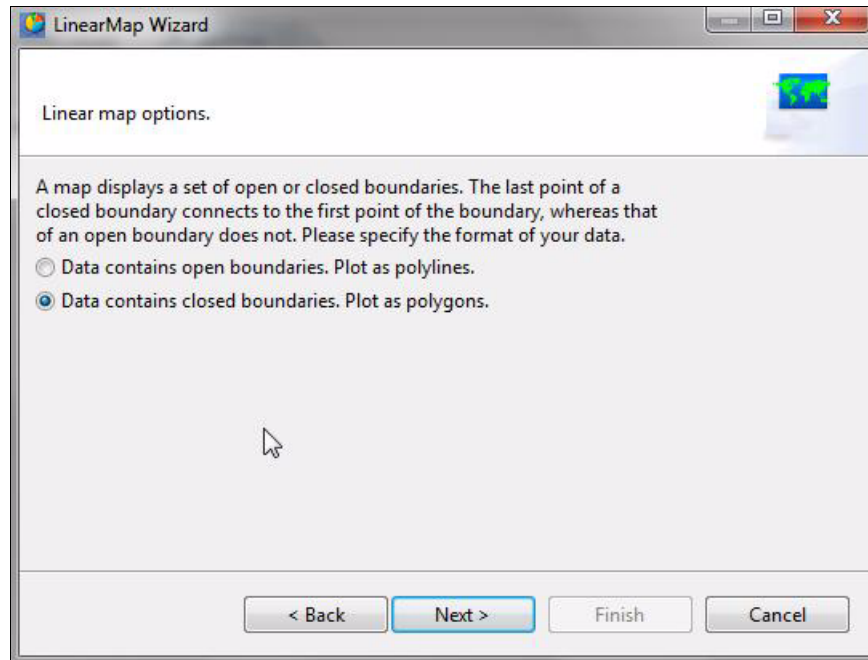


Figure 14-68 Selecting how QMF should plot the map

6. As shown in Figure 14-69, in the Boundary Field drop-down list, specify the data column in your result set that contains the map vertices. In our case, it is the BOUNDARY column.

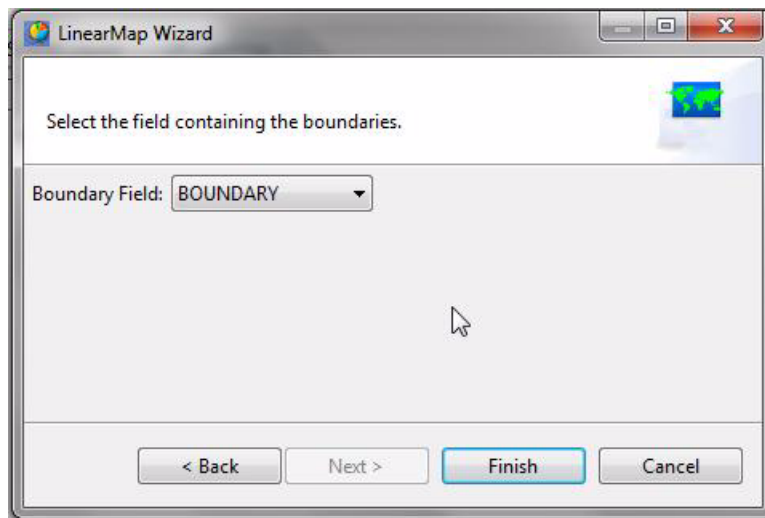


Figure 14-69 Specifying the column in the query that will provide the map vertices

7. Click **Finish** to see the resulting map object, as shown in Figure 14-70.

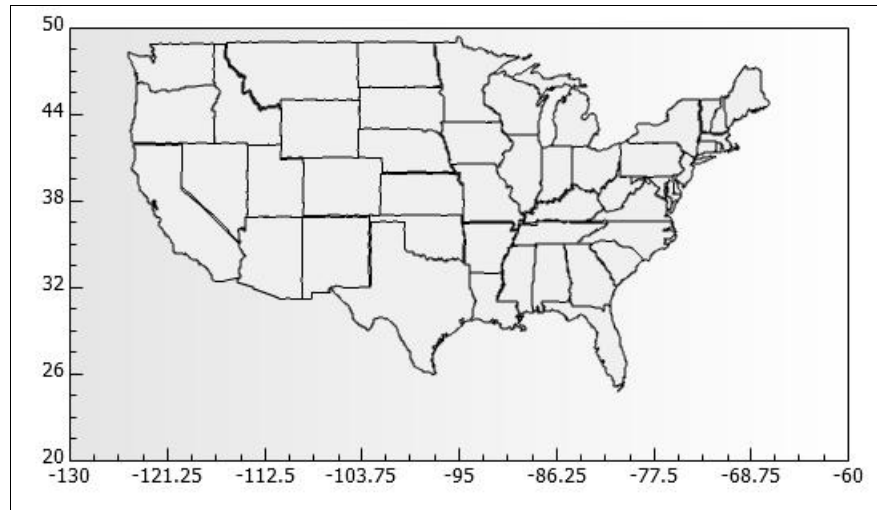


Figure 14-70 Default linear map

You can use the Expression Designer to set the color scheme for the map, resulting in a map that looks similar to the one shown in Figure 14-71.

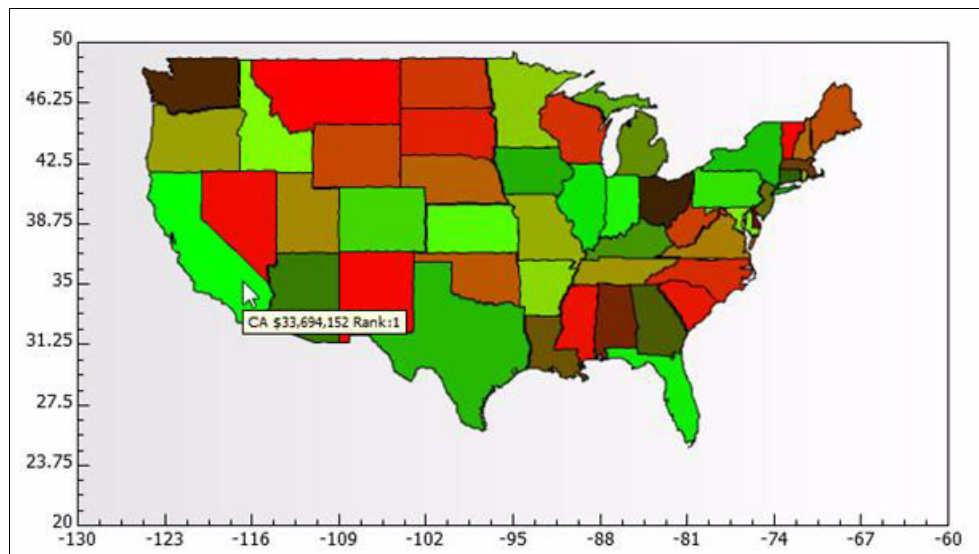


Figure 14-71 Linear map object with shading logic ranging from green for high sales to red for low sales

The tooltip and the shading are properties of the multipolygon for LinearMap1. To set the tooltip, select the multipolygon in the Project Explorer tree (MultiPolygon1 in this case), and then select the **ToolTipText** property in the Properties view, as shown in Figure 14-72.

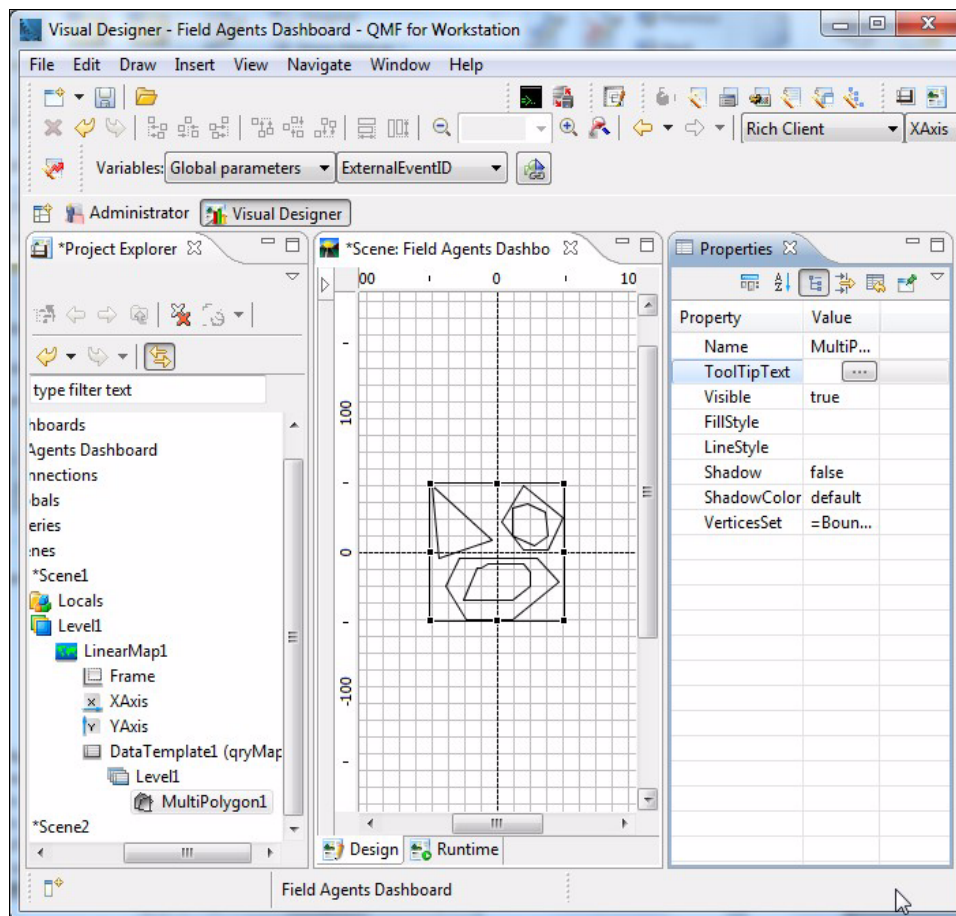


Figure 14-72 Setting the tooltip for states on the map

After ToolTipText has been selected, start the Expression Designer by clicking the **Edit with Expression Designer** icon, as shown in Figure 14-73.

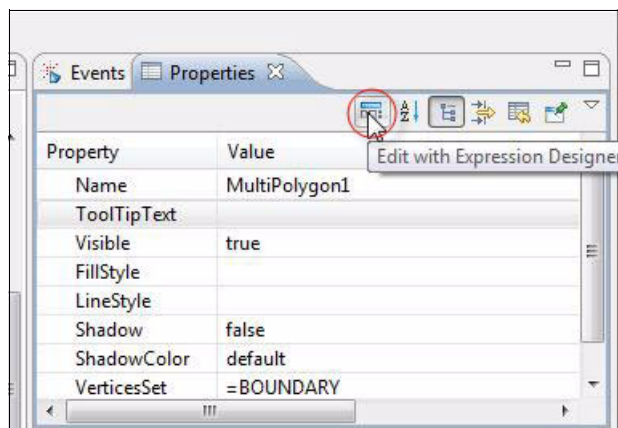


Figure 14-73 Starting the Expression Designer by clicking the Edit with Expression Designer icon

A multipolygon is drawn for each row contained in the qryMap result set, but each drawn multipolygon only knows about the result set data for its own row. So if we set ToolTipText to “=STATE” (where STATE is one of the output columns from the qryMap query shown earlier in Figure 14-66 on page 338), then STATE will have one text value, which will be the state abbreviation for the row being drawn.

Likewise, SUMAMOUNT, the value of total sales for each state, will only contain one value for each multipolygon, so the following tooltip text entry would look like “CA \$XX,XXX” for California:

```
=STATE + ' ' + formatNum("$#,##0;($#,##0)",SUMAMOUNT)
```

See Figure 14-74 for the completed ToolTipText option.

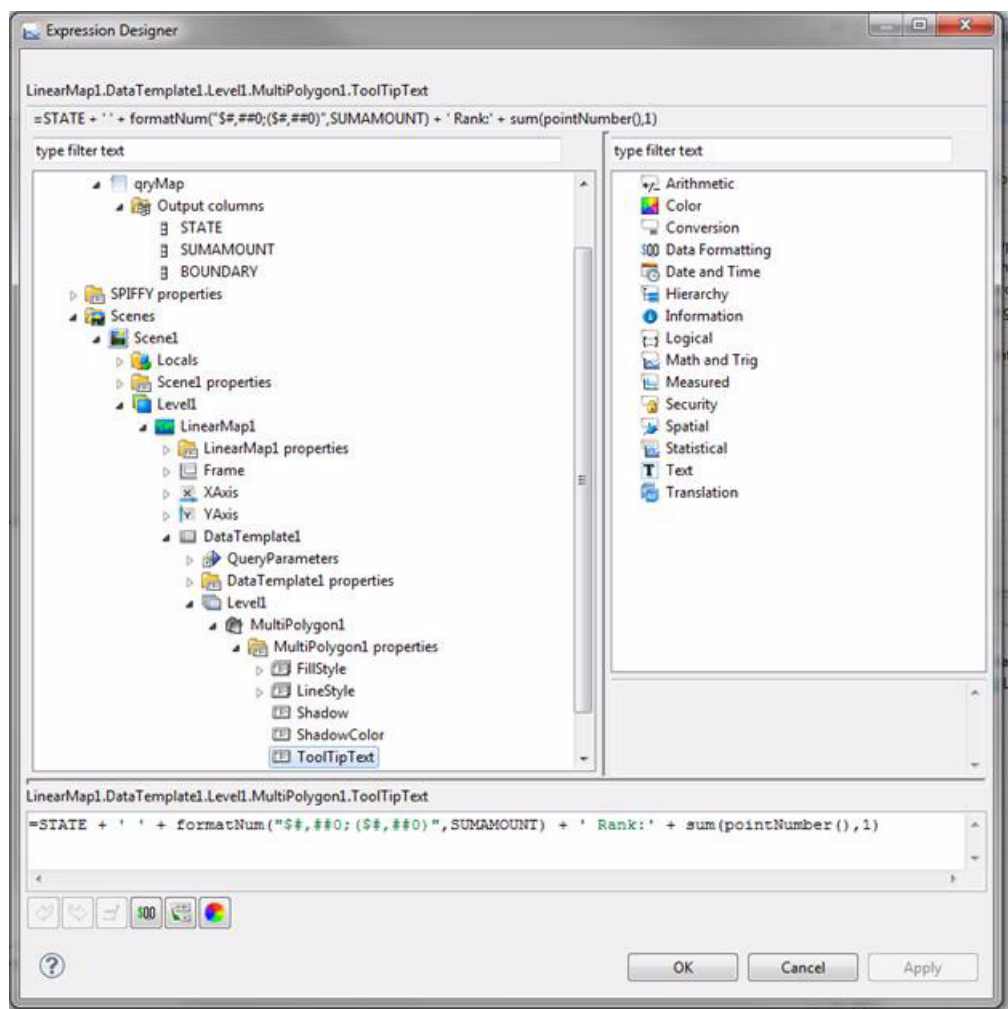


Figure 14-74 Completed tooltip text for the multipolygon object

Tip: If you cannot remember the result set column names, you can expand the known queries at the top of the Expression Designer and see the output columns for each query. Double-clicking an output column places its name in the editor pane.

Passing the selected date range from the column chart to the map

Our qryMap query contains two parameters that we can use to pass the selected date range when a column in the OLAP column chart is clicked. The following methods can be used for setting the values for the query parameters; whether we need to reuse the values set in the parameters again will determine the best approach to take:

- ▶ The query parameters can be set to “listen” for a change in the value of a control. (Notice that, when a query parameter changes value, the query is resubmitted.)
- ▶ An event from a control (such as a mouse click) can be used to push a value to the query parameter.
- ▶ An event from a control (such as a mouse click) can be used to place values into scene parameters or Spiffy parameters. The query parameter can be pointed to the scene parameter so that, when the scene parameter changes value, the query runs.

The third approach allows us to save the date ranges in scene parameters so that, when a state is selected and we populate a table with customers for the state, we can use the same date range that was used in the qryMap query.

To create two scene parameters that will hold the start and end date values for the selected chart column:

1. In design mode, right-click **Parameters** in the Project Explorer tree and select **Insert Parameter**, as shown in Figure 14-75.

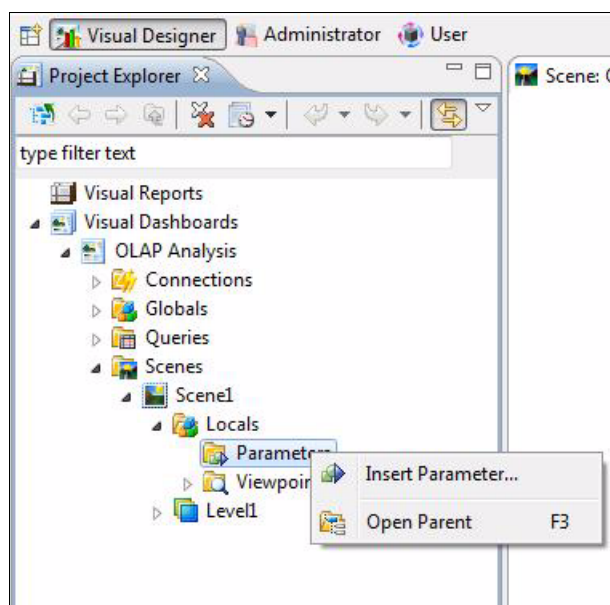


Figure 14-75 Inserting a scene parameter

2. Give the new parameter a meaningful name. In our example, our name starts with “s_” so that we know that it is a scene parameter, not a Spiffy parameter.
3. Because this parameter is going to be a date, we set the data type to DateTime, as shown in Insert scene parameter (Figure 14-76).

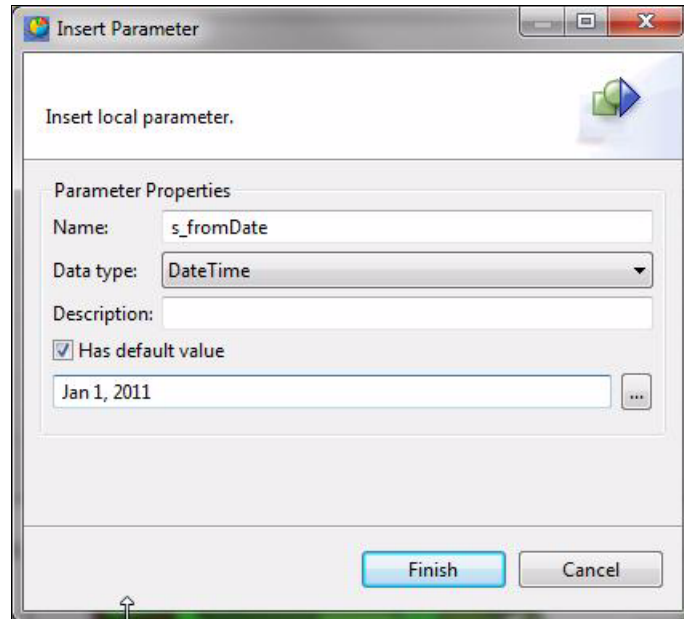


Figure 14-76 Insert scene parameter

4. Click **Finish**, then in the same manner create a second parameter called s_endDate, to hold the end date.

5. Place the dates from the selected column of the OLAP column chart within the scene parameters by adding a click event on the column chart in the following manner:
 - a. In design mode, select **VerticalValueBar1** under ColumnChart1, then double-click the click event (Figure 14-77).

