



IBM Informix: Integration Through Data Federation

Managing information and protecting
development assets

Integrating heterogeneous
sources of data

Data federation and join
optimization



Chuck Ballard
Nigel Davies
Marcelo Gavazzi
Martin Lurie
Jochen Stephani



International Technical Support Organization

IBM Informix: Integration Through Data Federation

August 2003

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

First Edition (August 2003)

This edition applies to DB2 Information Integrator Version 8.1, Informix Enterprise Gateway Manager Version 7.3.1, Informix Dynamic Server Version 9.4, Informix Extended Parallel Server Version 8.4, IBM Red Brick Version 6.2, DB2 Version 8.1, and Oracle9i.

© Copyright International Business Machines Corporation 2003. All rights reserved.

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

Contents

Notices	vii
Trademarks	viii
Preface	ix
The team that wrote this redbook	xi
Become a published author	xiv
Comments welcome	xiv
Introduction	1
Executive summary	7
Chapter 1. Data federation overview	13
1.1 The challenge of information integration	14
1.2 To federate or not to federate	19
1.3 Data federation examples	23
1.3.1 Data federation with DB2	25
1.3.2 Data federation with Informix	26
1.4 Considerations	28
1.4.1 Naming conventions	28
1.4.2 Data types	29
Chapter 2. The data federation project	31
2.1 Environment and server configuration	33
2.1.1 Client configuration	34
2.1.2 Database connectivity using DB2 Information Integrator	35
2.1.3 Database connectivity using Informix EGM	36
2.1.4 Other configuration options	36
2.2 Schema definition	38
2.3 Naming conventions	43
2.3.1 Database servers	44
2.3.2 Database identifiers	44
2.3.3 References to remote tables within the federated database	45
2.3.4 User IDs	45
Chapter 3. Overview of project data sources	47
3.1 DB2 UDB	49
3.1.1 DB2 UDB architecture	49
3.1.2 DB2 UDB functions and features	52
3.2 Informix Dynamic Server (IDS)	53

3.2.1	IDS architecture	55
3.2.2	Functions and features	57
3.3	Informix Extended Parallel Server (XPS)	61
3.3.1	XPS architecture	62
3.3.2	Functions and features	63
3.4	IBM Red Brick Warehouse	65
3.4.1	Red Brick architecture	66
3.4.2	Functions and features	68
3.5	Oracle9i and Microsoft Excel	70
Chapter 4. DB2 Information Integrator		73
4.1	Product overview	74
4.1.1	Systems environment	74
4.1.2	Components and packaging	81
4.2	Installation and configuration	82
4.2.1	Pre-installation requirements	82
4.2.2	Installation instructions	83
4.2.3	Post-installation tasks	93
4.3	Creating and using wrappers and nicknames	97
4.3.1	Create a wrapper	98
4.3.2	Define a server	99
4.3.3	Define user mapping	102
4.3.4	Create and authorize nicknames	104
4.3.5	Testing the connection to a data source	107
4.4	Considerations for use	108
4.4.1	Statistics and index specifications	108
4.4.2	Informix synonyms and I-STAR	111
4.4.3	Transaction support	113
4.4.4	User mapping authentication for Informix data sources	113
4.4.5	Pass-through	114
4.4.6	Data type mapping	114
4.4.7	Server options	116
4.4.8	Capacity planning for DB2 Information Integrator	119
4.5	Troubleshooting	119
4.5.1	Connecting to data sources	119
4.5.2	Error messages	120
4.5.3	Informix environment settings and DB2DJ.INI file	120
4.5.4	Capturing server options and nickname definitions	121
4.5.5	Access plans and the query optimizer	123
4.5.6	Diagnostic information to collect	124
4.5.7	Tracing DB2 Information Integrator wrapper calls	126
Chapter 5. Informix Enterprise Gateway Manager		131