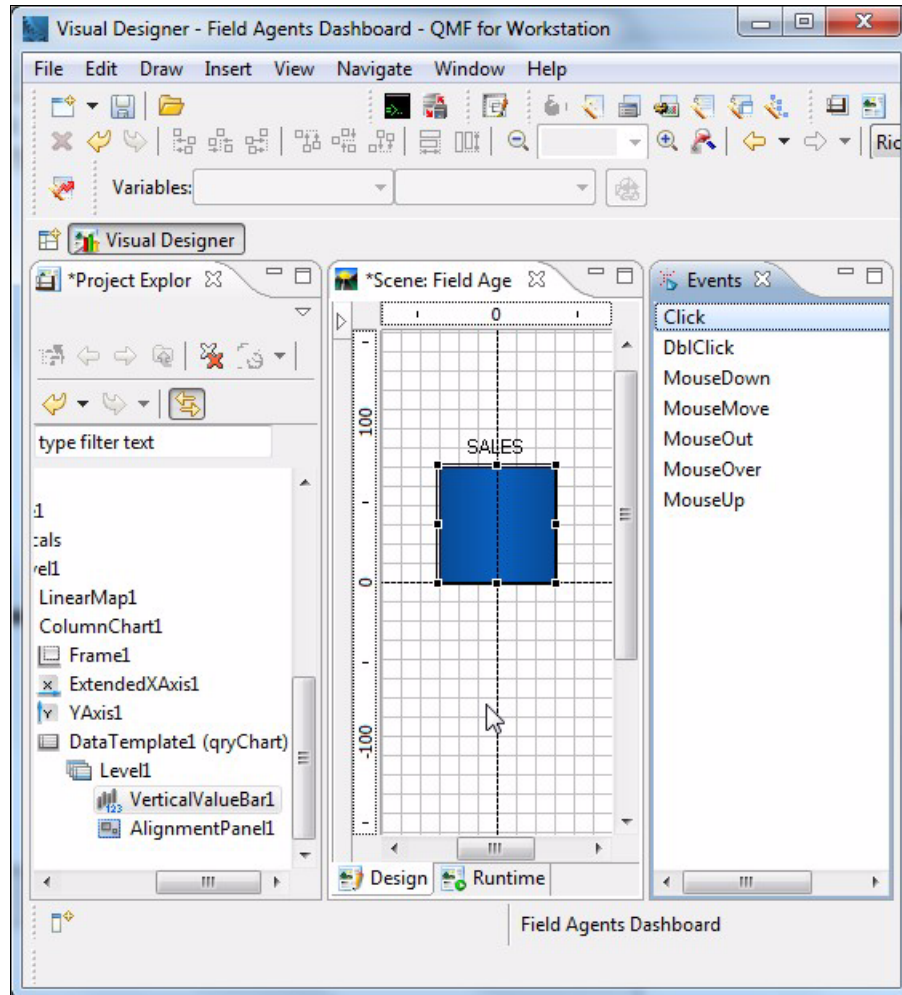


Figure 14-77 Adding a click event to a column

- b. Click the **Add New Action** icon (Figure 14-78).

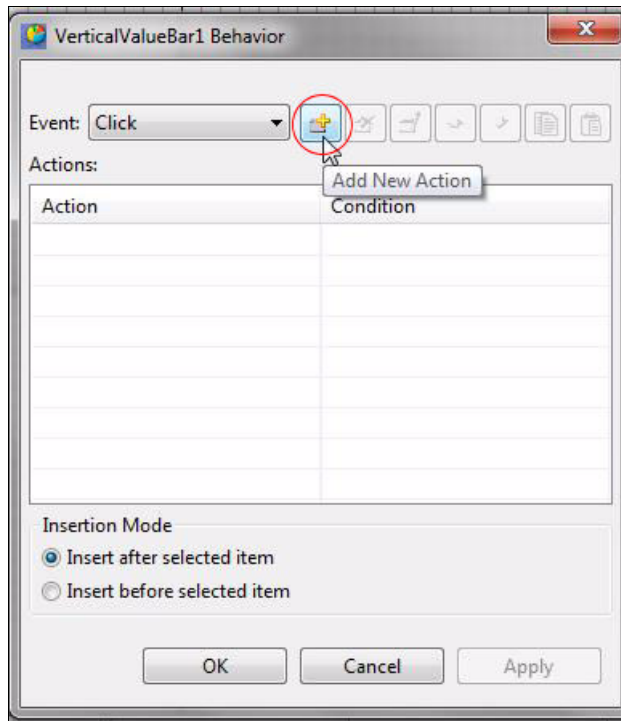


Figure 14-78 Adding a new click event

- c. In the Add New Action window, select **Set values** (Figure 14-79). This action configures the click event so that the click in the column chart sets values in the map.

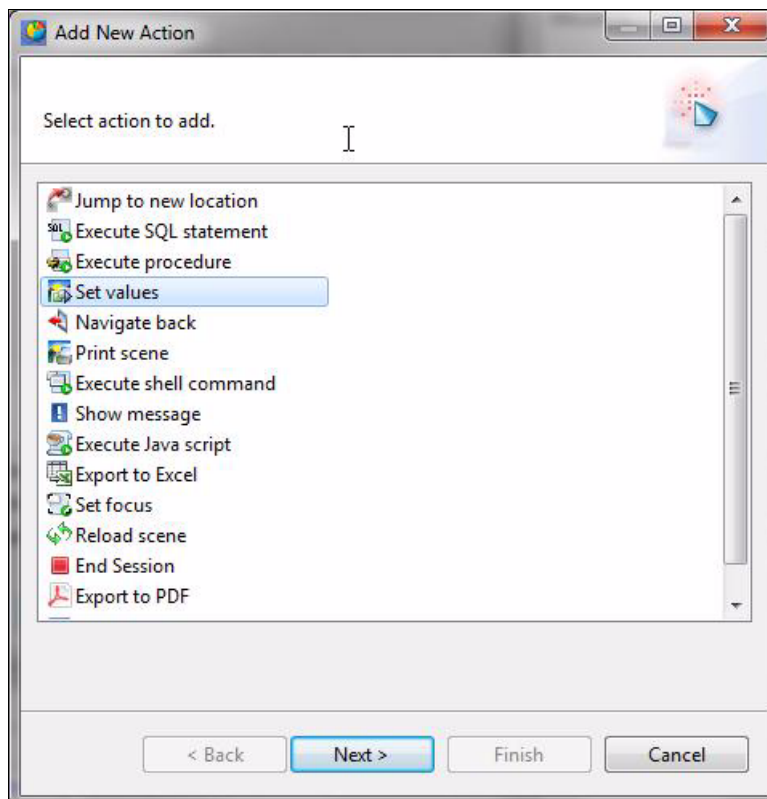


Figure 14-79 Set values action; the click in the column chart sets values in the map

- d. Click the **Add New Entry** icon (Figure 14-80).

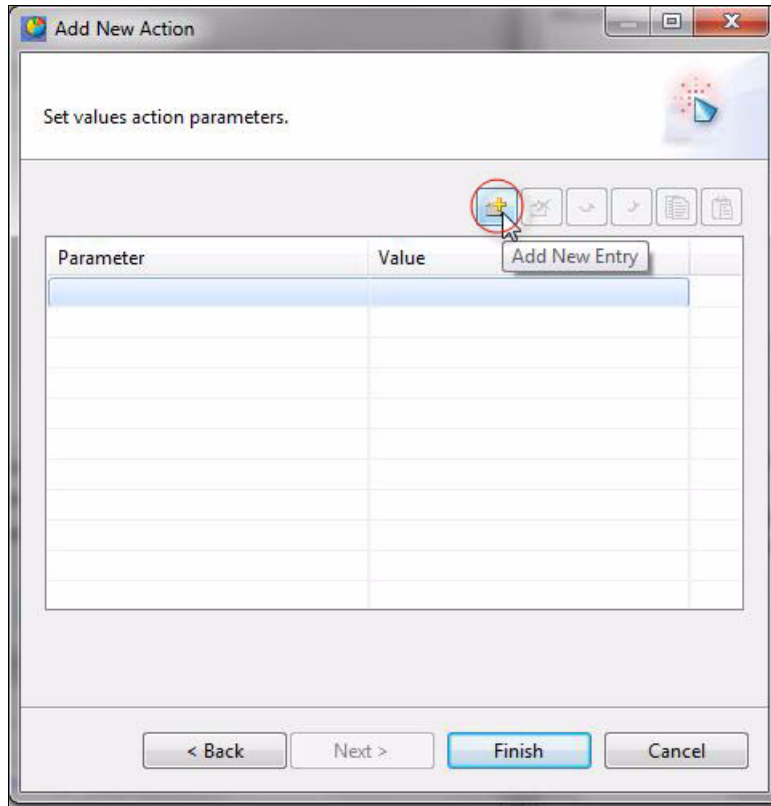


Figure 14-80 Add New Entry icon

- e. In the Expression Designer, double-click **s_fromDate** as the item that is having its value set (Figure 14-81).

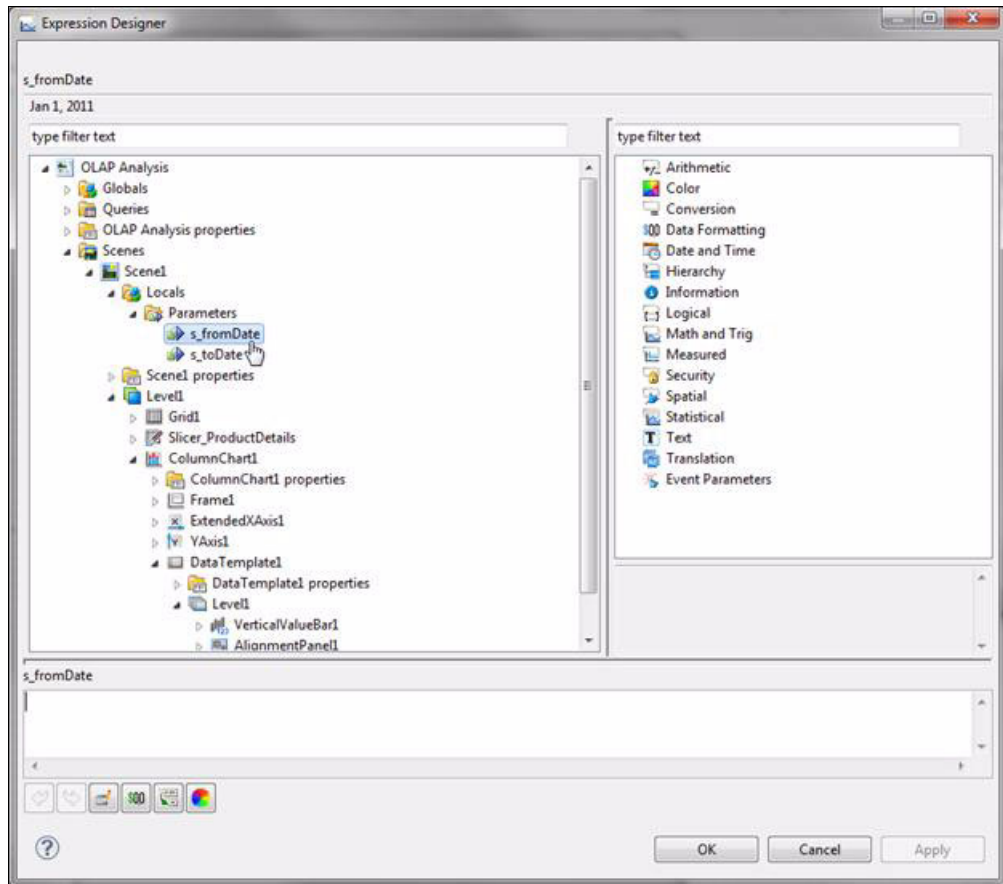


Figure 14-81 Selecting *s_fromDate* to set its value

- f. Specify the following built-in function to get the start date from the selected column (Figure 14-82), then click **OK**.

```
=getDimensionStartDate( VerticalLabelBar1.Label )
```

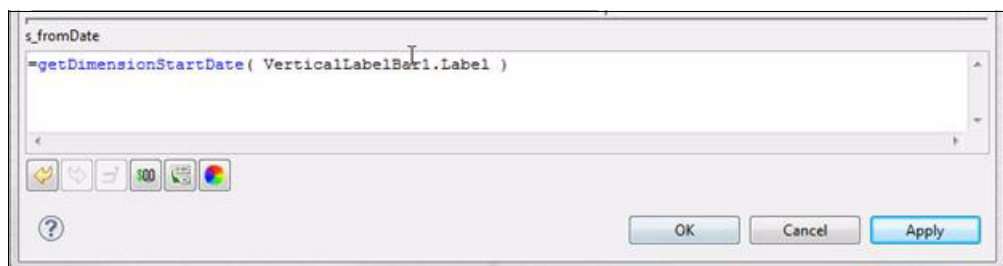


Figure 14-82 Using the *getDimensionStartDate()* function

Now follow the same steps to populate the *s_toDate* scene parameter, but use the following built-in function instead:

```
=getDimensionEndDate( VerticalLabelBar1.Label )
```

6. Link the scene parameters to the qryMap query in the following way:
 - a. In design mode, select **qryMap** from the Project Explorer tree, which then displays the query parameters in the Properties view, as shown in Figure 14-83.

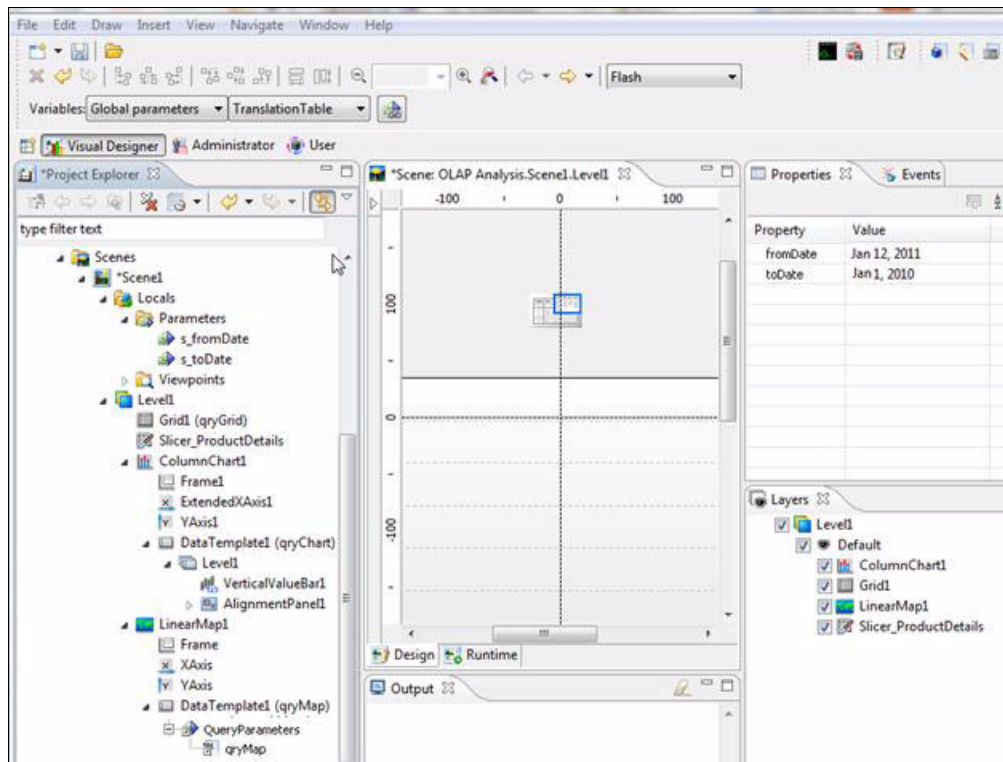


Figure 14-83 Populating the Properties view with the query parameters from the qryMap query

- b. In the Value field in the Properties view, enter the scene parameter names, as shown in Figure 14-84.

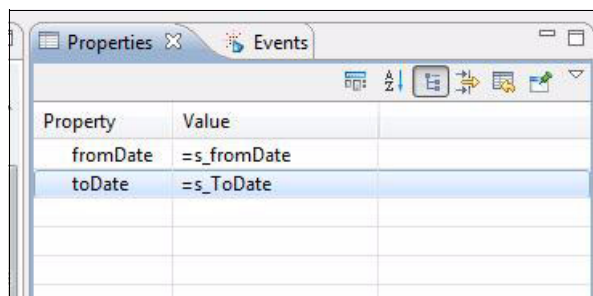


Figure 14-84 Linking the qryMap query parameters to the scene parameters just defined

Now, when the dashboard user clicks a column in the column chart (the column for Quarter 2, 2011, for example), the linear map is automatically updated to reflect the selected date range, as shown in Figure 14-85.

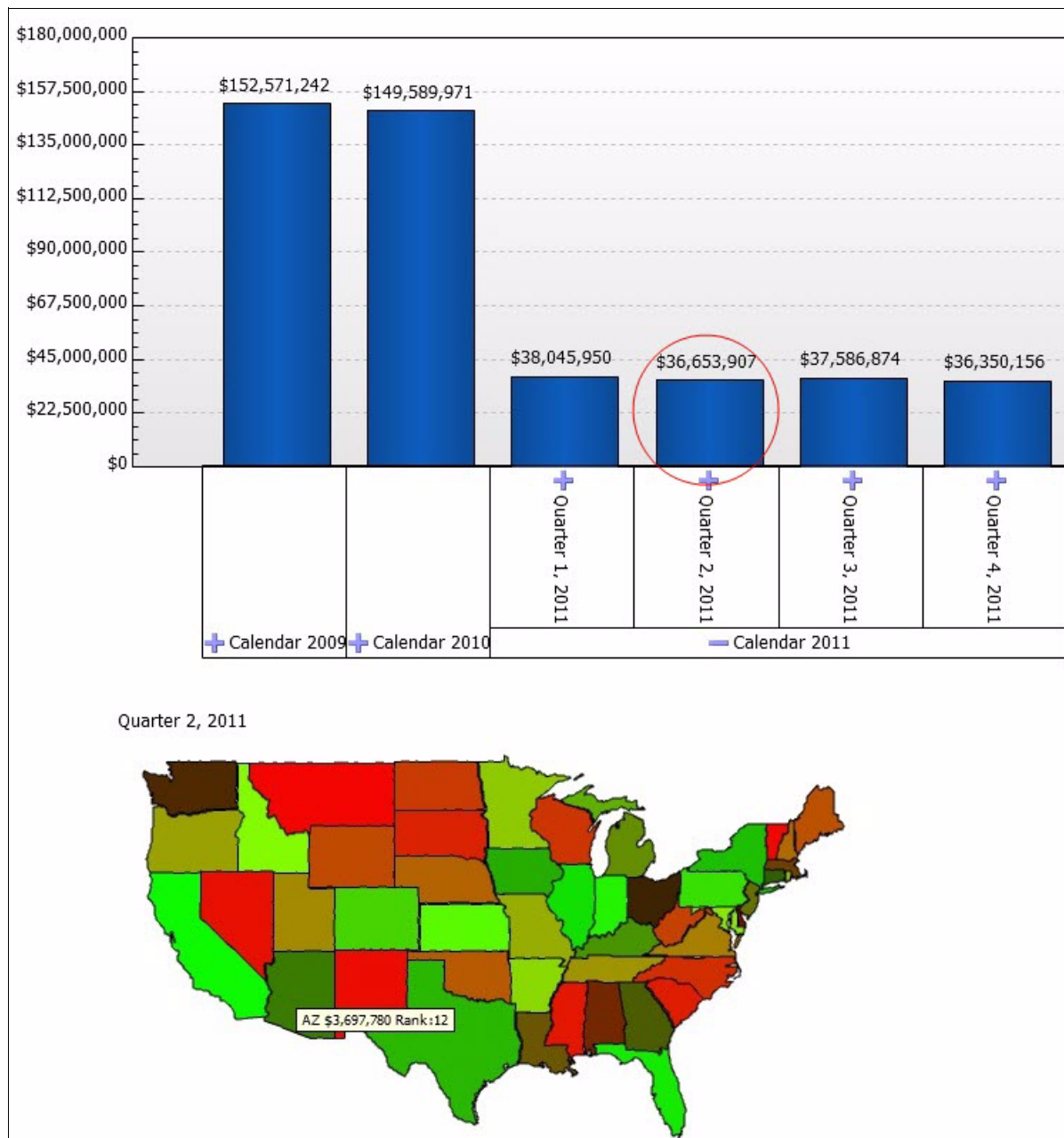


Figure 14-85 When Quarter 2, 2011 is selected, the linear map is automatically updated to show data for that quarter

14.4.6 Designing the scene: Adding table driven by state selected on the map

Spiffy's market analysts want to be able to drill deeper into the data directly from the map, down to the underlying customer records for each state. Thus, we need to add to the dashboard a table that will display Spiffy customer records for whichever state is currently selected on the map.