5.1	Product overview	132
5.1.1	Systems environment	135
5.2	Installation and configuration	139
5.2.1	Pre-installation requirements	139
5.2.2	Installation instructions	140
5.2.3	Configuration tasks	142
5.3	Considerations for use	149
5.3.1	Using non-Informix SQL statements	149
5.3.2	Object names in queries	150
5.3.3	Using views versus synonyms	150
5.3.4	Using Informix 4GL	151
5.3.5	Distributed transactions	151
5.3.6	Multiple data sources within a single application	151
5.3.7	Effect of accuracy of statistical information	151
5.3.8	Effect of Informix style system catalog	152
5.3.9	Oracle date and datetime conversion	152
5.3.10	Using cursors	153
5.3.11	Temporary files for cursor handling	153
5.3.12	Limiting the number of rows returned in a query	153
5.4	Troubleshooting	154
Chapter 6. Data federation in action		159
6.1	Test application examples	160
6.1.1	Query 1 - Summarizing regional data	160
6.1.2	Query 2 - Federated views	163
6.1.3	Query 3 - Consolidating the data	167
6.2	Testing with DB2 Information Integrator	169
6.2.1	Join and insert testing with DB2 II	170
6.3	Testing with Informix Enterprise Gateway Manager	181
6.3.1	Join and insert testing with EGM	182
6.3.2	Updating multiple data sources	190
Chapter 7. Optimization in a federated environment		193
7.1	Performance options and considerations	194
7.2	Remote data source tuning	194
7.3	Server options in DB2 II	194
7.3.1	Pushdown	195
7.3.2	Maximal pushdown	199
7.3.3	Collating sequence	200
7.4	Use of Materialized Query Tables (MQTs)	204
7.5	Remote data source catalog statistics	209
7.6	Optimizer hints in query text	212
Chapter 8. Hints and tips		217

8.1 Federated inserts with Informix	218
8.2 Using DATE columns for a Union operation	219
8.3 Use of current schema with DB2 Interactive Explain	219
8.4 DB2 problem determination series tutorial	220
8.5 Losing connection to a data source.	220
8.6 Using views versus synonyms with data federation	220
Appendix A. DB2 II and Informix EGM function summary	225
Data federation summary table	226
Appendix B. Nonrelational wrappers.	229
Microsoft Excel wrapper	230
XML wrapper	232
Glossary	237
Abbreviations and acronyms	239
Related publications	241
IBM Redbooks	241
Other publications	241
Online resources	242
How to get IBM Redbooks	242
Help from IBM	243
Index	245

Notices

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

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

Trademarks

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

AIX®	@server™	NetVista™
CT™	@server™	Notes®
DataBlade™	Informix®	OS/390®
DataJoiner®	IBM®	Red Brick™
Distributed Relational Database Architecture™	ibm.com®	Red Brick Vista®
DB2 Universal Database™	IMS™	Redbooks™
DB2®	iSeries™	Redbooks (logo)  ™
DRDA®	Lotus Notes®	Tivoli®
	Lotus®	z/OS™

The following terms are trademarks of other companies:

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

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

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

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

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

Preface

This IBM® Redbook is primarily intended for use by IBM Informix® Customers and Business Partners, but much of the information in this publication can be used in any data federation implementation. The purpose is to demonstrate how your heterogeneous sources of information can be federated and enable the implementation, use, and maintenance of a robust, managed information environment.

The information in this publication will help you understand how the IBM Informix database family of products can better be used together, and with other information sources. The primary focus is on moving towards an integrated information environment. There are several ways to provide this type of environment, and we will discuss those alternatives.

We demonstrate how you can start down the path towards an integrated information environment through the use of data federation technology. There are two primary tools available to help you along, and we will discuss them and their capabilities in later chapters. Those tools are DB2® Information Integrator and Informix Enterprise Gateway Manager.

We have implemented Informix Dynamic Server (IDS), Informix Extended Parallel Server (XPS), IBM Red Brick™ Warehouse, DB2 UDB, and Oracle9i, in a federated data implementation, by using DB2 Information Integrator and Informix Enterprise Gateway Manager. Then we provide examples of the capabilities of those two data federation products, and compare them at a high level to give you an understanding of the capabilities of both.

In addition to data federation and integration, there are other related topics of interest in this publication. The following is a brief description of the topics and how this publication is organized:

- ▶ “Introduction” on page 1 provides a high level overview of the contents of this publication. It enables the reader to get a basic understanding of the contents and conclusions presented in this publication, without the requirement of reading all the detailed and technical supporting information.
- ▶ Chapter 1, “Data federation overview” on page 13, gives an overview of the topic of data federation. It explains why it is strategic and describes the benefits. Some best practices are provided to help guide you through planning for and implementing your federated data environment. It can deliver significant advantages and enable you to get faster and easier access to your information assets.