Creating the query that will populate the table

Before we can add the table object, we need to create the query that will be used to populate it, which we call qryTable. To populate the table, our qryTable query uses the following data:

- ▶ The date range selected by the dashboard user in the column chart. This range is stored in scene parameters s_fromDate and s_toDate, defined earlier.
- ▶ The state selected by the dashboard user in the linear map.

The qryTable query will join four tables from DB2 LUW. We use the query diagram view to create these joins in the following way:

1. In design mode, right-click **Queries** in the Project Explorer tree and select **Insert Query**.
2. Change the query name to qryTable, making sure the connection is for CustomerData (the connection to our DB2 LUW data source), then click **Finish**.
3. Click the **Diagram** tab, shown in Figure 14-86, then right-click in the query diagram area.

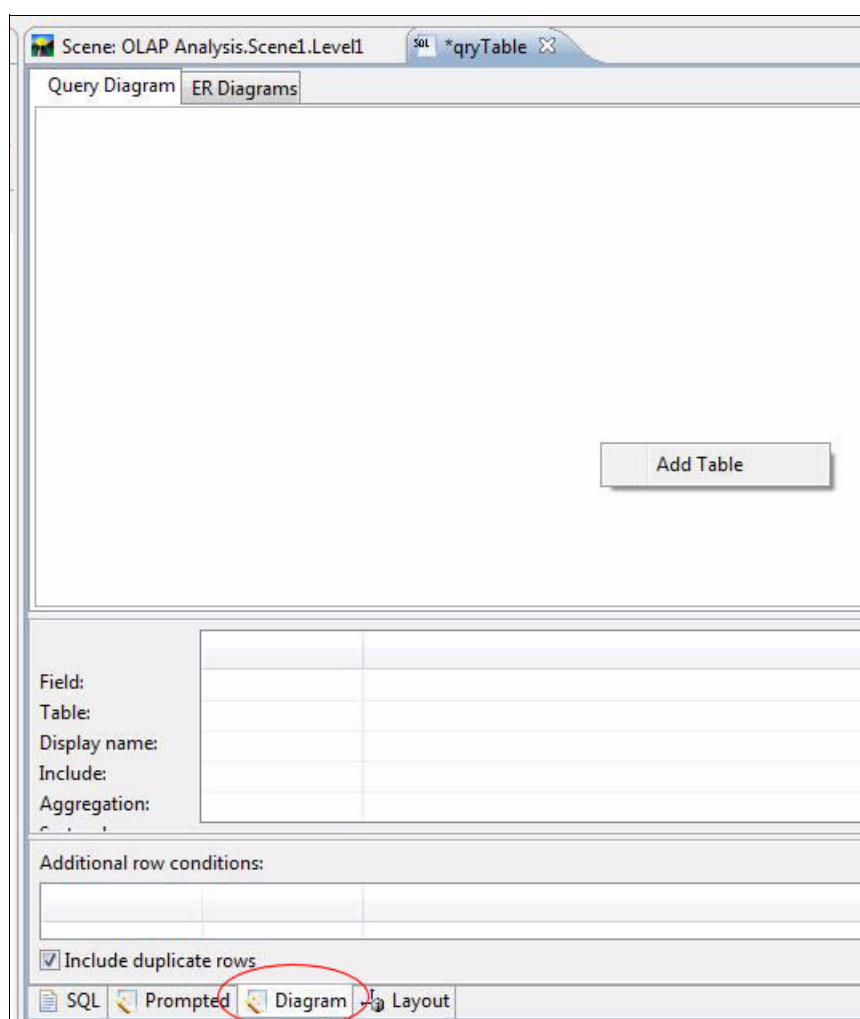


Figure 14-86 Creating a query with the query diagram view

4. Specify the tables to be used for the query.

Because we are adding four tables, we load a list of tables with the appropriate schema, then select the four we need. SPIFFY is the table owner in our example, so we enter **SPIFFY** in the Table owner field, then click **Add From List**, shown in Figure 14-87.

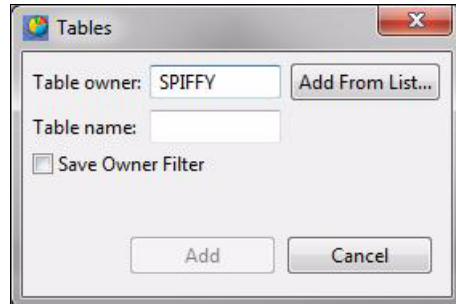


Figure 14-87 Adding tables from a list in the query diagram view

A list of table names that meet the criteria is displayed. To select four table names, hold down the Ctrl key and click each table name, as shown in Figure 14-88. You can also individually select each table and click **Add**.

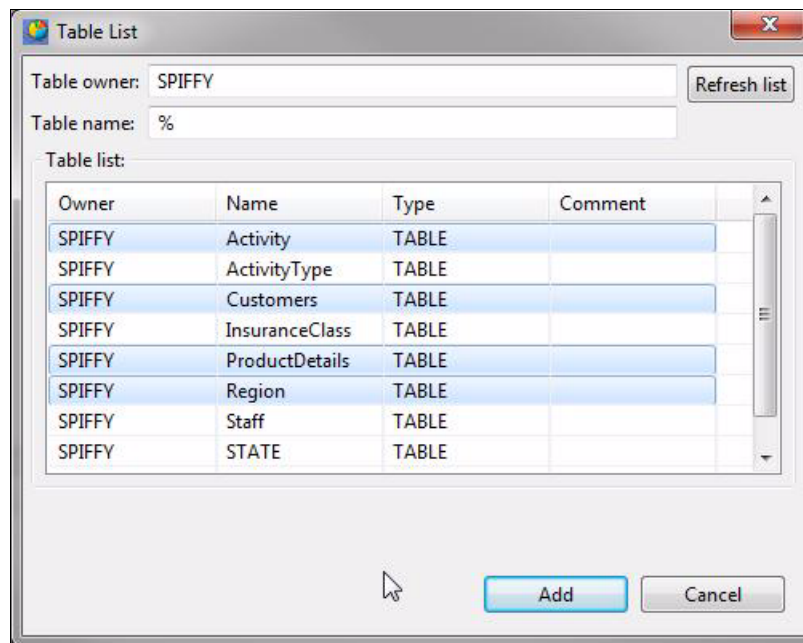


Figure 14-88 Adding tables from a list

When the required tables have been added, click **Close**. (The Cancel button changes to Close when at least one table has been added.)

5. Define the join relationships between the tables.

To define the table relationships, you can drag and drop the join columns between the tables, or you can switch to the SQL view and manually edit the SQL for the joins. We drag and drop the joins as shown in Figure 14-89.

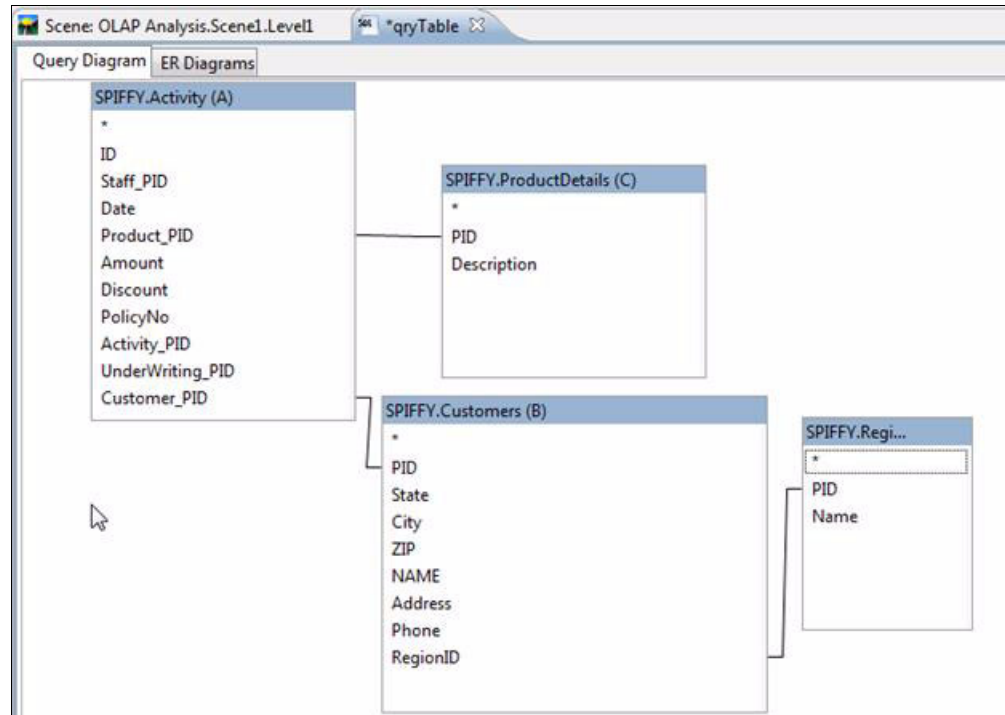


Figure 14-89 Defining table joins in the query diagram view

By default, all columns are selected, so we select only the columns we are interested in. We do this by dragging a column from the diagram and dropping it on the column grid (Figure 14-90).

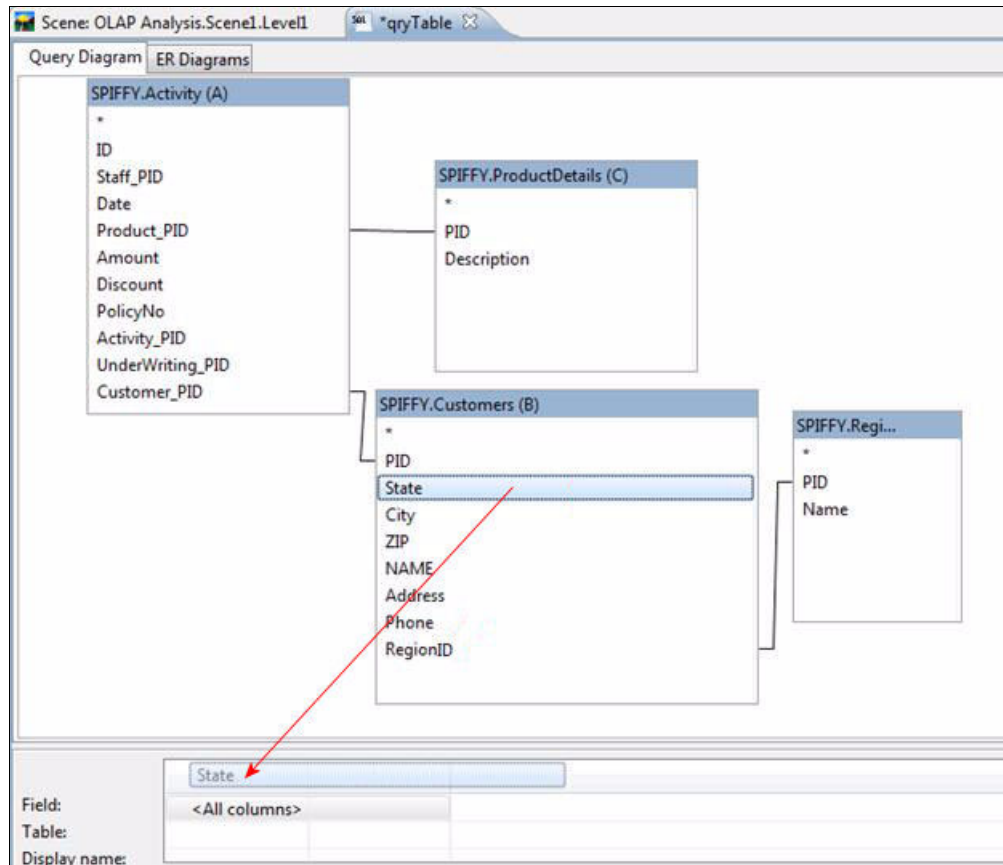


Figure 14-90 Selecting columns to return

6. If necessary, change the column names that will be displayed.

We have two Name columns, so we change the display name of the Region.Name column to Region by typing the new name in the Display name field, under the column we want to rename, as shown in Figure 14-91.

Field:	Name	State	City	NAME	Description	PolicyNo	Amount
Table:	SPIFFY.Region (D)	SPIFFY.Customers (B)	SPIFFY.Customers (B)	SPIFFY.Customers (B)	SPIFFY.ProductDetails (C)	SPIFFY.Activity (A)	SPIFFY.Activity (A)
Display name:	Region						
Include:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Aggregation:							

Figure 14-91 Changing the name displayed for a column

7. Add parameters to the query to pass in the date range selected in the column chart and the state selected in the linear map.

Two of these qryTable query parameters are the same date parameters we added to the qryMap query, and the third pulls the state that is currently selected in the map. The finished query is shown in Figure 14-92.

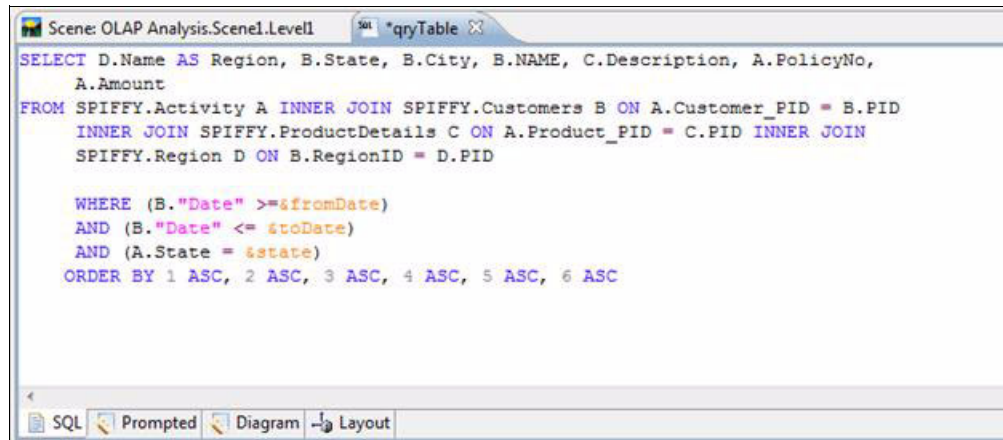


Figure 14-92 Adding to the qryTable query parameters that specify the date range and the state

We defined the data type of the query parameters for the qryMap query earlier, and we now need to do the same for the qryTable query. The data type indicates to the query engine how to format the data passed to it.

As with the qryMap query, we define the fromDate and toDate parameters with the Datetime data type. We define the state parameter as text, which tells the query engine to place single quotes around the value passed in.

To define the data types for each parameter, from the Project Explorer tree, expand the qryTable query and double-click each parameter to set its data type (Figure 14-93).

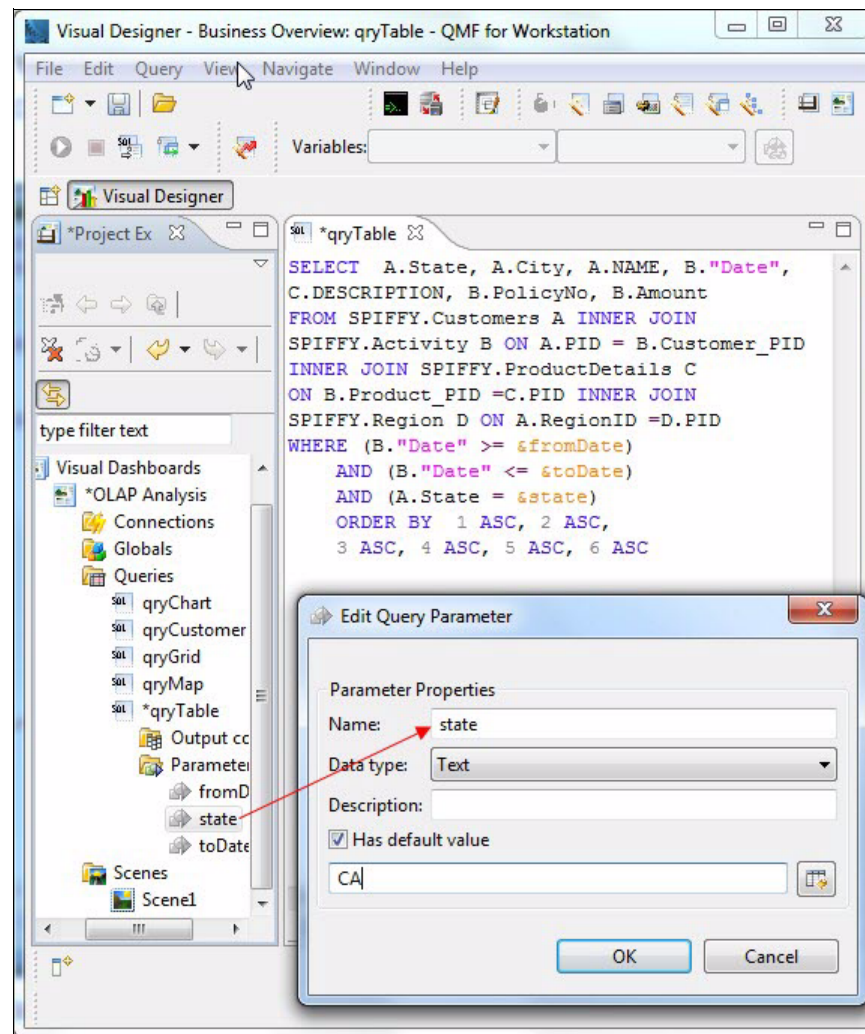


Figure 14-93 Defining the data type for the “state” query parameter

Adding the table object to the canvas

After the query that will populate the table of customer names is defined, we can add the table object to the dashboard and link both the selected state from the linear map and the selected date range from the OLAP column chart.

To add the table object to the dashboard:

1. In design mode, double-click the **Table** object in the Layouts palette of the Palette view. The Table wizard opens.
2. Accept the default data source by clicking **Next**.

3. Select the **qryTable** query, as shown in Figure 14-94, and click **Next**.

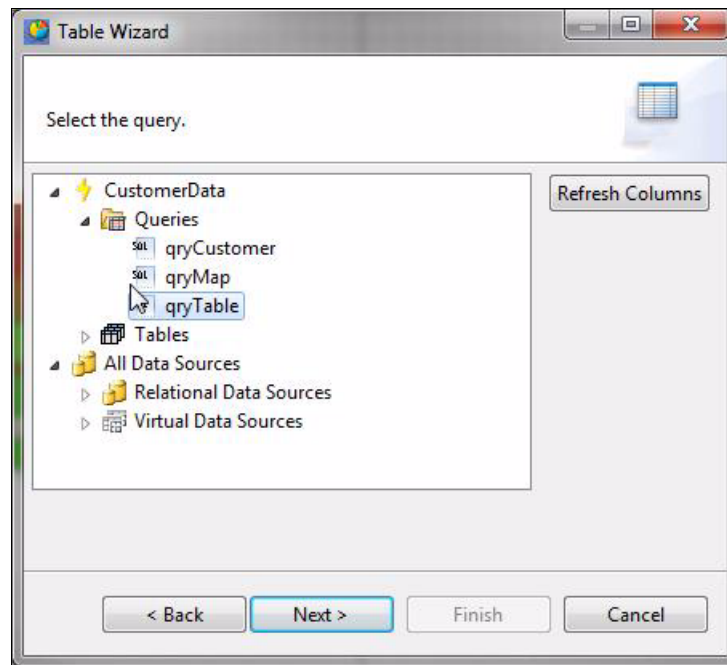


Figure 14-94 Selecting qryTable as the query that drives the table

4. Select all columns in the query, as shown in Figure 14-95, and click **Next**, then **Next** again on the following page of the wizard.

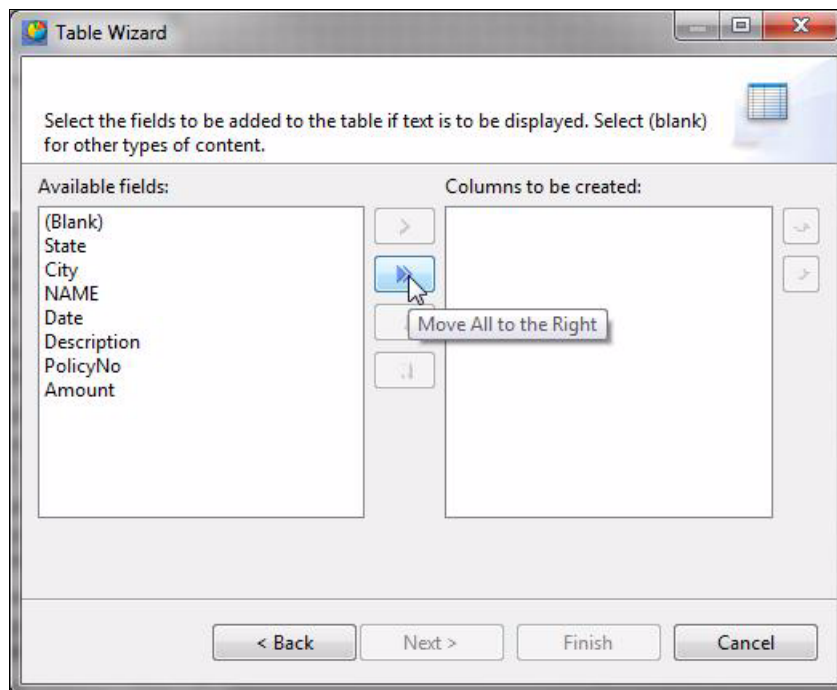


Figure 14-95 Selecting all columns in a query in the Table wizard

5. In the paging options dialog of the Table wizard shown in Figure 14-96, check the **Enable paging** check box. Set the rows per page to 15 and check the **Add navigation bar** check box.

Being able to set paging options is beneficial when you do not know how many rows to expect back in a result set. Rather than displaying a table that might possibly be many rows long, paging allows you to display a fixed number of rows to the dashboard user, with navigation controls to move between each page.

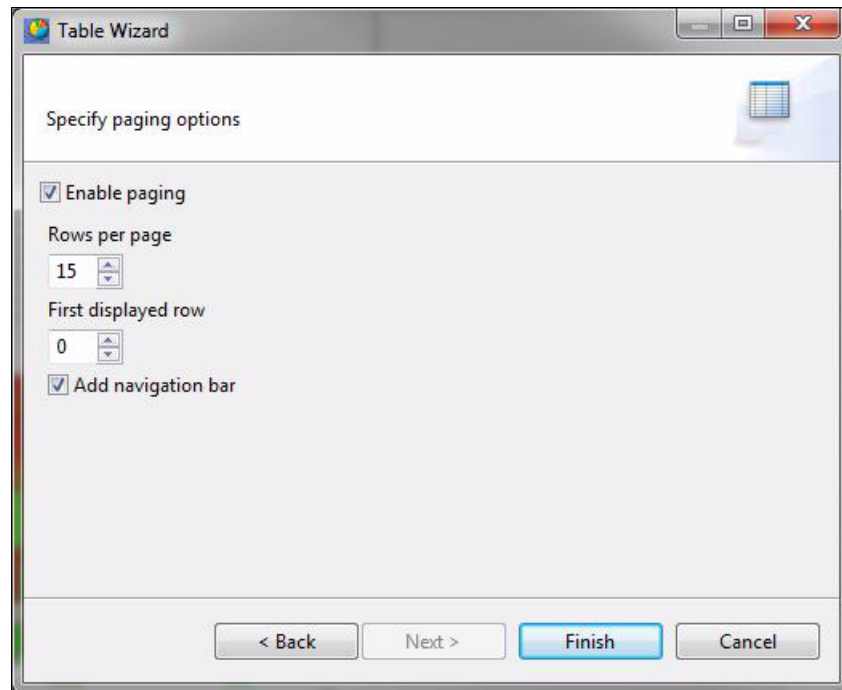


Figure 14-96 Specifying paging options for the table

Hiding the table until a state on the map is selected

The table that we just added to the dashboard is populated when the dashboard user clicks a state on the map. Because we really only want the table to be displayed once a state has been selected, and otherwise remain invisible, we want the click event on the map to trigger the table's visibility. Thus, we set the table's Visible property to 'false' now and then define its visibility when we define the click event later in the process.

To change the Visible property to “false”, click the **Value** field for the Visible property in the Properties view and select **false**, as shown in Figure 14-97.

Tip: When you set a control's visibility to false, the query associated with the control does not execute. Conversely, when a control's visibility is set to true, the query associated with the control will run.

Thus, when you set the visibility to false, be sure that the visibility is defined by an event, such as the click event that we show later in the design process.

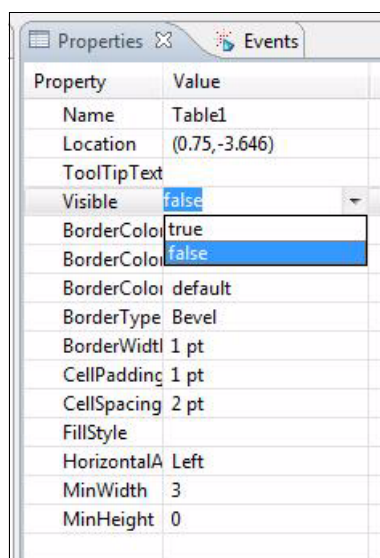


Figure 14-97 Setting table Visible property to false so visibility can be controlled by click event on map

Defining a click event to link the table to values selected in the column chart and map

The query parameter values passed from the column chart to the map were a date range. For the table, we define a click event on the map to push down values for the parameters defined in qryTable, the query that populates the table.

To define this click event:

1. In design mode, from the Project Explorer tree, expand **LinearMap1** and select **Multipolygon1**, as shown in Figure 14-98.

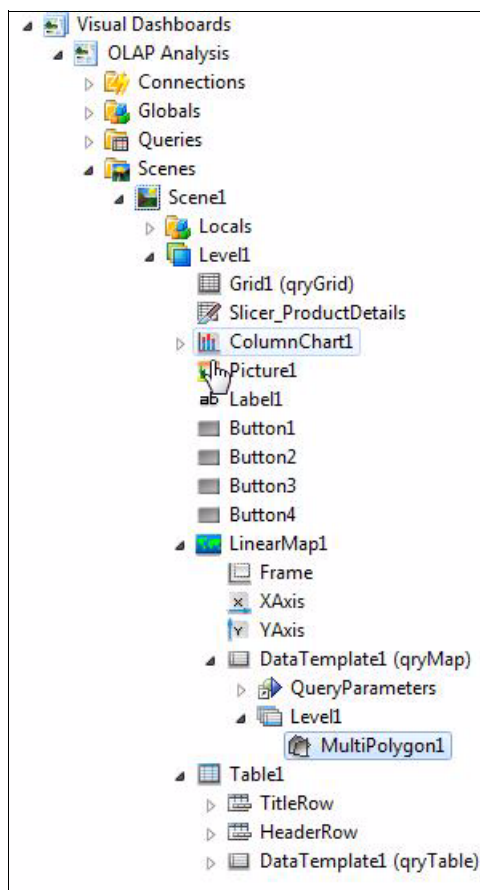


Figure 14-98 Selecting the multipolygon for the linear map

2. Open the Events view (the next tab over from the Properties tab at the right of the interface), as we need to add an action (set the qryTable parameter values) when a state on the map is clicked.
3. In the list of events, double-click **Click** to open the behavior dialog for multipolygon1, as shown in Figure 14-99.

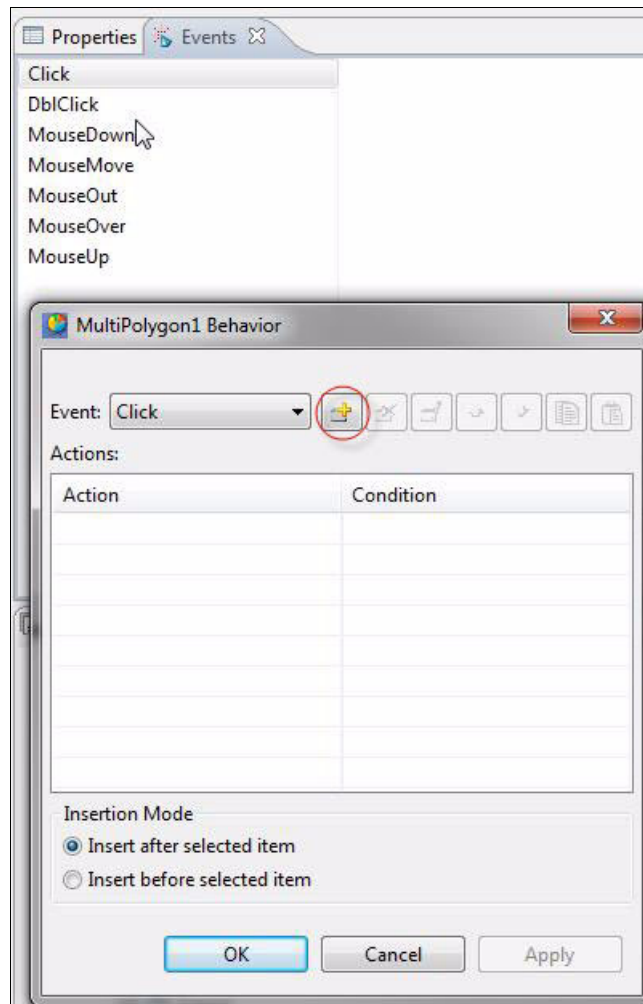


Figure 14-99 Adding a click event for the multipolygon

4. Click the **Add New Action** icon, which is highlighted in Figure 14-99. The Add New Action window is displayed.

5. Select the **Set Values** action, as shown in Figure 14-100, and click **Next**.

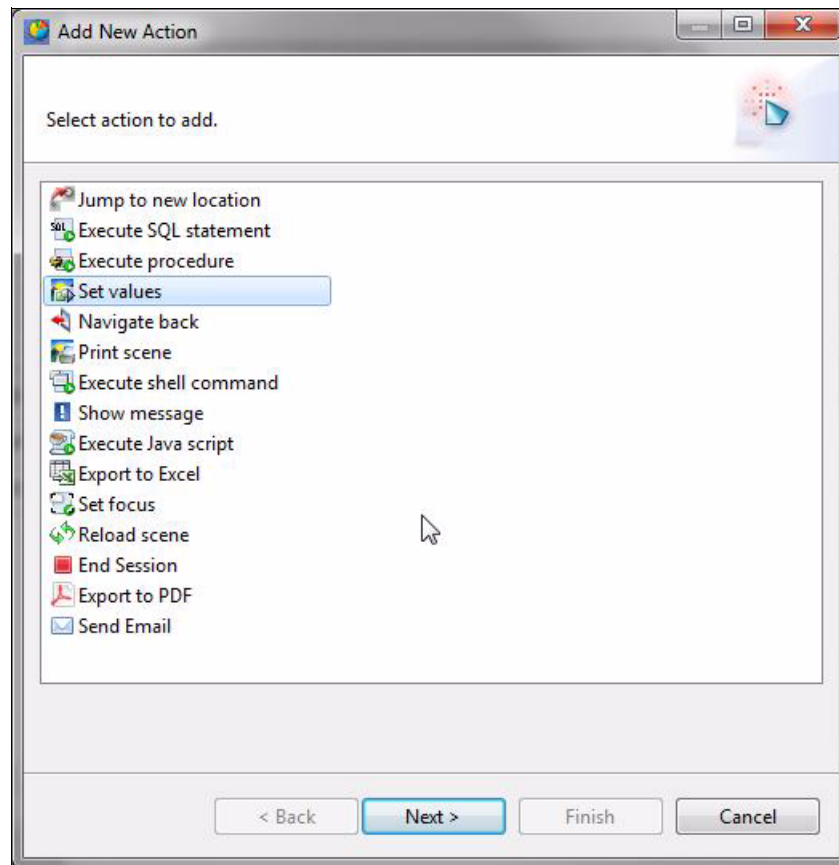


Figure 14-100 Assigning a Set Values action for the click event

6. In the Expression Designer, we now locate Table1 and fully expand it, as shown in Figure 14-101, so that we can see the query parameters for the qryTable query, which is the query that populates the table.

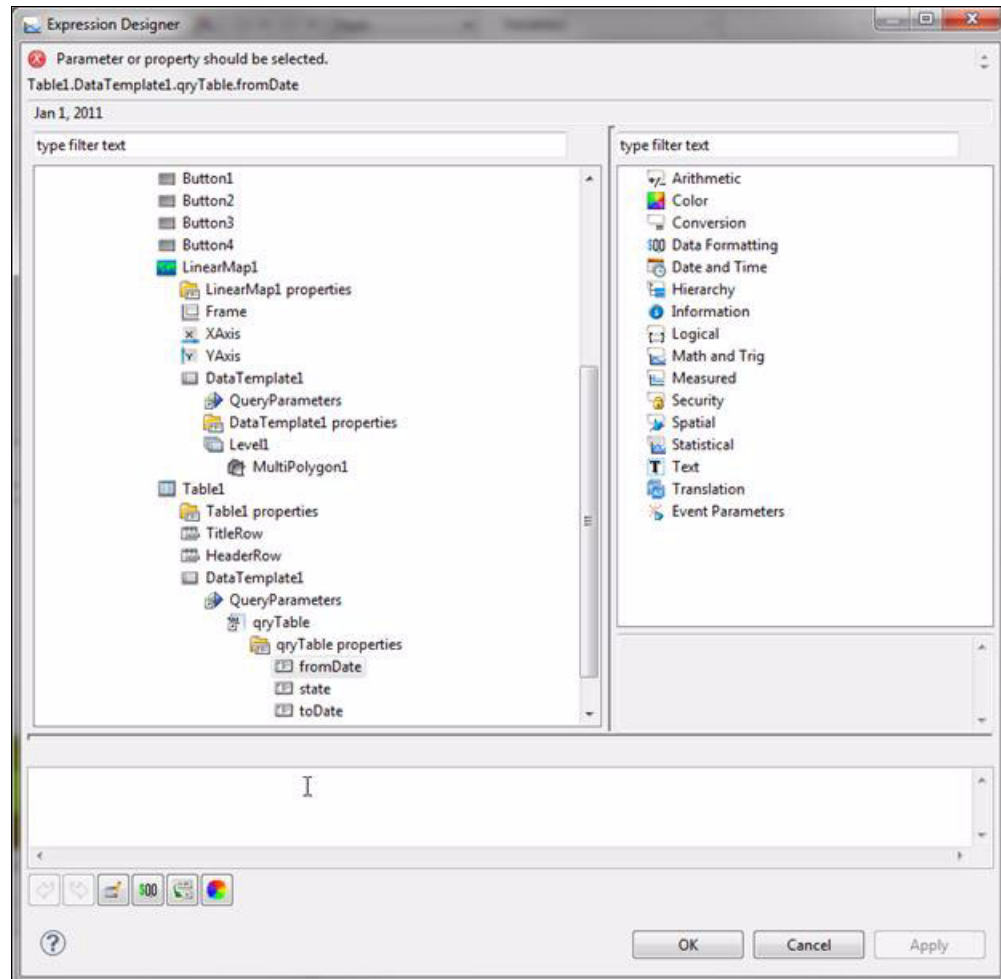


Figure 14-101 Selecting the query parameter values for Table1, driven by the qryTable query

7. “Push” parameter values to the qryTable query by defining **Set values** actions that pass the parameter values. We have four parameter values to set: two for the date range, one for the state, and one that will set the table to be visible only when a state on the map is clicked.

Follow these steps to pass these parameter values:

- a. To pass the fromDate parameter value:

As shown in Figure 14-102, double-click **fromDate** in the Expression Designer, and you will see that the full path of the parameter appears below the pane showing the navigation tree. Double-click the scene parameter named `s_fromDate`, which we created to hold the start of the date range for the time period of the column selected in the OLAP column chart. Then click **OK**.

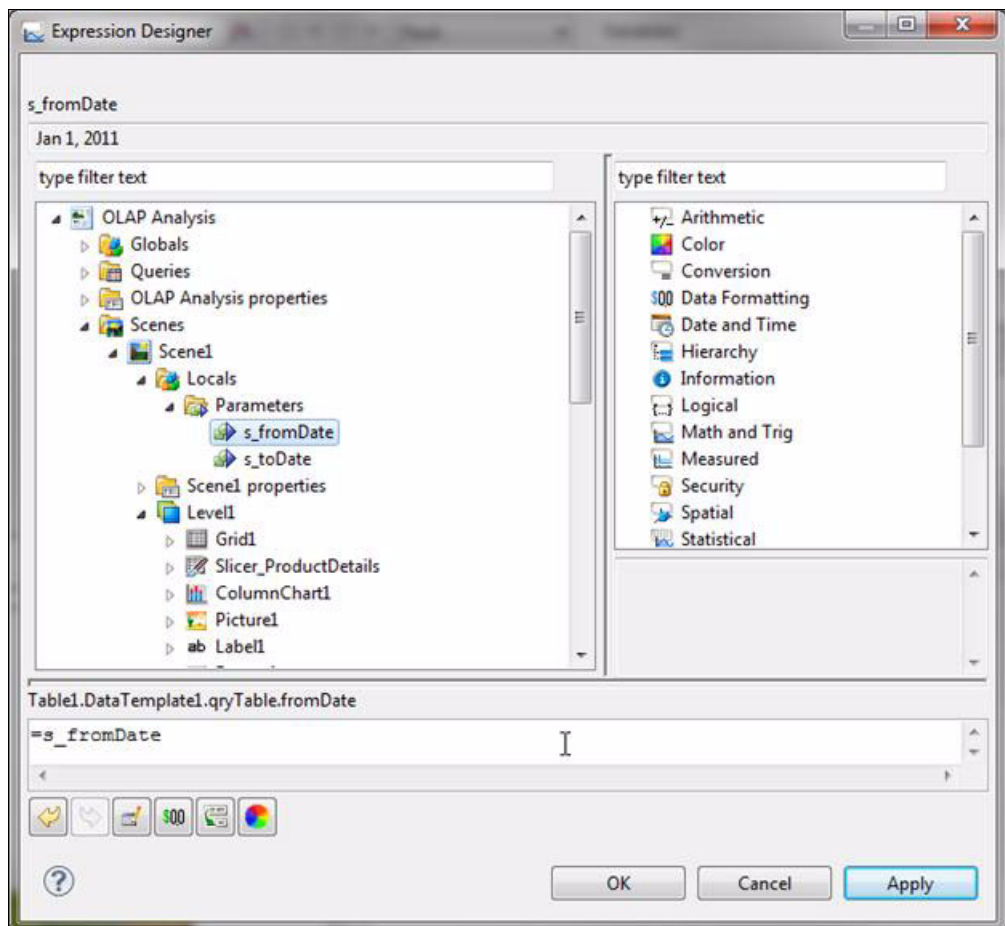


Figure 14-102 Setting the fromDate query parameter

- b. To pass the toDate parameter value:

We repeat the process for toDate that we just completed for fromDate. To indicate that you want to set another parameter value, click the first empty row in the Add New Action window, then click the ... button that appears, as shown in Figure 14-103.