- ▶ Chapter 2, “The data federation project” on page 31, describes the project environment and the systems architecture used during the development of this publication. It defines the hardware and software products used, and the operating systems. It also describes the database and data schema used for our testing. We used the Informix Stores Demo database as the basis, which is typically very familiar to Informix customers. However, the schema was modified somewhat to provide the tables and data distribution required in our federated environment. The sample application that is part of the Stores Demo database was used to simulate a real production environment for our testing.
- ▶ Chapter 3, “Overview of project data sources” on page 47, presents an overview of products used for our federated data environment. This chapter is primarily for those not familiar with the database management systems (DBMS) used. We take a brief look at the architecture, and some of the features and functions that will be valuable in configuring the databases for use in a federated environment. The DBMS’s used were DB2 UDB, Informix Dynamic Server, Informix Extended Parallel Server, IBM Red Brick Warehouse, and Oracle9i. We also demonstrated data federation with Microsoft Excel and XML files. This topic is discussed in Appendix B, “Nonrelational wrappers” on page 229
- ▶ Chapter 4, “DB2 Information Integrator” on page 73, is an overview of the DB2 Information Integrator product. It became generally available this year and is well positioned to help access, manipulate, and integrate the many data sources present in most typical companies. We cover the architecture, some functions and features, and instructions for installing and configuring the product. In addition, we provide some considerations for its use to help in your implementation project.
- ▶ Chapter 5, “Informix Enterprise Gateway Manager” on page 131, is an overview of the Informix Enterprise Gateway Manager (EGM) product. This is a product with similar functionality to the DB2 Information Integrator, that has been available from Informix. We describe the architecture, some of the functions and features, and instructions for installing and configuring the product. In addition, we provide some considerations for its use.
- ▶ Chapter 6, “Data federation in action” on page 159, takes you through some of the testing we did with the data federation products. We describe the sample queries used and present the results of our tests. The information primarily centers around join execution in a heterogeneous data source environment. Most of the tests were performed with both the DB2 Information Integrator and the Informix Enterprise Gateway Manager products, as there is interest in both products.
- ▶ Chapter 7, “Optimization in a federated environment” on page 193, takes the testing in Chapter 6, “Data federation in action” on page 159, a step further. In Chapter 6 we focused on the capability of join processing. This chapter

focuses on how well, and how fast, those joins can be performed. The objective here was to test the optimization capabilities of the products. This is a very important consideration that can impact resource utilization, performance, and response times.

- ▶ Chapter 8, “Hints and tips” on page 217, presents some hints and tips to help you as you implement your federated environment. These are typically situations and issues we encountered in our installation and testing. We provide you with the solution or workaround to help make your implementation easier and faster.
- ▶ Appendix A, “DB2 II and Informix EGM function summary” on page 225, presents a summary table of the high-level functions and features desirable in a data federation environment. Then we list if, and how well, the DB2 Information Integrator and Informix Enterprise Gateway Manager products provide that capability. This helps clarify some of the basic capabilities of each product.
- ▶ Appendix B, “Nonrelational wrappers” on page 229, presents information on two of the DB2 Information Integrator wrappers that can be used for nonrelational data sources. In particular, those are Microsoft Excel and XML files. In each case we demonstrate how data from relational tables and these nonrelational files can be joined together as if the data were from the same data source.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world. Four members worked at the International Technical Support Organization, in San José, California and one worked remotely in Boston, Massachusetts.



Figure 1 Left to right: Nigel Davies, Marcelo Gavazzi, Chuck Ballard, Jochen Stephani

Chuck Ballard is a Project Leader at the International Technical Support Organization, in San Jose, California. He has over 35 years of experience, holding positions in the areas of product engineering, sales, marketing, technical support, and management. His expertise is primarily in the areas of database, data management, data warehousing, business intelligence, and process re-engineering. He has written extensively on these subjects, taught classes, and presented at conferences and seminars worldwide. Chuck has both a Bachelors degree and a Masters degree in Industrial Engineering from Purdue University.

Nigel Davies is an IT Architect working at the Cumberland Forest Application Centre in Sydney for IBM Global Services Australia. Although now living and working in Australia, he is originally from the United Kingdom, where he earned his Masters degree in Chemistry at Oxford University. He has over 22 years of experience in the IT industry. During this time he has worked as a programmer, designer, business analyst, DBA, and even project manager, before finally settling down as an IT architect. Nigel has been working with DB2 since 1987 and has previously published articles on DB2 performance for the UK trade journal *Info Plus*. He has worked for IBM GSA for seven years.