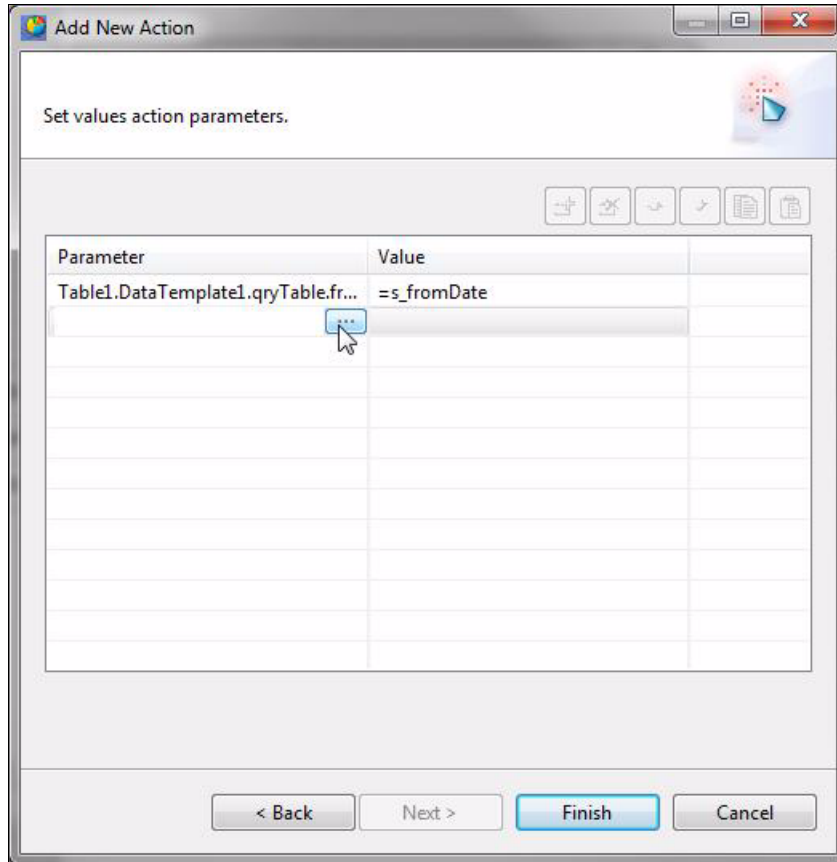


Figure 14-103 Setting the value for another query parameter in the Add New Action window

Using similar steps, double-click the **toDate** property for Table1 in the Expression Designer, then double-click the **s_toDate** scene parameter shown in Figure 14-104, then click **OK**.

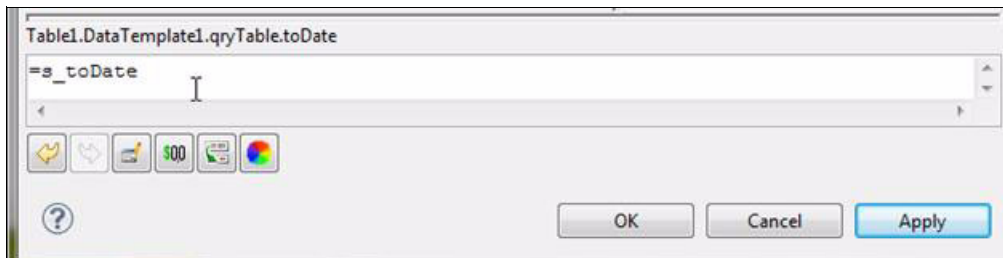


Figure 14-104 Setting the toDate query parameter

The parameter value for toDate is added in the Set values action parameters dialog, as shown in Figure 14-105.

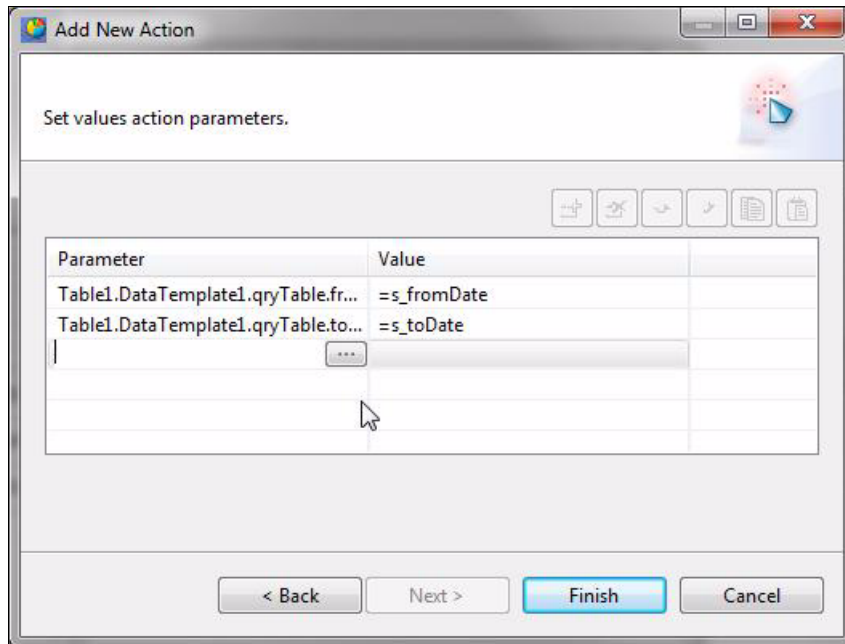


Figure 14-105 The toDate parameter value is now set

- c. To pass the value of the state that is selected in the map:
 - i. Indicate that you want to set another parameter value by clicking the first empty row in the Add New Action window, then click the ... button that appears, as shown in Figure 14-105.
 - ii. Double-click the **state** query parameter in the Expression Designer as was done previously for fromDate and toDate.
 - iii. Set the state parameter to the state for the selected area of the map by expanding the qryMap query, which populates the linear map, and selecting the STATE column, as shown in Figure 14-106. Then click **OK**.

Tip: As mentioned earlier, a multipolygon is drawn for each row in the result set that populates the linear map. Each multipolygon only knows about the data for its own row. Thus, when the user clicks on an area of the map, the data contained in the STATE column will have only one value.

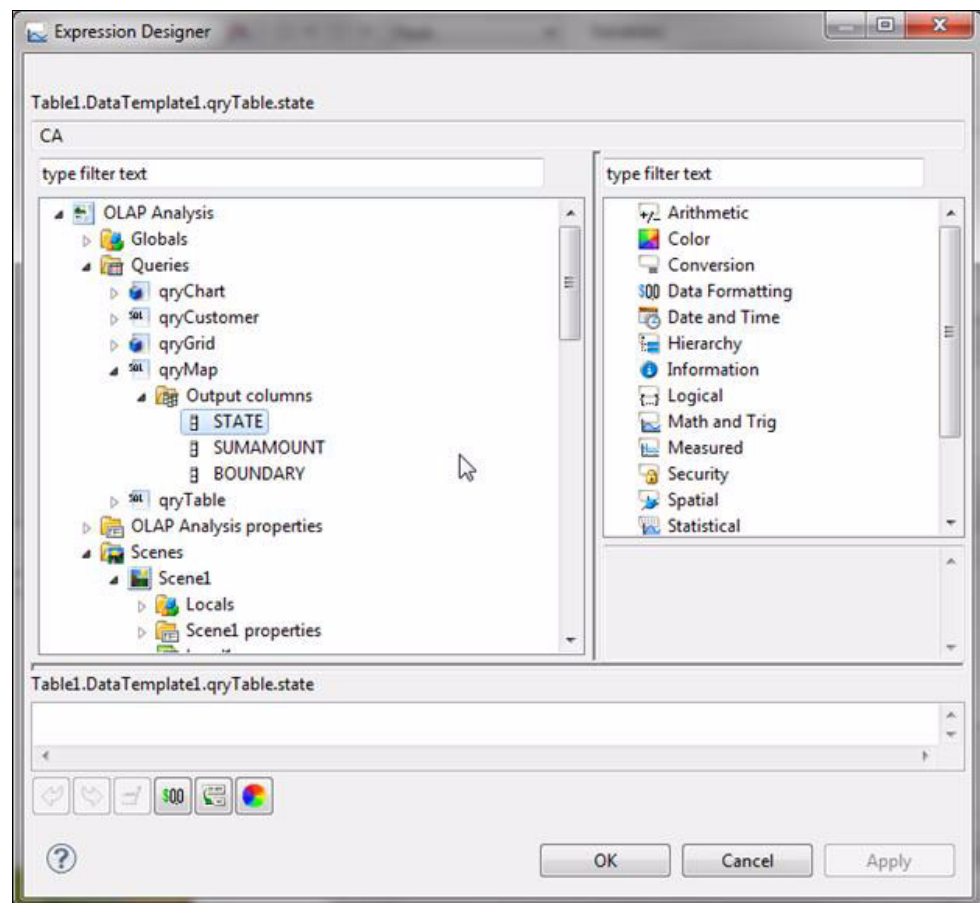


Figure 14-106 Select the STATE column from the qryMap

The parameter value for state now appears in the Set values action parameters dialog, as shown in Figure 14-107.

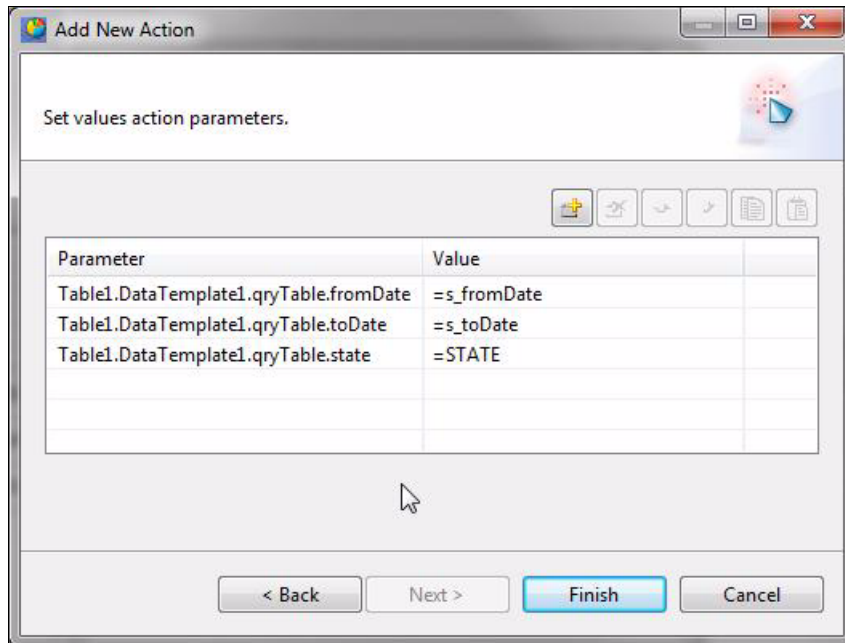


Figure 14-107 The value of the state parameter is now set

- d. To set the table's visibility to "true" only when a state is selected on the map (thus triggering the qryTable query, which populates the table, to run):
 - i. As we did with the fromDate, toDate, and state parameters, to indicate that we want to set another parameter value in the Add New Action window, we click the first empty row in the dialog, then click the ... button that appears.
 - ii. Double-click the **Visible** property for Table1 in the Expression Designer, as shown in Figure 14-108, then click **OK**.

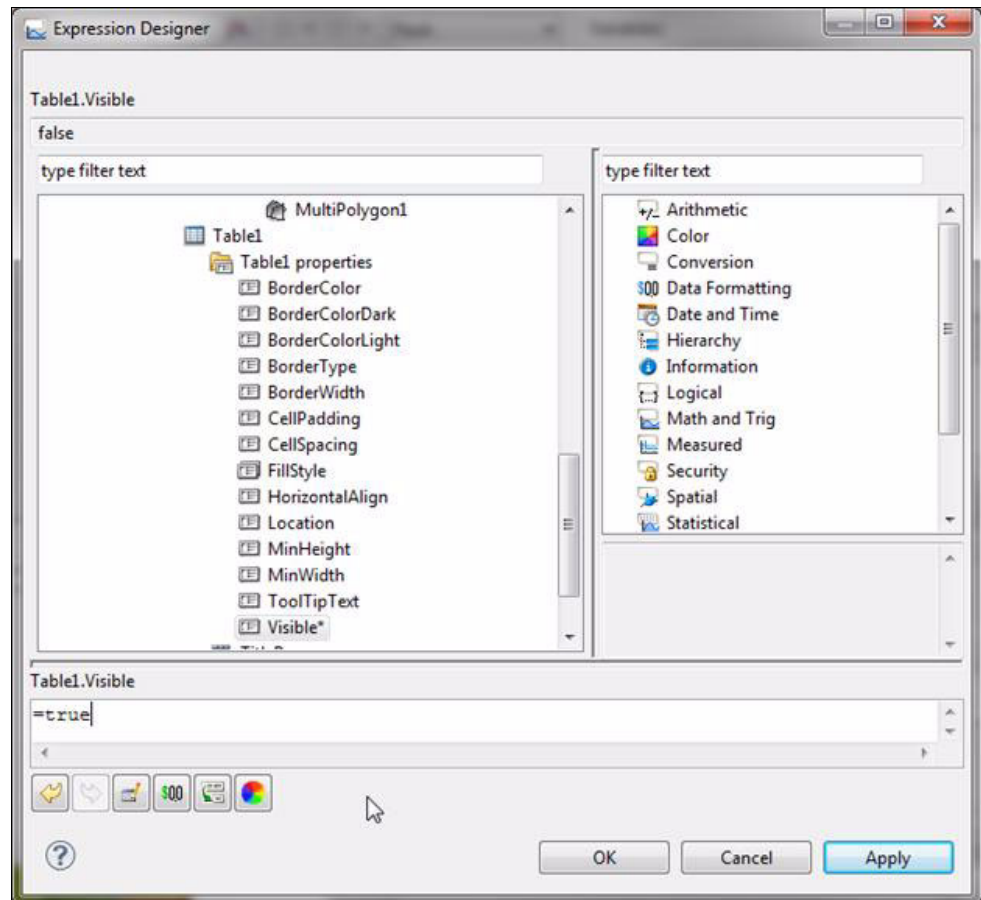


Figure 14-108 The Visible property for Table1

Figure 14-109 shows the four “set values” actions that we have now assigned to the qryTable query parameters. Click **Finish** to close the Add New Action window.

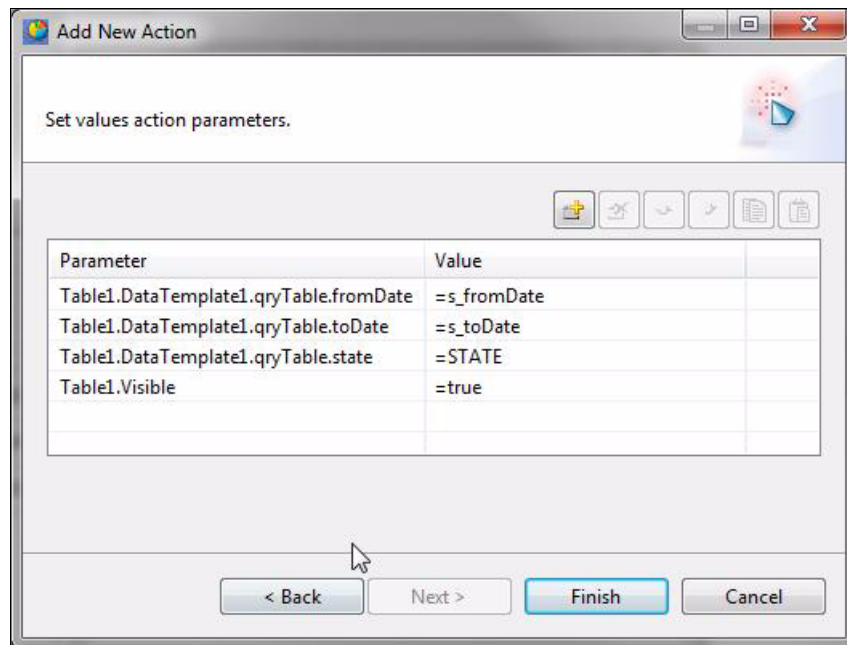


Figure 14-109 Completed click event parameters for the qryTable query

Now that all the click event actions for the multipolygon have been assigned, we are now ready to run the dashboard with all the controls linked to each other. To do so, we click the **Runtime** tab. As seen in Figure 14-110, the table is not visible until a state on the map is clicked:

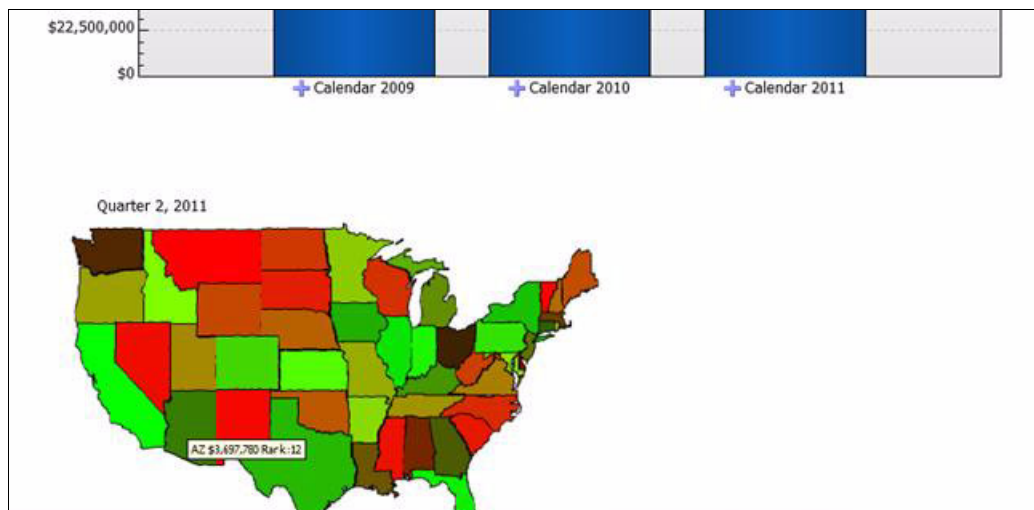


Figure 14-110 Table in the market analysis dashboard is not visible until a state on the map is clicked

When the dashboard user clicks a state on the map, the query that populates the table (qryTable) executes. Before it runs, the query takes as parameters the date range that was selected in the OLAP column chart and the state that was selected on the map. Clicking a state in the map displays the customer data table, populating it with data for the selected state. The example in Figure 14-111 shows the behavior of the finished dashboard when a user selects the state of Texas. The table appears and is populated with Spiffy customer data for the state of Texas.

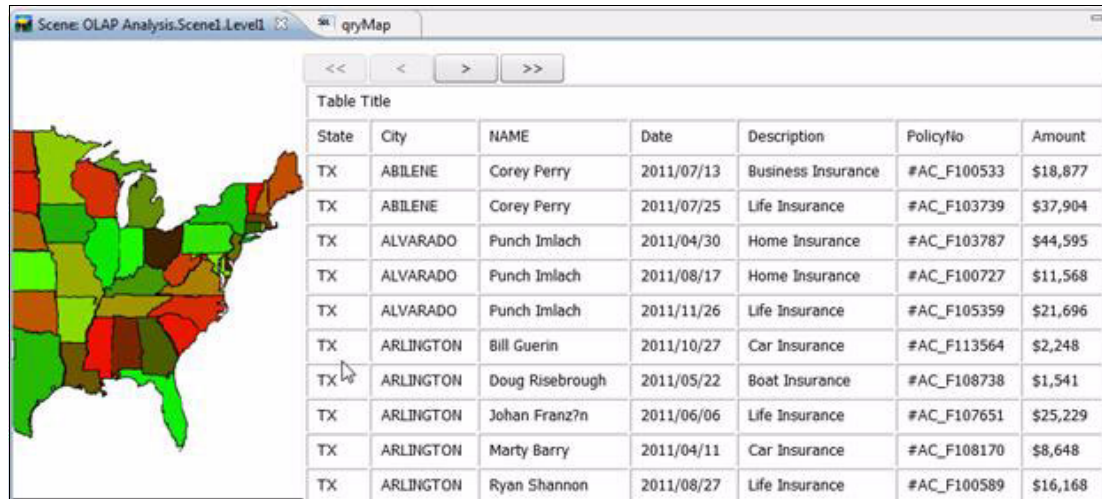


Figure 14-111 Clicking a state in map displays customer data table, populated with data for the state



Deploying created content to QMF users

In the prior chapters, we explained how to create QMF objects such as queries, reports, procedures, and dashboards. After you have created and tested these objects, you can deploy them to QMF users. This chapter walks you through deployment methods, including how to perform these tasks:

- ▶ Include QMF content in enterprise mashups.
- ▶ Deploy content to users by a web link.