Marcelo Gavazzi is an Advisory Software Engineer within the IBM Software Group organization working at the Latin America Call Center in Miami, Florida. He is originally from Brazil and holds a degree in Computer Sciences from Universidade Mackenzie. He has more than ten years of experience supporting RDBMS systems, including Informix database servers and DB2 UDB. Marcelo is a certified “Informix Dial-up engineer” for interventions on critical systems throughout the Latin America region, providing diagnostics and fixes in the emergency support line. He is also involved in training projects disseminating new technologies to the technical support community. He holds professional certifications on both Informix and DB2 UDB products.

Jochen Stephani is a DM Consultant/IT Specialist working on the Technical Sales Team in IBM’s Information Management Division in Munich, Germany. He joined Informix in April 1996 and has more than seven years of experience working with Informix databases on high-end OLTP and business intelligence projects, including benchmarks, prototyping, and performance tuning. He has contributed to many important projects across the region and is a ‘trusted



advisor’ for strategic customers. Based on his experience and product skills, his major focus as a DM Consultant is to provide architectural guidance. He holds a degree in Computer Sciences from the Fachhochschule of Rosenheim.

Martin Lurie (pictured at left) was the remote team member. Marty is a Systems Engineer in IBM’s Data Management Division, and is located in Boston, Massachusetts. He is an IBM-certified DB2 DBA, IBM certified Business Intelligence Solutions Professional, and an Informix-certified Professional.

Other Contributors

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

Omkar Nimbalkar - Informix Product Management and Marketing
Dwayne Snow - Product Manager, Informix Dynamic Server
Patricia Quinn - Brand Manager, Informix Dynamic Server
Mohan Natraj - WW Brand Manager, XPS and Red Brick Warehouse
IBM Informix Marketing, Support, and Development

Benjamin Wilde - DB2 II Development Manager
Walter Alvey - DB2 DataJoiner® Development
Kenneth Gee - Red Brick Warehouse Support

Yang Jason Sun - UDB Query Compiler
Jacques Labrie - DB2 II Scenario Testing
Yannick Barel - Information Integration Technology Solutions
Josette Huang - Manager, Information Integration
Glenn Smith - Information Integration Install Developer
Joe Baginski - Manager, Information Integration Build and Install
IBM Silicon Valley Development Lab

Kenro Yamagata - Information Management Technology Development
Yamato Software Development Lab - IBM Japan

Christina Hanna - Brio IT Specialist
Cumberland Forest Application Centre, Sydney Australia

Mary Comianos - Operations and Communications
Deanna Polm - Residency Administration
Emma Jacobs - Graphics Designer
International Technical Support Organization, San Jose Center

Julie Czubik - Technical Editor
International Technical Support Organization, Poughkeepsie Center

Become a published author

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

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

ibm.com/redbooks

- ▶ Send your comments in an Internet note to:

redbook@us.ibm.com

- ▶ Mail your comments to:

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



Introduction

Information technology is constantly changing, and improvements in hardware and software are coming at an ever increasing pace. At the same time, the business environment is also changing rapidly. For example, the results of the many acquisitions and mergers are having a dramatic impact on those businesses. This is because each business typically brings with it different business processes, hardware, software, and applications, and they are all incompatible with each other. The burden of keeping it all working, so the business can continue to function, falls primarily on the Information Technology (IT) organizations.

This fast moving, heterogeneous, and ever growing type of business environment becomes a stumbling block in making the acquisition, merger, or move to a new technology successful. Because business cannot afford to stop doing business until all the processes are integrated and working smoothly, it can be a huge challenge. It can impact day-to-day business processes, the business transaction processing systems, and the population of data warehouses that are so critical in helping management make timely and informed decisions.

Data federation to the rescue

Many businesses are now looking to data federation technology to remove this stumbling block. Data federation is not really new—the concepts have been around for some time. However, the technology has not been there to support it. That technology is here now, and IBM has it. There is a brief overview that

follows, but for more detailed information please read Chapter 1, “Data federation overview” on page 13.

Defining data federation

Data federation is a process that enables data from multiple heterogeneous data sources to appear as if it is contained in a single relational database. We use the terms data federation and information integration in this publication. However, we do not consider these two terms to mean the same thing. In this publication we primarily focus on the topic of data federation.