15.1 Including QMF content in enterprise mashups

An *enterprise mashup* is a web application that combines information from different sources. These sources can be both internal, such as data from corporate databases, and external, such as data from lookup services available from public websites.

QMF offers support for enterprise mashups by making it possible for you to embed any QMF object in a custom web page or portlet.

To embed QMF objects in a web application, follow these steps:

1. Create a URL address based on the following example (shown next over multiple lines):

```
http://[server]:[port]/QMFWebSphere101/getObject?repository=[repositoryconnection]&key=[objectpath]&run=[yes|no]&user=[userID]&password=[password]&toolbar=[0|1]&[parametername]=[value]
```

In the foregoing syntax:

- *QMFWebSphere101* is the default context root for the QMF for WebSphere application. Here, QMFWebSphere101 is the folder name of the web application under the /webapps directory, as we have deployed QMF under Apache Tomcat.
- *repository* is the name of the repository connection where the object resides.
- *key* is the full name and path of the object in the repository. The object key can be obtained from the Key property value for the object in both the workstation and web user interfaces.

To view the Key property, click the object in the **Repository Explorer** or **Workspace Explorer** and refer to the value of the Key property in the **Properties** view.

- *run* is optional and specifies whether you will run the object (display data) or open the object (display query text). This parameter only applies to query objects. All other objects open in run mode.

Specify **yes** to run a query or **no** to open the query.

- *user* is optional and specifies the user ID that will be used when logging onto the repository. If a repository is secured, and you do not specify a default user ID in the URL address, you will be prompted for a user ID by the Logon Information dialog.
- *password* is optional and specifies the password that will be used for the user ID when logging into the repository.
- *toolbar* is optional and specifies whether or not the object's toolbar and the accompanying user interface data will be displayed around the embedded window.

For example, if the embedded object is a query and you specify 1, the toolbar will be included, along with the query tab at the top of the window and the editor tabs at the bottom. If you specify 0, the toolbar is omitted and only the contents of the query results will be displayed in the embedded window.

- *parametername* is optional and specifies the value of a parameter in a dashboard, report, or query. The parameter name can be from 1 to 17 characters long and the value can be from 1 to 55 characters long. You can specify any number of parameters and values. For example, if you are embedding a QMF query that has global parameters named CustID and ProdID, the URL could contain the following string:

```
...CustID=12&ProdID=15
```

2. Create the iframe HTML element that will be used in the web application or portlet that is accessing the repository object. Use the following syntax:

```
<iframe src="[URLtorepositoryobject]" scrolling="auto" frameborder="no"
align="left" height = "100%" width = "100%"></iframe>
```

In the foregoing syntax:

- *src* is the URL address you created in Step 1.
- *scrolling* specifies whether scroll bars will be defined. (We recommend that you use scroll bars.)
- *frameborder* specifies whether the content of the URL is displayed with or without a frame.
- *align* specifies whether the frame should align to the left or right side of the page and that text should flow around it
- *height* specifies the height of the iframe.
- *width* specifies the width of the iframe.

As an example, suppose that we want to embed the TeamPerformance dashboard shown in Figure 15-1 in a web application.

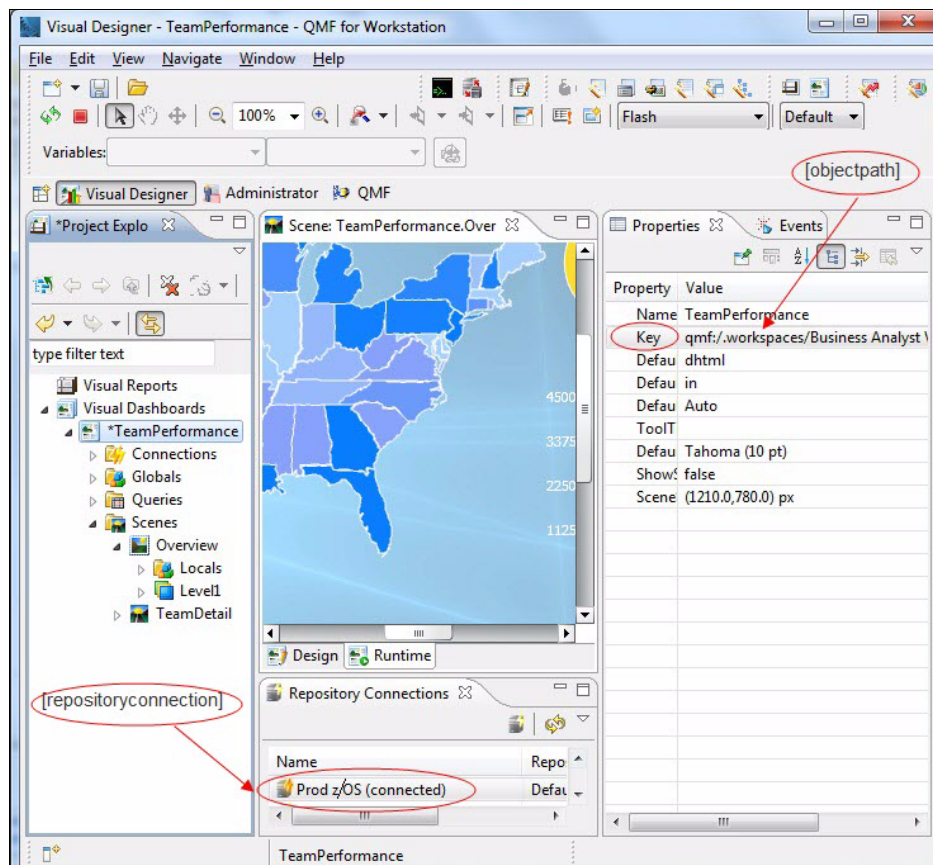


Figure 15-1 Populating the fields of the getObject function for a specific dashboard

To embed this dashboard, we first need to create the URL that points to the location of the object. Figure 15-1 shows that this object resides in the Prod z/OS repository, so we will use that repository name for the *[repositoryconnection]* part of the URL. Figure 15-1 also shows the object's Key property, which will form the *[objectpath]* part of the URL.

The complete HTML for this embedded dashboard would look like that shown in Figure 15-3.

```
<html>
<head>
<title>Team Performance</title>
</head>
<body>

<iframe src="http://[server]:[port]/QMFWebSphere101/getObject?repository=Prod
z/OS&key=qmf:/.workspaces/Business Analyst
View/Dashboards/TeamPerformance&toolbar=1" name="frame1" scrolling="auto"
frameborder="no" align="left" height="100%" width="100%">
</iframe>

</body>
</html>
```

Figure 15-3 Full example of HTML page containing an iFrame with a getObject function as its source

15.2 Deploying content to users by a web link

You can use the Web Link wizard to create web links to QMF objects. You can then use these links to open the objects directly in web browsers.

To create a web link to a QMF object, follow these steps:

1. Select **File** → **New** → **Other Web Link**, as shown in Figure 15-4, to open the Web Link wizard.

Tip: You can also access this wizard from the **Create Web Link** context menu option in the Repository tree. When you use this method, the **Repository Object** field and user-defined parameters are prepopulated.

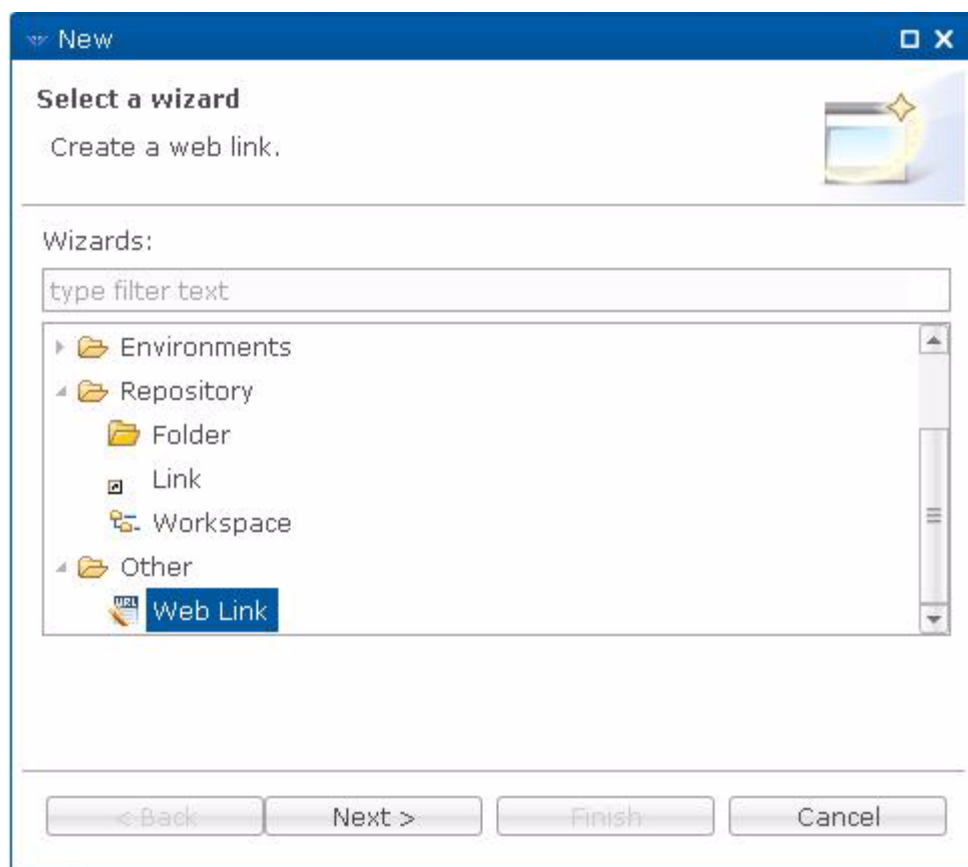


Figure 15-4 Starting the Web Link wizard

2. In the **Repository Object** field, click the **Open from Repository** button (...) to select the object to which you want to create a web link. In this example, we select the TeamPerformance dashboard, as shown in Figure 15-5.

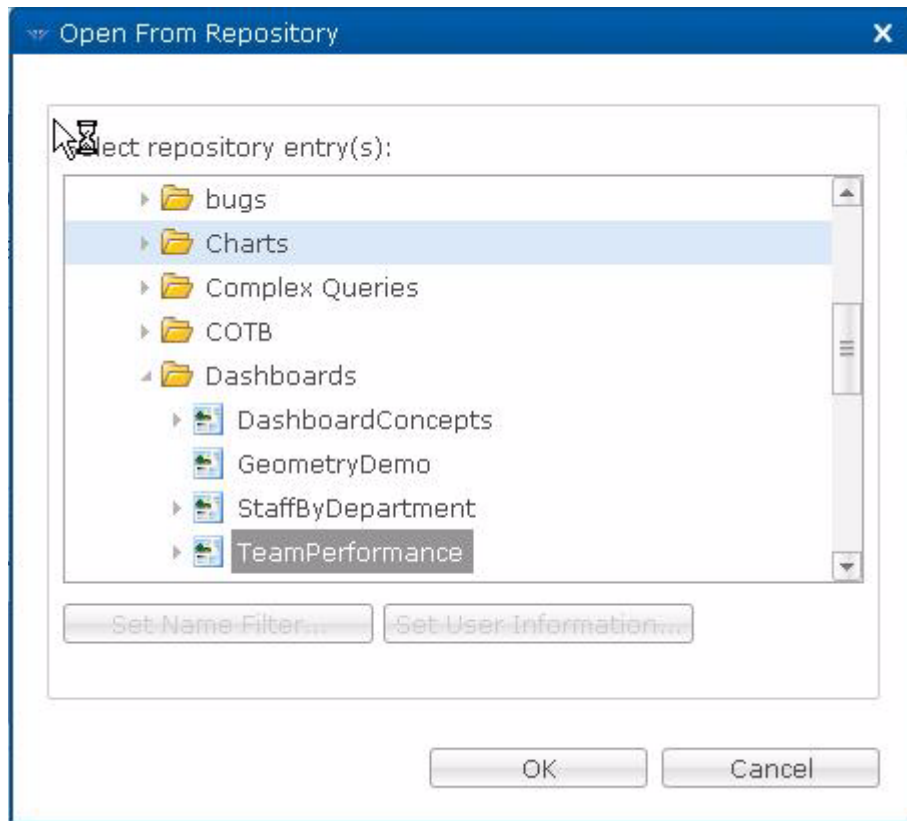


Figure 15-5 Selecting the object to which you want to create a web link

3. Select settings for the following check boxes in the **Predefined Parameters** section of the wizard:
 - a. **Toolbar**: This check box indicates whether to show or hide the toolbar. By default, the check box is selected, which means the toolbar is shown.
 - b. **Run**: This check box indicates whether to run the object when it is opened. By default, the check box is selected, which means the object is run.
 - c. **Environment**: This check box indicates whether to use the default environment to run the object. By default, the check box is not selected. It means that the default environment will be used. If you want to use an environment other than the default environment, select the check box and choose from the list of environments. This check box is disabled if there are no environments to choose from.
 - d. **Renderer mode**: This check box is available for visual reports or dashboards. It determines the mode in which the visual object is rendered. The default render mode is Flash.

Figure 15-6 shows the settings for our sample dashboard, Team Performance.

The screenshot shows a 'Web Link' dialog box with the following sections:

- Repository object:** qmf:/workspaces/Business Analyst View/Dashl
- Predefined Parameters:**
 - ☒ Toolbar
 - ☒ Run
 - ☐ Environment:
 - ☐ Renderer mode: Flz
- User Defined Parameters:**
 - Variables:**

Include in URL	Name	Value
<input type="checkbox"/>	g_year	
<input type="checkbox"/>	g_startMonth	7
<input type="checkbox"/>	g_endMonth	9
<input checked="" type="checkbox"/>	g_team	0
<input type="checkbox"/>	g_month	0
<input type="checkbox"/>	g_timeId	15

At the bottom are buttons: < Back, Next >, Finish, and Cancel.

Figure 15-6 Setting parameters for the object to be deployed

- Also on this page (Figure 15-6), you can specify parameters for the selected repository object in the **User Defined Parameters** section. For example, you can define variables to be passed to the object at run time. You can add parameters if they are needed. All parameters can be sorted by name.

Parameters that you add can be included in the link by selecting the **Include in URL** check box. Because the `g_team` parameter is selected to be included in the URL, as shown in Figure 15-6, the finished URL will have the following reference to `g_team` in it:

`&g_team=0`

5. Click **Next** to display the next page of the Web Link wizard, shown in Figure 15-7.

Save	Data source	Login
<input checked="" type="checkbox"/>	Prod z/OS	

Web link name: TeamPerformance

Create

Web link: http://localhost:8080/QMFWebSphere101/g?TeamP

< Back Next > Finish Cancel

Figure 15-7 Setting login and password information for the data sources

On this page, you can enter authentication credentials to free users from having to log on when they use the object. (These credentials must be predefined to the data source before they are entered here.)

You can do the following actions:

- Edit existing logins (user IDs) and passwords for data sources.
- Click the **Add** icon to add logins and passwords for data sources.
- Click the **Delete** icon to remove logins and passwords.

You can optionally have the user provide data source credentials at run time by leaving the login and password information blank.

In the **Web link name** field, enter the name of the object as you want it to appear in the URL. The name that you enter here does not have to be the same as how the object is named in the repository or QMF catalog. For example, if you wanted to name the Team Performance dashboard STAFF in the web link, you would enter STAFF in the Web link name field and that name would appear in the URL:

`http://{host}:{port}/{app_context}/g?STAFF`

If your chosen name is already taken, QMF prompts you to change the name you entered. If you leave this field empty, QMF automatically assigns a name.

6. Click **Create** to generate the web link, which appears beneath the **Create** button.

The full web link generated when **Create** is clicked in Figure 15-7 is as follows:

`http://localhost:8080/QMFWebSphere101/g?TeamPerformance&g_team=0`

The general format of the link is as follows:

`http://[host]:[port]/QMFWebSphere101/g?[weblink_name]&variable1=value ...`

7. Click the **Run** icon to open the created web link in a browser.
8. Click **Finish** to close the Web Link wizard.

At this point, you can copy and paste the generated web link in the URL field of a browser to display the object.



Analytical functions available in the Expression Designer

This appendix lists the broad range of functions available in the QMF for Workstation Expression Designer and provides a brief description of each. For more information about how to use each of these functions, see the QMF for Workstation help.

The following categories of functions are available:

- ▶ Arithmetic functions
- ▶ Color functions
- ▶ Conversion functions
- ▶ Data formatting functions
- ▶ Date and time functions
- ▶ Hierarchy functions
- ▶ Information functions
- ▶ Logical functions
- ▶ Math and trigonometry functions
- ▶ Measured functions
- ▶ Security function
- ▶ Spatial functions
- ▶ Statistical functions
- ▶ Text functions
- ▶ Visual report functions

A.1 Arithmetic functions

Arithmetic functions allow you to calculate values based on the specified function. Supported functions include:

- ▶ `add()` — Adds two numeric arguments.
- ▶ `divide()` — Divides two numeric arguments.
- ▶ `mod()` — Returns the modulus, or integer residue of a number following an integral division by a second number.
- ▶ `multiply()` — Multiplies two numeric arguments.
- ▶ `percent()` — Divides number by one hundred.
- ▶ `subtract()` — Subtracts two numeric arguments.
- ▶ `sum()` — Returns the sum of an expression, calculated based on the contents of the query result set.
- ▶ `unaryMinus()` — Negates the numeric argument.

A.2 Color functions

Color functions help you customize visual presentation of reports and dashboard objects. These functions include:

- ▶ `blend()` — Blends two colors using a specified alpha ratio.
- ▶ `blueVal()` — Returns the blue component of the supplied RGB color value.
- ▶ `brighten()` — Brightens a color by blending it with pure white, according to a specified alpha ratio.
- ▶ `colorMap()` — Returns a color based on a numeric value that has been defined in a given color map.
- ▶ `colorSeq()` — Returns a color based on an index value that specifies a wedge in a color sequence wheel.
- ▶ `darken()` — Darkens a color by blending it with pure black, according to a specified alpha ratio.
- ▶ `Granville()` — Returns the green component of the supplied RGB color value.
- ▶ `hsl()` — Returns a color with the specified hue, saturation, and intensity.
- ▶ `hue()` — Returns the hue of a color.
- ▶ `luminance()` — Returns the luminance (intensity) component of a supplied color.
- ▶ `redVal()` — Returns the red component of the supplied RGB color value.
- ▶ `rgb()` — Returns a color with the specified amounts of red, green, and blue.
- ▶ `saturation()` — Returns the saturation component of a supplied color.