While we are discussing terms, be aware also that the terms *information integration* or *data integration* are typically used rather loosely around the world. You must always insist on definitions of these terms rather than accepting them at face value. There can be varying levels of integration, but any form of integration is typically quite difficult, resource intensive to achieve, and quite expensive. But, depending on your definition, it could be said that data federation is actually some level of information integration.

Data federation uses a single language to access all data sources, and that is SQL. So, all data can be accessed and used from an application program, tool, or program product using the same relational SQL. This means that companies can begin moving to a standardized data access language for all their data sources, which can mean a huge productivity boost for them. It also means huge savings in finding, training, and maintaining skilled programming resources. This standardization should result in a reduction in the number of different skilled resources that a company will require to manage, maintain, and use their legacy data.

How data federation works

Each heterogeneous data source is defined and described to the data federator, as virtual tables. The data sources can reside anywhere, so that definition contains the access path to the real location of the data. There is also a mechanism that transforms the federator SQL into the language used by the real data source. In the case of the DB2 Information Integrator product, that mechanism is called a *wrapper*.

However, there is more to data federation than simple data access. Data from these multiple heterogeneous data sources typically will need, as examples, to be joined together, aggregated, and summarized. This is not a simple task, especially when the data sources reside in distributed systems in different operating environments. Also, in addition to accessing the data, performance and resource requirements must be considered. For example, where will these operations take place? Will all the data have to be transferred back to the federator? What will be the impact on the network?

All of these concerns must be addressed by the federator if it is to provide adequate performance and result in the savings promised from reduced resource requirements. These types of concerns and considerations are discussed in various chapters of this publication. In general, they all fall into a category we refer to as *optimization*. That is, we need to optimize the access and use of the heterogeneous data sources so it will satisfy the performance and response time criteria of our users.

What is included in this redbook

This publication contains information on the topic of data federation and information integration. It is primarily oriented to users of Informix database management systems, which includes Informix Dynamic Server, Extended Parallel Server, and Red Brick Warehouse. We wanted to provide information that would describe how they can implement data federation and gain the benefits from it. So, we define and position data federation and information integration with respect to this particular set of users.

To do so, we have included other data sources that are also typically implemented by this set of users. For example, we also include Oracle9i, DB2 UDB, and Microsoft Excel. We discuss the concepts of, and considerations for using, data federation. The benefits are many and are discussed in numerous books and articles on the subject. We present some examples of those benefits in this publication just to give you a feel for why you should consider implementing a federated data environment.

We implemented a federated data environment during the development of this publication and ran numerous tests on it to understand its behavior. We discuss the results of that testing in the chapters that follow. We also provide considerations for you as you prepare to implement your federated environment. In addition to the obvious benefits of being able to work with data from heterogeneous sources, there are many other tangible benefits to data federation. For example, it can enable you to begin standardizing your application development efforts. This can be a huge benefit. It impacts development times, the type of development resources needed, and the availability of skilled resources.

Data federation is described and discussed in later chapters of this publication, so we will keep the discussion here very short. Data federation is simply a means of enabling you to access and use data from heterogeneous sources as if they were all the same source. Think about the savings that means to you and your application development organization. Think about what it means for easing the burdens during a merger or acquisition. Think about the effort it can save as you move to new hardware and software technologies.

All data federation implementations are not alike. You must understand the concepts, but you also must understand the criteria that can enable you to evaluate the candidates. You must come to know the functions, features, performance, ease of use, resource requirements, and future product direction, to select wisely. Once you know all this, we think you will select DB2 Information Integrator.

The project systems environment

In this publication we give you information to help in the installation, configuration, use, and management of a federated data environment. With a product such as DB2 Information Integrator (II); for example, you can access data from many data sources, such as DB2, Informix, Microsoft, ODBC, Oracle, Sybase, Teradata, OLE DB, Excel, XML, message queues, Web services, flat files, life sciences data sources, and IBM Lotus Extended Search sources such as content repositories, LDAP directories, WWW, e-mail databases, and syndicated content. The good news is that you can access all these sources as if they were the same source. That can represent a huge savings in development time, requirement for skilled resources, and conversion or migration costs.

There are two products available from IBM that can participate in a federated data environment. We chose to test both of these products and document the results in this publication. They are:

- ▶ DB2 Information Integrator (II)
- ▶ Informix Enterprise Gateway Manager (EGM)

However, in addition to the above products there is an additional capability for federating data in an Informix database environment, called I-STAR. This capability is built into the Informix Dynamic Server and Extended Parallel Server. Therefore, when these two data sources are used together, I-STAR will typically be used to perform the data joins. Also, we have demonstrated that even when these two data sources are used in conjunction with other heterogeneous sources, I-STAR may be invoked during the federation of the data. This verifies yet another level of federation that is possible in an Informix database environment. For a definition of I-STAR refer to “Informix I-STAR Technology” on page 27.

We configured a test environment that would appropriate for most IBM Informix customers. That environment consisted of the following five database management systems:

- ▶ Informix Dynamic Server V9.4
- ▶ Informix Extended Parallel Server V8.4
- ▶ IBM Red Brick Warehouse V6.2
- ▶ DB2 UDB V8.1
- ▶ Oracle9i

The tests were to demonstrate that the heterogeneous data sources could be accessed and joined together with the federation products. We also explored the architectures of those products and their capabilities. There is a function/feature summary table in Appendix A, “DB2 II and Informix EGM function summary” on page 225. This information will help you evaluate their capabilities and determine which you can best use to work with the many other sources of data that are out there—especially those already in your own organization.

To help you get you started towards implementing a federated environment, we implemented the products ourselves so we could provide you with some guidelines and instructions to make your implementation easier and faster. For our implementation, we used three Intel-based IBM servers, each with 760 MB of memory. They had the following software configuration:

- ▶ Server 1 was running SuSE Linux V8.1 with the following software:
 - Informix IDS V9.4
 - Informix Extended Parallel Server V8.4
 - IBM Red Brick Warehouse V6.2
 - DB2 UDB V8.1
- ▶ Server 2 was running Windows/2000 with the following software:
 - Informix IDS V9.4
 - DB2 UDB V8.1
 - Oracle9i
 - DB2 Information Integrator V8.1
- ▶ Server 3 was running Windows/2000 with the following software:
 - Informix Enterprise Gateway Manager V7.31

Servers 1 and 2 were primarily database servers. We used Server 3, a Windows/2000 server, to run Informix Enterprise Gateway Manager. Although it could have run on Server 1 or Server 2, we configured a third server as a test variable. We wanted to demonstrate a feature of this product that enables it to run on a separate processor that has no local RDBMS. This feature worked properly and there were no problems encountered that were a function of running on this separate server.

Rather than writing sample applications to test the federated environment we decided to use existing query products. The schema for the Informix Stores Demo database that we used is configured to support a Stores Application, so that was all that was necessary for us to meet our objectives. The primary objective of the tests was to demonstrate the capabilities of DB2 Information Integrator and Informix Enterprise Gateway Manager for accessing and joining heterogeneous data sources. To do this we simply needed to issue appropriate queries against the federated environment and view the results. In addition we needed to explore the query optimization capabilities of DB2 Information

Integrator and Informix Enterprise Gateway Manager, and their ability to rewrite the query plans. These products also enabled us to view those new query plans to validate any optimization that took place.

To query the federated environment and validate optimization, we used the following three products:

- ▶ Server Studio Java Edition for Informix
- ▶ DB2 Command Center
- ▶ Brio Explorer V6.5.2

For all the specific details on the project environment, refer to Chapter 2, “The data federation project” on page 31. Here you will also discover the reasons behind our selections for the environment.

Using these products simplified the data access and enabled fast development and execution of the test queries. A sample database, the Stores Demo database, comes with IDS and was used as the test vehicle for the implementation. It is a database, and application scenario, familiar to most Informix customers. Using it should promote easier understanding of the results of our tests. Using this environment provided us a means of demonstrating how quickly solutions can be developed and implemented, even when using multiple heterogeneous data sources. It truly demonstrates the value of a federated data environment.

We believe this publication will provide, in one document, the information you need to begin implementing and testing a powerful federated data environment. It will support the movement to a more standard development environment, saving you time, money, and minimizing the need for skilled development resources on all of those data sources.

The DB2 Information Integrator has demonstrated the capability to support a heterogeneous data environment and provide improved performance with the use of its powerful optimization capabilities. Once you have tried it, we believe you will realize its power, ease of implementation and use, and significant savings. You will then be on a path towards a more standardized and highly productive development environment.