A.3 Conversion functions

Conversion functions are used to convert the parameter value from its current data type to another data type. These functions include:

- ▶ `asDate()` — Converts the specified number to a date.
- ▶ `asDateTime()` — Converts the specified number to a date and time.
- ▶ `asTime()` — Converts the specified number to a time.
- ▶ `boolToNum()` — Converts Boolean true and false values to 1 and 0, respectively.
- ▶ `boolToStr()` — Converts the Boolean true and false values to the text string True or False, respectively.
- ▶ `formatNum()` — Formats the supplied numeric value according to the supplied format template.
- ▶ `getX()` — Returns the X coordinate of a given point.
- ▶ `getY()` — Returns the Y coordinate of a given point.
- ▶ `hex()` — Returns the hexadecimal value of a given number.
- ▶ `indexOf()` — Returns a list of indices that represent elements from the second list that are contained in the first list.
- ▶ `numToBool()` — Converts a number to a Boolean (true or false) value.
- ▶ `numToColor()` — Converts a number to a color.
- ▶ `numToStr()` — Converts a number to a string with default number formatting.
- ▶ `poly()` — Constructs a pointset from a series of point values.
- ▶ `polySet()` — Constructs a polygon set from a series of polygon values.
- ▶ `polySetToStr()` — Converts a polyset into a text string.
- ▶ `polyToStr()` — Converts a pointset into a text string.
- ▶ `pt()` — Constructs a point from individual x-y values.
- ▶ `ptToStr()` — Converts a point into a text string.
- ▶ `rwinding()` — Reverses the order of the supplied points.
- ▶ `spoly()` — Converts a text value into a pointset.
- ▶ `spolySet()` — Converts a text value into a polygon set.
- ▶ `spt()` — Converts a text value into a point.
- ▶ `str()` — Converts an object to a text string.
- ▶ `strToBin()` — Converts a text string to binary notation.
- ▶ `strToHex()` — Converts a text string to hexadecimal notation.
- ▶ `strToNum()` — Converts a string value into a numeric value.
- ▶ `strToTextSet()` — Converts an arbitrary number of explicit strings into a collection of strings.

A.4 Data formatting functions

Data formatting functions reformat data values to and from database and operating system formats. These functions include:

- ▶ `dbDateToUSLocale()` — Converts a date from database format to USA format.
- ▶ `dbDateToWin()` — Converts date values from a database format to a text format compatible with the current user locale.
- ▶ `dbMoneyToUSLocale()` — Converts a money amount from database format to USA format.
- ▶ `dbMoneyToWin()` — Converts money values from database format to a text format compatible with the current user locale.
- ▶ `dbNumToUSLocale()` — Converts a number from database format to USA format.
- ▶ `dbNumToWin()` — Converts numeric values from database format to a text format compatible with the current user locale.
- ▶ `dateToDB()` — Converts a date and time that is specified as a text string in USA format to database format that is displayed as YYYY-MM-DD HH:MM:SS.
- ▶ `dateToDBInformix()` — Converts a date from USA format to Informix database format.
- ▶ `dateToDBInformixInt()` — Converts a date from USA format to Informix database format using specified interval settings.
- ▶ `dateToDBOracle()` — Converts the date from the operating system format to an Oracle database format.
- ▶ `moneyToDB()` — Converts a money amount from USA format to JDBC style.
- ▶ `winDateToDB()` — Converts the date from the operating system format to a database format.
- ▶ `winDateToDBInformix()` — Converts the date from the operating system format to an Informix database format.
- ▶ `winDateToDBInformixInt()` — Converts the Informix date from Windows format to database format based on specified interval settings.
- ▶ `winDateToDBOracle()` — Converts the date from the operating system format to an Oracle database format.
- ▶ `winMoneyToDB()` — Converts the money from operating system format to the database format.
- ▶ `winNumToDB()` — Converts the number from operating system format into database format.

A.5 Date and time functions

Date and time functions return specific elements of date and time values. These functions include:

- ▶ `date()` — Returns a number that represents the specified date.
- ▶ `dateTime()` — Returns a number that represents the specified date and time.
- ▶ `day()` — Returns the day of the month corresponding to the supplied date number.
- ▶ `dayName()` — Returns the name of the day corresponding to the supplied date number.
- ▶ `dayOfWeek()` — Returns a number that represents the day of the week corresponding to the supplied date number.
- ▶ `dayOfYear()` — Returns the number of days since the start of the year corresponding to the supplied date value.
- ▶ `hour()` — Returns the hour component corresponding to the supplied date number.
- ▶ `jday()` — Returns the Julian day for the given date.
- ▶ `microsecond()` — Returns the fractional seconds for the given date and time number.
- ▶ `millisToDate()` — Returns a date and time for a numeric value specified in milliseconds.
- ▶ `minute()` — Returns the minute component corresponding to the supplied date and time number.
- ▶ `month()` — Returns the month component corresponding to the supplied date number.
- ▶ `monthName()` — Returns the name of the month corresponding to the supplied date number.
- ▶ `now()` — Returns the current date and time as a floating point number.
- ▶ `quarter()` — Returns the quarter corresponding to the supplied date number.
- ▶ `second()` — Returns the second component of the supplied date and time number.
- ▶ `time()` — Returns the number that represents the specified time.
- ▶ `today()` — Returns the current date as a floating-point number.
- ▶ `year()` — Returns the year component of the supplied date number.

A.6 Hierarchy functions

Hierarchy functions identify a parameter value's position in hierarchical representations, such as organization charts. These functions include:

- ▶ `childCount()` — This function computes the number of child data points for each parent data point in the hierarchical layout.
- ▶ `childDepth()` — Returns the depth of the current data point relative to the top-most data point in a hierarchical layout object.
- ▶ `hasChildren()` — Returns whether the current item in the data template has child items in the hierarchical layout.
- ▶ `hasParent()` — Returns whether or not the current item in the data template has a parent item.
- ▶ `hasSiblings()` — Returns whether or not the current item in the data template has a sibling item.
- ▶ `siblingCount()` — Returns the number of siblings of the current item in a hierarchical layout such as a cluster graph, organization chart, or a tree chart.
- ▶ `siblingNumber()` — Returns the 0-based index of the current item with respect to other siblings at the same level.

A.7 Information functions

Information functions return information about specified data objects. These functions include:

- ▶ `accum()` — This function computes the sum of values for a given numeric value, across a specific, contiguous group of rows in a query result set. The function can only be called within a data template.
- ▶ `coalesce()` — Evaluates the specified arguments in order and returns the first one that is not null.
- ▶ `field()` — Executes the specified query and returns from the specified query result set a textset containing all values for the specified column.
- ▶ `fieldValue()` — Executes the specified query and returns the value for the specified field from the specified row.
- ▶ `groupRowNumber()` — Returns the number of the row that contains the specified group.
- ▶ `isNull()` — Tests whether or not the supplied parameter is null.
- ▶ `isSummaryRow()` — Determines whether or not the given row is the table's summary row.
- ▶ `pointCount()` — Returns the number of data points contained in the current set of query results.
- ▶ `pointNumber()` — Returns the zero-based index of the current row in the current set of query results.
- ▶ `stockImage()` — Returns the image data from the stock images collection corresponding to the supplied name.

A.8 Logical functions

Logical functions return values based on logical operations performed on parameter values. These functions include:

- ▶ `and()` — Returns true if both numeric arguments are true.
- ▶ `equal()` — Returns true if two numeric arguments are equal.
- ▶ `greaterOrEqual()` — Returns true if the first numeric argument is greater than or equal to the second numeric argument.
- ▶ `greaterThan()` — Returns true if the first numeric argument is greater than the second numeric argument.
- ▶ `if()` — Evaluates a condition and returns the corresponding true or false expression.
- ▶ `@if()` — Tests for specific values within an expression and returns the results as either values or additional tests.
- ▶ `lessOrEqual()` — Returns true if the first numeric argument is less than or equal to the second numeric argument.
- ▶ `lessThan()` — Returns true if the first numeric argument is less than the second numeric argument.
- ▶ `not()` — Returns true if the numeric argument is not true.
- ▶ `notEqual()` — Returns true if two numeric arguments are not equal.
- ▶ `or()` — Returns true if either of the numeric arguments is true.

A.9 Math and trigonometry functions

Math and trigonometric functions calculate values based on the specified mathematical function. These functions include:

- ▶ `abs()` — Returns the absolute value of a given number.
- ▶ `acos()` — Returns the inverse cosine in degrees.
- ▶ `acosh()` — Returns the inverse hyperbolic cosine in degrees.
- ▶ `actn()` — Returns the inverse cotangent in degrees.
- ▶ `asin()` — Returns the inverse sine in degrees.
- ▶ `asinh()` — Returns the inverse hyperbolic sine in degrees.
- ▶ `atan()` — Returns the inverse tangent in degrees.
- ▶ `atan2()` — Returns the inverse tangent, derived from the supplied opposite (y) and adjacent (x) values.
- ▶ `atanh()` — Returns the inverse hyperbolic tangent in degrees.
- ▶ `ceiling()` — Returns the closest integer greater than or equal to the supplied number.
- ▶ `cos()` — Returns the cosine of the supplied angle.
- ▶ `cosh()` — Returns the hyperbolic cosine in degrees.
- ▶ `ctn()` — Returns the cotangent of the supplied angle.
- ▶ `degrees()` — Converts radians to degrees.
- ▶ `exp()` — Returns e, raised to the power of a given number.

- ▶ `fact()` — Returns the factorial of a supplied number.
- ▶ `floor()` — Returns the closest integer less than or equal to the supplied number.
- ▶ `int()` — Rounds the supplied number to the nearest integer.
- ▶ `ln()` — Returns the natural logarithm of a given number.
- ▶ `log()` — Returns the logarithm of a number with a specified base.
- ▶ `log10()` — Returns the base-10 logarithm of a given number.
- ▶ `max()` — Returns the largest number in an explicit series.
- ▶ `min()` — Returns the smallest number in an explicit series.
- ▶ `pi()` — Returns the value of pi.
- ▶ `power()` — Raises a given number to a supplied power.
- ▶ `radians()` — Converts degrees to radians.
- ▶ `rand()` — Returns a random value, uniformly distributed between lower and upper limits.
- ▶ `randInt()` — Returns a random value between 0 and 32,767.
- ▶ `randn()` — Returns a random value, with a Gaussian Normal distribution.
- ▶ `round()` — Returns the closest integer.
- ▶ `sign()` — Returns the sign of the value, reflecting whether it is negative, zero, or positive.
- ▶ `sin()` — Returns the sine of the supplied angle.
- ▶ `sinc()` — Computes the SINC value, normalized to 1.
- ▶ `sinh()` — Returns the hyperbolic sine in degrees.
- ▶ `sqrt()` — Returns the square root of the supplied number.
- ▶ `tan()` — Returns the tangent of the supplied angle.
- ▶ `tanh()` — Returns the hyperbolic tangent in degrees.

A.10 Measured functions

Measured functions express a given value as a specified unit of measure. These functions include:

- ▶ `cm()` — Specifies a number in centimeters.
- ▶ `in()` — Specifies a number in inches.
- ▶ `mm()` — Specifies a number in millimeters.
- ▶ `pixels()` — Specifies a number with a description of pixels.
- ▶ `points()` — Specifies a number with a description of points.
- ▶ `removeUnits()` — Removes units from the specified measured value.

A.11 Security function

There is one security function, as follows:

- ▶ `isSecurityLevel()` — Specifies the name of the security list that will be used to tailor visual report or visual dashboard content based on the security level of the user.

A.12 Spatial functions

Spatial functions are used when mapping spatial data. These functions include:

- ▶ `centerPoint()` — Determines the center point of a polygon described by a pointset (a series of x-y vertices).
- ▶ `moveCenter()` — Centers the pointset at the specified location.
- ▶ `rotate()` — Rotates a pointset clockwise by the specified angle about a reference point.
- ▶ `rotateInPlace()` — Rotates a pointset by the specified angle about its center point.
- ▶ `scale()` — Scales the pointset by the specified scale factors relative to a reference location.
- ▶ `scaleInPlace()` — Scales the pointset by the specified scale factors relative to its center point.
- ▶ `translate()` — Translates the pointset by the specified offset.
- ▶ `xspan()` — Computes the width of a bounding rectangle encompassing the supplied pointset.
- ▶ `yspan()` — Computes the height of a bounding rectangle encompassing the supplied pointset.

A.13 Statistical functions

Statistical functions are used to perform standard statistical analysis on supplied parameter values. These functions include:

- ▶ `avedev()` — Returns the average deviation of a given expression, computed across all rows in the query result set, regardless of the index of the current row in the result set.
- ▶ `average()` — Returns the average of a given numeric value, computed across all rows in the query result set, regardless of the index of the current row in the result set.
- ▶ `chi2()` — Computes the chi-square probability function with specified degrees of freedom.
- ▶ `comb()` — Returns the number of distinct combinations that can be drawn from a collection of items.
- ▶ `kurt()` — Returns the kurtosis (4th moment) of a given numeric value within the scope of a data template.
- ▶ `largest()` — Returns the largest value in a given expression, within the scope of a data template.
- ▶ `linreg()` — Performs a linear regression on a given data series, across a specific, contiguous group of rows in a query result set.
- ▶ `perm()` — Returns the number of permutations of a given size that can be drawn from a collection of a larger size.
- ▶ `skew()` — Returns the skew of a series about the mean (3rd moment).
- ▶ `smallest()` — Returns the smallest value in a given numeric query column within the query result set.
- ▶ `stddev()` — Returns the standard deviation of an expression, calculated within the scope of the query result set.
- ▶ `variance()` — Returns the variance of the supplied numeric value, taken across the entire contents of the query result set.

A.14 Text functions

Text functions allow you to format and manipulate textual data or return specific information about supplied data sources. These functions include:

- ▶ `concat()` — Concatenates two text strings into a single string.
- ▶ `decrypt()` — Decrypts the supplied encrypted message.
- ▶ `encrypt()` — Encrypts the supplied plaintext message.
- ▶ `filterTextSet()` — Returns a filtered text set according to a filter pattern. The pattern can contain SQL LIKE special characters, such as '%' and '_'.
- ▶ `filterTextSetRegexp()` — Returns a filtered textSet according to a filter pattern. The pattern should be in JAVA REGEX syntax.
- ▶ `find()` — Returns the index of the first occurrence of the supplied substring in a supplied string, starting the search from a given location in the string.
- ▶ `findNoCase()` — Returns the index of the first occurrence of the supplied substring in a supplied string, starting the search from a given location in the string.
- ▶ `format()` — Returns a string whose content is formatted based on a supplied expression.
- ▶ `insert()` — Inserts a substring into the string at the indicated offset.
- ▶ `left()` — Extracts the leftmost n characters from a larger text string.
- ▶ `length()` — Calculates the number of characters in the supplied string.
- ▶ `lineCount()` — Returns the number of lines of text that are separated by carriage returns and/or line feeds.
- ▶ `lineDelete()` — Creates and returns a new textset with a deleted line.
- ▶ `lineInsert()` — Creates and returns a new textset with an inserted line.
- ▶ `lineReplace()` — Creates and returns a new textset with a replaced line.
- ▶ `lineText()` — Returns a specific line of text from the supplied textset.
- ▶ `lower()` — Converts the supplied text to lower case.
- ▶ `ltrim()` — Returns the string with all leading blanks removed.
- ▶ `mid()` — Extracts a portion of text from a specified position for a specified number of characters from a larger text string.
- ▶ `repeat()` — Returns the given string repeated a given number of times based on the value of the second parameter.
- ▶ `reverse()` — Reverses the order of the characters in the supplied text.
- ▶ `right()` — Extracts a portion of text from the right.
- ▶ `rtrim()` — Returns the string with all trailing blanks removed.
- ▶ `space()` — Creates a string of blanks with the number of blanks in the string based on the value of the parameter.
- ▶ `substitute()` — Finds and replaces a specified occurrence or all occurrences of a string of text.
- ▶ `trim()` — Removes the white space from both the left and right of the supplied text.
- ▶ `upper()` — Converts the supplied text to upper case.

A.15 Visual report functions

Visual report functions are used in visual reports. These functions include:

- ▶ `count()` — Returns the count of rows in the group. This function is only used in the summary row of the given group.
- ▶ `first()` — Returns the first column value in the group. This function is only used in the summary row of the given group.
- ▶ `last()` — Returns the last column value in the group. This function is only used in the summary row of the given group.
- ▶ `max()` — Returns the maximum column value in the group. This function is only used in the summary row of the given group.
- ▶ `min()` — Returns the minimum column value in the group. This function is only used in the summary row of the given group.
- ▶ `sum()` — Returns the sum of the column values in the group. This function is only used in the summary row of the given group.
- ▶ `csum()` — Returns the cumulative sum of the column across the group and all prior groups. This function is only used in the summary row of the given group.
- ▶ `average()` — Returns the average of the column values in the group. This function is only used in the summary row of the given group.
- ▶ `stddev()` — Returns the standard deviation of the column values across the group. This function is only used in the summary row of the given group.
- ▶ `pct()` — Returns the percentage value that the given column represents with respect to the total group. This value also overrides the corresponding detail value with the line item percentage figure. This function is only used in the summary row of the given group.
- ▶ `tpct()` — Returns the percentage value that the group represents with respect to the values across all groups. This value also overrides the corresponding detail value with the line item percentage figure. This function is only used in the summary row of the given group.
- ▶ `cpct()` — Returns the cumulative percentage value that the given column represents with respect to the total group. The corresponding detail value is replaced with a percentage value of the group that is expressed as a cumulative total on a per-row basis (for example, 10%, 30%, and 60% for lines contributing 10%, 20%, and 30%). This function is only used in the summary row of the given group.
- ▶ `tcpct()` — Returns the cumulative percentage value that the group represents with respect to the values across all groups. The corresponding group value is replaced with a percentage value of all groups that is expressed as a cumulative total on a per-row basis. For example, groups contributing 10%, 20%, and 30% of the total would be listed as 10%, 30%, and 60% (if listed smallest to largest). This function is only used in the summary row of the given group.
- ▶ `Date` — Returns the current date.
- ▶ `Time` — Returns the current time.
- ▶ `Row` — Returns the row number in the query.
- ▶ `Page` — Returns the current page number.
- ▶ `RowPage` — Turns the current row number in the query relative to the current page. For example, if 60 rows can be accommodated per page, the first row on page 2 will have a Row value of 61 and a RowPage value of 1.