Executive summary

“Introduction” on page 1, discusses the need for information integration. Companies, and perhaps yours included, have grown through differing data management technologies over the years, which has left them with a legacy. That legacy includes:

- ▶ Lots of data of many different types and in many different formats
- ▶ Data created by and residing in many different applications
- ▶ Data developed in several data management technologies
- ▶ Data stored in a number of different data management products
- ▶ Data management products purchased from a number of different vendors
- ▶ Data management products installed that are no longer supported or available
- ▶ Numerous different data schemas
- ▶ Data in many locations around the world
- ▶ Data from mergers or acquisitions
- ▶ Data with the same, or similar, names but having different meanings
- ▶ Data entities, such as customer, vendor, or company, that are the same but spelled or named differently

All this adds significant costs to your organization to maintain, manage, and use. It is because of all this that information integration is needed. Also, there is significant savings associated with it.

In this publication we explore and discuss information integration. It can come in many flavors with differing capabilities. Enterprise information integration is not an easy or a short-term proposition. However, it is a direction in which you should be moving. The volume of data an organization collects is growing at a significant rate, particularly with the advent of the Internet. To accurately analyze it and extract the value of informed decision making from it, requires you to manage it.

Implementing information integration, of course, requires supporting technology and products. One approach that is demonstrating significant capabilities and benefits is data federation, and that is the main focus of this publication. We discuss and demonstrate data federation as implemented by two IBM data federation products:

- ▶ DB2 Information Integrator (DB2 II)
- ▶ Informix Enterprise Gateway Manager (EGM), in conjunction with the Informix Database Servers

We have purposely restricted our scope of the project to federation of relational data sources. Our project environment and architecture is defined in detail in Chapter 2, “The data federation project” on page 31.

To make this exercise more realistic, we have constructed a case study centered around a fictitious company based in the United States. This fictitious company has the states organized into seven regions. Each region uses the same database schema, but has implemented it using a different database server or operating system. Most businesses hopefully would be less complex than this. However, we wanted to demonstrate the robust data federation capabilities that are available to you. We have used the Informix Stores Demo database schema for our case study. That is a sample database supplied with Informix Dynamic Server (IDS) and Informix Extended Parallel Server (XPS). Most Informix users are already familiar with this schema, which should aid the understanding of our case study.

We use this case study to explore the data federation technology available today from IBM. We discuss the strengths and weaknesses, and provide insight into to how it works and how to leverage the technology to your benefit.

Data federation technology is certainly impressive, giving you a great start down the path to fully realize the vision of information integration, as expressed in 1.1, “The challenge of information integration” on page 14.

The technology today requires some further development in the area of distributed update capability across heterogeneous platforms. However, in both the DB2 and Informix family of products, data integrity is assured for updates across distributed databases using a two-phase commit strategy. In the current releases of both DB2 II and EGM, there are limitations in the way updates can be performed across heterogeneous distributed databases. These limitations are discussed in 4.4, “Considerations for use” on page 108, and 5.3, “Considerations for use” on page 149.

The purpose of this publication is to explore the topic of data federation and demonstrate how it can be used by companies with Informix, and who also have other heterogeneous sources of data.

We created a federated data environment in a systems configuration built on Linux and Microsoft Windows/2000 operating systems. To enable the federated environment, we used the DB2 Information Integrator and Informix Enterprise Gateway Manager. These two products provide a number of similar capabilities, but have dissimilar architectures. We have documented the results from our testing and have used them to see how well they provide they functions and features typically desired in data federation. The overall results demonstrated that the DB2 Information Integrator provided more functionality, particularly in he area of higher degrees of optimization. The end result of optimization is that less

data may be transferred over the network and you will experience better performance and improved resource utilization. And, as mentioned in “The project systems environment” on page 4, we also demonstrated the data federation capabilities of an Informix database feature called I-STAR. I-STAR is a data federation capability that was developed to federate data from the Informix Dynamic Server and Extended Parallel Server data sources. It complements and extends the federation capabilities of DB2 Information Integrator and the Informix Enterprise Gateway Manager.

To enable client access to the test environment, we used the following query products:

- ▶ Brio Explorer
- ▶ DB2 Command Center
- ▶ Server Studio JE (Java Edition)

These products enabled easy development of queries required to test the data federation and to display the results. The graphical representation of the data schema made visualizing the testing being done very simple. In addition, it enabled us to view any modification to the SQL as a result of query optimization by DB2 Information Integrator or Informix Enterprise Gateway Manager.