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 publications

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- ▶ *DB2 10 for z/OS Technical Overview*, SG24-7892
- ▶ *Optimizing DB2 Queries with IBM DB2 Analytics Accelerator for z/OS*, SG24-8005
- ▶ *Enterprise Data Warehousing with DB2 9 for z/OS*, SG24-7637

You can search for, view, download, or order these documents and other Redbooks publications, Redpaper publications, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Other publications

These publications are also relevant as further information sources:

- ▶ *Introducing QMF*, GC19-2885
- ▶ *Getting Started with DB2 QMF for Workstation and DB2 QMF for WebSphere*, SC19-2893
- ▶ *Installing and Managing DB2 QMF for TSO and CICS*, GC19-2886
- ▶ *Installing and Managing DB2 QMF for Workstation and DB2 QMF for WebSphere*, GC19-2892
- ▶ *Developing DB2 QMF Applications*, SC19-2887
- ▶ *Using DB2 QMF*, SC19-2888
- ▶ *QMF High Performance Option User's Guide for TSO and CICS*, SC19-2891
- ▶ *QMF Reference*, SC19-2889
- ▶ *DB2 QMF Messages and Codes*, GC19-2890

Online resources

These websites are also relevant as further information sources:

- ▶ DB2 Query Management Facility:
<http://www.ibm.com/software/data/qmf/>
- ▶ DB2 Query Management Facility Library:
<http://www.ibm.com/support/docview.wss?uid=swg27021603#manuals>

Help from IBM

IBM Support and downloads:

ibm.com/support

IBM Global Services:

ibm.com/services

Index

A

about xii, 13, 17, 41, 67, 89, 134, 176, 219, 266, 302
accessibility 66
accessing 49, 192
Add Calculated Column 212
adding repositories 79
administrator 60, 68, 71, 92, 127, 147, 300
Administrator perspective 78, 103
aggregating data 8
aggregation 9, 25, 67, 115, 200, 219–220, 338
alias 290, 293
alignment 43, 294
ALL 54
all 2, 17–18, 45, 48, 70, 81, 90, 95, 135, 137, 147, 154, 201, 205, 227, 231, 235, 257, 267, 289–290
alter 2
AND 55
API 56–57, 116
application 2, 18, 39, 68, 89, 158, 208, 231, 299
application development 5, 42, 117
application interfaces 45
application server 73
architecture 2–3, 17–18, 76, 86, 121
Arithmetic functions 384
AS 139
assigning 298
authorization 46, 100
authorization ID 112
Average 216

B

bind 95, 99, 102
bind packages 112
binding 112
BREAK 44
breaks 44
business intelligence xii, 2
Button 176

C

cache 19
CALC 44
calculated columns 67, 137, 211
calculations 43, 137, 236
CALL statement 41, 52
CALL statements 271
callable interface 46
cancel 60–61, 203, 353
carriage control 54
category 221
challenge 5, 18
change 4, 20, 43, 64, 67, 98–99, 105, 156, 160, 169, 217, 221, 230–231, 252, 257, 316, 321, 328

character 54
characteristics 20, 299
charts 5, 9, 66, 191, 210–211, 239, 289, 294, 299
check 71, 73, 99, 101, 108, 169–170, 235, 249, 328, 337, 359
CICS 5, 42, 79, 108, 152, 233, 265
classic reports 233
Close 20, 353
collection name 102
Color 253
Color functions 384
column 7, 41, 66–67, 119, 134, 152, 194, 230–231, 235, 308
Column chart 176
ColumnChart 331
Columns 140
Combo 292
command bar 52, 192
command interface 46
commands 41–42, 67, 117, 192, 200, 265
Comment 266
comments xiii
commit 116, 230–231
components 2, 4, 15–16, 54, 59–60, 70, 186, 244, 290, 299
concepts 65, 77
concurrent access resolution 42
condition 188, 190, 196, 198, 201, 213, 230, 261, 304
configuring 5, 95, 160
CONFIRM 58
CONNECT 47
CONNECT command 47
Connections 77, 123, 290
Connectivity 298
constants 200
content assist 201
control objects 298
Controls 292
Conversion function 385
CONVERT 43
Copy 74, 101, 139
cost 1–2, 17–18, 21, 39, 51, 65
CREATE 100
create 6, 9, 20, 22, 44, 46, 55, 57, 84–85, 89–90, 95, 133, 136, 138, 143, 146, 148, 152, 160, 173–174, 192, 201, 205, 216, 233–235, 265–266, 291–292
 procedure 55, 57, 266
 sample tables 174
 table 100, 136, 152, 304, 314
create new 23
Create New Visual Dashboard wizard 292
creating 7, 19, 47, 79, 103, 138, 206, 266, 290
CSV 199, 267
Cube Structure tree 223
customize 46, 66–67

D

- data xii, 1–2, 15, 39, 41, 66, 89, 133–134, 145, 156, 191, 227, 236, 239, 266, 289
 - access 1, 5, 16, 47, 66, 76, 89, 133–134, 156, 191, 227–228, 289
 - analysis 2, 22, 25, 117, 143, 292, 320
 - retrieval 134
- Data formatting functions 386
- data source 10, 49, 84, 86, 90, 138, 170, 204, 228, 267, 290, 292
- data source connections 82, 170, 303
- data sources 3, 10, 23, 74, 77, 89, 134, 139, 156, 204, 222, 228, 289, 292
- data types 357
- database xii, 2, 4–5, 18, 21–22, 24, 41–42, 67–68, 81, 84, 90, 133–134, 146–147, 192, 201, 227–228, 263, 290, 292
 - relational 24, 171, 204
- database access 116, 204
- database connection 96, 98, 101
- database objects 105
- DATAFORMAT 58
- date 13, 19, 51, 134, 143, 294, 320
- Date and time functions 387
- date parameters 356
- DateTime 345
- DB2 Connect 91, 95
- DB2 for z/OS 4, 17, 42, 90, 143, 158, 299
- decimal floating-point 41
- decision support 22
- default 41, 54, 70, 74, 76, 92, 98, 105, 150, 152, 155, 192, 194, 230, 258, 261, 293, 326, 332
- DELETE 43, 58, 101, 228
- Delete 175, 230
- delete 7, 67, 155, 168, 198, 227
- detail 5, 25, 39, 90, 134, 146, 193, 254, 291
- Details 176, 246, 327
- DISPLAY 41–42, 192, 273
- display 6, 9, 49, 54, 63, 77, 81–82, 122, 134, 192, 213, 219, 221, 229, 239, 244, 266, 292–293
- DISPLAY command 192
- Display Excel Sheet 215
- Display Report 236
- Display Report wizard 236
- DISTINCT 152
- distributed 14, 17, 19, 47, 86, 143
- DRAW 43, 49
- draw 4, 6, 143, 289
- DRAW command 49
- DRDA 90
- DROP 61
- drop 9, 83, 105, 107, 166, 175, 196, 237, 241, 328, 354
- dynamic reports 6
- dynamic SQL 62, 101

E

- Eclipse 76, 117
- Edit 105, 155, 228, 334
- edit codes 44, 46, 51

- editor window 267, 292
- elements 4, 19, 86, 187, 198, 238, 290
- e-mail 58
- embedded scenes 290
- Enable 41, 105, 147, 359
- end user 2, 4, 18, 42, 62, 66, 86, 111, 121, 168, 208, 289
- environment 3, 5, 14, 18, 23–24, 39, 45–46, 49, 79, 86, 116, 119, 130, 143, 147, 168, 266, 271
- environments 6
- ER diagrams 134
- ERDs 10
- errors 20, 54
- event actions 371
- events 20, 190, 291, 297
- Events view 362
- EXECUTE 102, 267
- Explore xiii, 292
- EXPORT 58, 67
- export 41, 201
- export data 67
- Expression 182, 296
- Expression Designer 383
- expressions 43, 180, 245
- extended 23, 41

F

- file 41, 67, 89, 199, 266
- filtering 24, 327
- find xii, 41, 67, 175, 203
- FIRST 24
- First 162
- fix pack 68, 102
- folders 131, 146, 292
- Font 213, 253
- footing 246, 255
- FORM 44, 239
- form xiii, 21, 42–44, 79, 98, 116, 143, 204, 208, 210, 233–234, 267, 275, 298
- FORM.COLUMNS 44
- FORM.MAIN 44
- Format 235
- format 1, 4, 24, 44–45, 115, 119, 136, 142, 213, 262, 267, 320, 332
- formatting 9, 43, 67, 211, 233, 267, 332
- forms 6, 41, 84, 108, 158, 236
- FROM 55–56, 152, 200
- FTP 56, 267
- function keys 42, 46

G

- global 1, 12, 23, 41, 54–55, 179–180, 228, 248
- global variables 43, 46, 179, 267
- governing 42, 60
- grant 102, 112, 143, 228
- graphic objects 294
- grid 8–9, 67, 203, 208, 211–212, 229–230, 235–236, 298, 320, 322, 326
- grouping 8–9, 25, 84, 127, 131, 209, 236
- Grouping and Aggregation 214

H

header 9, 188, 211, 246
heading 41, 45, 235, 246–247
heading text 250, 252
headings 67, 235
Hierarchy functions 388
High Performance Option 5, 39, 59
highlighting 254, 258
host 19, 57, 66, 90
host name 57, 98
hostname 57
HPO/Compiler 59
HPO/Manager 59
HTML 5, 57–58, 88

I

icons 71, 245
ID 45, 99, 146, 195, 271, 312
images 290
IMPORT 41
import 41, 149
importing 152
Information functions 388
Informix 10, 91
infrastructure 1–3, 16, 86, 90, 95, 143, 299
inner join 195
INSERT 43, 101, 152, 228
insert 293
install 11, 49, 65
installation 25, 42, 65, 89, 170
INTERACT 45
internal security 158
ISPF 46

J

JDBC 10, 74, 84, 90, 204, 227, 292
JDBC drivers 116
JDBC Libraries 94
JDBC libraries 94
job scheduler 66
join 7, 25, 117, 135, 194–195, 338, 352
join conditions 7, 207
Jump to new location 190

L

L2 54
Label 176, 251, 349
labels 254, 334
Last 41
Layout 67, 193, 263, 291, 294
layout objects 294
Layout tab 225
LDAP 146
LDAP security 157
legend 186
limits 84, 114, 147, 228, 266
Line 44, 256
linear procedure 45, 54, 265, 267

linear procedures 265
links 130
Linux 5, 16, 62, 68, 153, 267, 320
LIST 41, 117
List 180, 292
list 10, 48, 50, 60, 70, 75, 81, 94, 105, 107, 154–155, 166, 201, 223, 237, 241, 266–267, 292
local 47, 66, 84, 169, 171–172, 267
Logical functions 389
login mapping 173
Lotus 13–14, 119, 267

M

MAIL TO 267
main 5, 77, 222, 262, 290
manipulation 211
Math and trigonometric functions 389
Maximum 18, 216
Measured functions 390
measures 223, 323
MESSAGE 46
messages 54, 83
metadata 10, 19, 116, 133
Microsoft Excel 191, 215
middleware 17
multidimensional 4, 25, 89, 156, 191, 320
multidimensional data sources 114, 156, 321

N

names 43, 47, 68, 84, 91, 94, 135–136, 147, 201–202, 272, 312, 314
national language 52, 54, 69
navigation features 292
needs 3, 12–13, 20, 46, 66, 84, 102, 123, 173, 186, 289, 300, 309
new users 149
null 54
number 3, 6, 16–17, 21, 39, 44, 56, 77, 79, 91, 121, 123, 140–141, 168, 170, 192, 194, 199, 231, 265, 271, 359

O

object 12, 41–42, 103, 110–111, 153, 155, 192, 208, 211, 234, 256, 265, 293–294
object list 111
object names 110
Object REXX 265, 271
object type 131, 296
objects 5–6, 39, 66, 84, 89–90, 146, 153, 165, 167, 244, 265–266, 291–293
ODBC 94, 119
OLAP 10, 22, 114, 222, 299
OLAP queries 222
Open 46, 48, 80–81, 92, 95, 208, 228, 243, 265, 271, 292
operating system 45, 66, 89, 171–172, 265, 267
operator 196
operators 43
order 13, 20, 121, 190, 195–196, 275, 300
ORDER BY 58, 339

Output 83
Output view 83

P

packages 42, 95, 99–100, 168
palette 244, 292
palette objects 294
Palette view 250, 293
passwords 158, 170
path 22, 84, 205, 273, 298
PDF 267
performance 3–4, 16–17, 59
period 134
permissions 79, 84, 105, 146–147
personal repositories 175
Personal view 83
perspectives 48, 66, 92, 266, 292
PF keys 42
Plug-ins 108
port 74, 98
positive SQL codes 41
predefined 10, 45, 120, 155, 297
preview 254
PRINT 52, 54–55, 267
print 63, 67, 267
printing 46, 67, 267
procedure 5, 10, 41–42, 45, 67, 112, 192, 201, 265
procedure commands 266
procedures with logic 265
program parameters 46
programming 46, 67, 76, 227
Project Explorer 222, 244
Project Explorer view 303
prompted query 197
Properties 83, 151, 213, 228, 244, 273, 296
Properties view 252, 328

Q

QMF xii, 15, 19, 22, 39, 65, 89–90, 133, 145–147, 191, 227–228, 233, 265, 289–290
QMF catalog 49, 105, 147, 236, 266
QMF Classic perspective 10, 48
QMF for WebSphere 5, 41, 66, 118, 201, 265
QMF for Workstation 5, 39, 66, 90, 134, 152, 192, 228, 233, 265
QMF profile 54, 152
Queries 19, 43, 176, 206, 290
queries 4–6, 10, 18, 41–43, 77, 79, 81, 96, 101, 108, 134–135, 139, 158, 166, 177, 191, 193–194, 240, 262–263, 265–266, 290, 292
query 2, 5, 18, 41–42, 67, 119, 133, 135, 152–153, 155, 191–192, 228, 233–234, 267, 290, 292
query parameters 310
query results 42, 210, 235, 324

R

range 9, 24, 66, 191, 289
Redbooks website 395

Contact us xiii
REFRESH 41
Refresh 115
registers 41
relational 24, 89, 172, 191, 299
relational data sources 114
relational queries 225
Rename 134
replacing 202
report 20, 42–43, 66–67, 79, 108, 135–136, 143, 146, 192, 201, 208, 234, 267
reporting 2, 4–5, 22, 42, 59, 134, 141
reports 5–6, 10, 41–42, 45–46, 66, 77, 79, 127, 131, 134–135, 143–144, 146, 174–175, 191, 205, 219, 233, 265, 271
Repositories view 96
repository 22, 49, 84, 89, 137, 145, 265
repository connections 169
repository objects 266
repository storage 84, 89, 147
resource limits 114, 147, 149, 228, 266
resource limits provider 114
response 294, 296
result 2, 9–10, 22, 42, 45, 52, 67, 116, 121, 143, 149, 197–198, 201, 234, 239, 267, 291, 294
Results tab 197, 230, 324
REXX 43, 45, 67, 265
REXX Console 82, 267
REXX Console view 82
Rollback 231
rollback 116
row conditions 43, 199
RUN 45, 56, 58, 265–266, 272
run xii, 7, 10, 12, 39, 45, 67, 101, 111, 130, 153, 155, 168, 192, 196, 200, 208, 239–240, 254, 265–266, 298, 332
run a query 52
Run Query 7, 50, 196
running 11, 17, 62, 68, 100, 142, 146, 199, 228, 233, 267
RUNTSO 52

S

sample 7, 54–55, 174, 176, 178, 194, 205, 229, 247, 263, 267–268, 292
samples 175
SAVE 41, 43, 54, 139, 267
save 43, 137, 139, 176, 217, 298, 344
saving 170, 298
scene 176, 290
scene parameters 344
scenes 43, 168, 176, 290
schedule 66, 265, 275
scheduling 66, 275
security 3–4, 17, 46, 105, 145, 228
Security function 390
security lists 176
SELECT 43, 55–56, 101, 111, 152–153, 200
select 5, 7, 74, 83, 94–95, 137, 140, 153, 158, 196, 206, 228, 235, 237, 266, 293, 298
selected columns 194

- series 2, 23, 175, 217
- server 3, 16–17, 19, 47, 58–59, 68, 73, 86, 98, 105, 107, 115, 144, 166–167, 192, 267
- server definition file 103
- server name 58
- SET 19, 43, 56, 100
- set 2–3, 5, 16, 22, 43–44, 49, 84, 95–96, 133, 147, 150, 152, 195, 197–198, 230–231, 233–234, 266–267, 296
- SET PROFILE 55
- Set User Information 108
- Set values 347
- setting up 73, 112
- setup 25, 68, 95, 107, 169–170, 267
- shapes 294
- SHARE 54
- shared 12, 20, 84, 89, 146–147, 158, 221, 267, 290
- SHOW 42–43
- SHOW command 44
- Side Group 235
- Slicer 327
- SMTP 58
- solution 2, 16, 18, 39, 65
- sort 43, 195–196, 198
- sort conditions 199
- sorting 8, 24, 196
- SPACE 58
- Spatial functions 391
- spill file 41
- SQL 10, 14, 19, 24, 41, 83, 90–91, 140, 142, 168, 193, 227–229, 297, 315, 320
 - expression 212
 - static 101
 - verbs 60–61
- SQL codes 41
- starting 86, 99, 136, 194
- STATE 343
- static SQL 62, 101
- Statistical functions 391
- status 53–54, 83, 123
- stored procedure 45, 201
- stored procedure interface 51–52
- stored procedures 45, 62, 101, 201
- structure 3–4, 19, 21, 42, 192, 209, 322–323
- SUBJECT 58
- subqueries 200
- Sum 235
- switching 175
- syntax 43, 107, 200
- SYSIBM.SYSUSERAUTH 111
- system 2, 15, 45, 66, 89, 143, 169, 204, 265, 294
- system requirements 67

T

- TABLE 41, 61
- table 21, 24, 45–47, 67, 86, 117, 131, 133–134, 152, 192, 227, 267, 312–313
- table data 188
- table editing 227
- Table Editor 43, 228
- tables 4, 7, 18, 21, 43, 60, 67, 84, 86, 89, 96, 100, 134,

- 158–159, 166, 192, 194–195, 267, 292, 294
- TCP/IP 117
- Text functions 392
- thin client 3, 5, 66, 73
- time 1, 4, 15, 39, 43, 89, 91, 143–144, 149, 152, 197, 211, 228–229, 244, 265, 271, 294
- time data 20
- Time Period 338
- TIMESTAMP data type 41
- title 176, 221, 255
- toolbar 7, 9, 81, 192, 292
- totals 134, 215, 235, 336
- Trace 54
- tracing 46
- TSO 5, 39, 41, 79, 108, 152, 233, 265–266
- TSO command 56
- TSO environment 56
- TYPE 58
- types 13, 42–43, 78, 84, 106, 168, 227, 294, 357

U

- unit of work 47
- UPDATE 62, 101, 117, 228
- update 20, 62, 110, 316
- user interface 42, 60, 63, 76–77, 79, 91–92, 153, 244, 289, 292
- User perspective 79, 126, 266
- user preferences 292
- UserName 84
- users and groups 46, 153

V

- variables 24, 43, 46, 57, 179, 266–267
- view 2–3, 23–25, 43, 47–48, 77, 82, 93, 134–135, 153, 155, 193–194, 244–245, 266–267, 292–293
- views 22, 25, 66, 100, 111, 193, 200, 292
- views and perspectives 66
- virtual 19, 41, 117, 133, 156, 300
- virtual data source 138, 300
- virtual data source editor 139
- visibility 25, 186, 360
- Visual Designer 79, 121, 243
- Visual Designer perspective 243, 291
- Visual report functions 393
- visual reports 233

W

- web 23, 67, 71, 73, 115, 221, 292
- WHERE 55, 178, 229, 267
- Windows scheduler 276
- work with 20, 41, 47, 49, 89–90, 134, 254, 292
- working with xii, 24, 67, 134, 228, 234, 339
- workspaces 83–84, 89, 105, 130, 147, 150, 273, 298

X

- XML 41, 75, 115
- XML data 41
- XY chart 294



Complete Analytics with IBM DB2 Query Management Facility Accelerating Well-Informed Decisions Across the Enterprise

(0.5" spine)
0.475" <-> 0.873"
250 <-> 459 pages



Complete Analytics with IBM DB2 Query Management Facility

Accelerating Well-Informed Decisions Across the Enterprise



Redbooks®

Gain actionable insight with visual reports and interactive dashboards

Deploy cost-effective, self-service analytics with zero coding

Leverage existing System z applications in new ways

There is enormous pressure today for businesses across all industries to cut costs, enhance business performance, and deliver greater value with fewer resources. To take business analytics to the next level and drive tangible improvements to the bottom line, it is important to manage not only the volume of data, but the speed with which actionable findings can be drawn from a wide variety of disparate sources. The findings must be easily communicated to those responsible for making both strategic and tactical decisions. At the same time, strained IT budgets require that the solution be self-service for everyone from DBAs to business users, and easily deployed to thin, browser-based clients.

Business analytics hosted in the Query Management Facility (QMF) on DB2 and System z allow you to tackle these challenges in a practical way, using new features and functions that are easily deployed across the enterprise and easily consumed by business users who do not have prior IT experience. This IBM Redbooks publication provides step-by-step instructions on using these new features:

- ▶ Access to data that resides in any JDBC-compliant data source
- ▶ OLAP access through XMLA
- ▶ 150+ new analytical functions
- ▶ Graphical query interfaces and graphical reports
- ▶ Graphical, interactive dashboards
- ▶ Ability to integrate QMF functions with third-party applications
- ▶ Support for the IBM DB2 Analytics Accelerator
- ▶ A new QMF Classic perspective in QMF for Workstation
- ▶ Ability to start QMF for TSO as a DB2 for z/OS stored procedure
- ▶ New metadata capabilities, including ER diagrams and capability to federate data into a single virtual source

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-8012-00

ISBN 0738437018