Working with a federated information environment can significantly reduce your application development efforts and protect your development investment. This is because the applications can be written as if they are accessing one common data source rather than many. This saves the time typically required to understand the various data structures and their related programming requirements. This also has the impact of minimizing the on-going maintenance effort for those applications. There can also be significant savings in avoiding the acquiring and training of skilled resources that is typically required when there are many different data sources that a company, as well as application developers and data administrators, must use and manage.

If you are in the position of having many data sources that are used to support your organization, this publication is required reading. It will enable you to start down the path to a more standardized approach for application development. One that can provide you with reduced development and maintenance costs, lower requirement for skilled resources, and less maintenance burden.

After you have read the highlights and benefits you will have a better understanding of the value of data federation. We then encourage you to read the remainder of this publication to get more in-depth knowledge that will solidify your understanding of, and appreciation for, data federation.

Highlights and benefits

This section contains some of the highlights and benefits of data federation that we have demonstrated during our testing for this project. They are discussed throughout the publication, but we have summarized some of them here for easy and fast reading. The benefits of data federation are many and can be realized from more than one of the highlight areas.

- ▶ Reduces your need for skilled resources

Using a federated data approach enables you to access data from multiple disparate data sources as if they all were the same. Therefore, you should not need as many skilled resources to work with the different data sources you might have in your organization. This is particularly true when it comes to your application development staff. Your developers can now focus on developing applications rather than trying to understand all the nuances of the different data sources. It will also make it easier for you to find and employ skilled resources.

- ▶ Use standard SQL

You can use standard SQL, and SQL Expressions, in your applications. And, since it is DB2 SQL, you will always have a ready pool of available resources from which to fill your development resource requirements.

- ▶ Query optimization

You will get better performance and resource utilization because your queries will be optimized. This is unique to DB2 II. It means that more data operations will be pushed down to their lowest point of execution, which means less data will be sent across your network. The result is better performance and less resource utilization.

- ▶ Use of summary tables

Summary tables are often used to give faster query performance. Now there is also another type of summary table created as the result of a query. It is called a Materialized Query Table (MQT). But, if programmers are not aware of them they will use the original detailed tables. With DB2 II, the DB2 optimizer will be aware of these MQTs and automatically rewrite the query to use them—for a significant performance boost.

- ▶ Access nonrelational data

You can also access nonrelational data sources, such as Excel files, with the same relational SQL used to access relational data. You can even join the nonrelational data with relational data, fast and easy. This means writing no more extract programs to take data from nonrelational sources and create new, or temporary, relational sources so you can join the data together. This represents a tremendous savings in application development and data maintenance.

- ▶ Enables integration and consolidation

There are several ways to enable integration of your data resources. You can physically consolidate or logically integrate. Of course you can do some of both and ease the consolidation effort. Data federation gives you the option to integrate or consolidate on a planned and flexible schedule. This would apply to integration required as the result of a merger or acquisition as well.

As you read the remainder of this publication you will discover many more highlights and benefits that will help as you make those critical decisions on data integration and consolidation.



Data federation overview

In today's e-business on-demand environment, integrating information across and beyond the enterprise is a competitive mandate. Initiatives such as customer relationship management, supply chain management, and business intelligence are based on successfully integrating information from multiple data sources.

In this chapter we discuss the challenge of information integration. As with most technology directions, there are multiple levels of information integration and multiple approaches to consider when trying to accomplish it. One such approach is data federation. It is this approach on which we will primarily focus during the discussions.

1.1 The challenge of information integration

In most corporations, it is a given that different functional areas and departments will use different database systems, operating environments, techniques, applications, and software products to product, store, retrieve, and analyze critical data. It happens for many reasons, such as:

- ▶ Mergers and acquisitions
- ▶ Personal preferences
- ▶ Changing technology
- ▶ New product availability

As a result, corporations have their data stored in many different formats, on many different database systems, in many different applications, and in many different operating environments. This poses a huge challenge for the information technology (IT) department when data from these different areas must be consolidated, either temporarily or permanently. Also, this must be done if management is to get an accurate picture of what is going on in their businesses. It is a difficult position in which most corporations find themselves.

There are disparate sources of data everywhere, little of which is, or can easily be, integrated. For examples refer to Figure 1-1 on page 15. Most of the decisions made that resulted in this situation were well justified and probably good business decisions—at the time. But, now they are finding it extremely costly to consolidate, much less integrate, these sources of data to give management the information they need for decision making. So why do they not integrate all their sources of data?

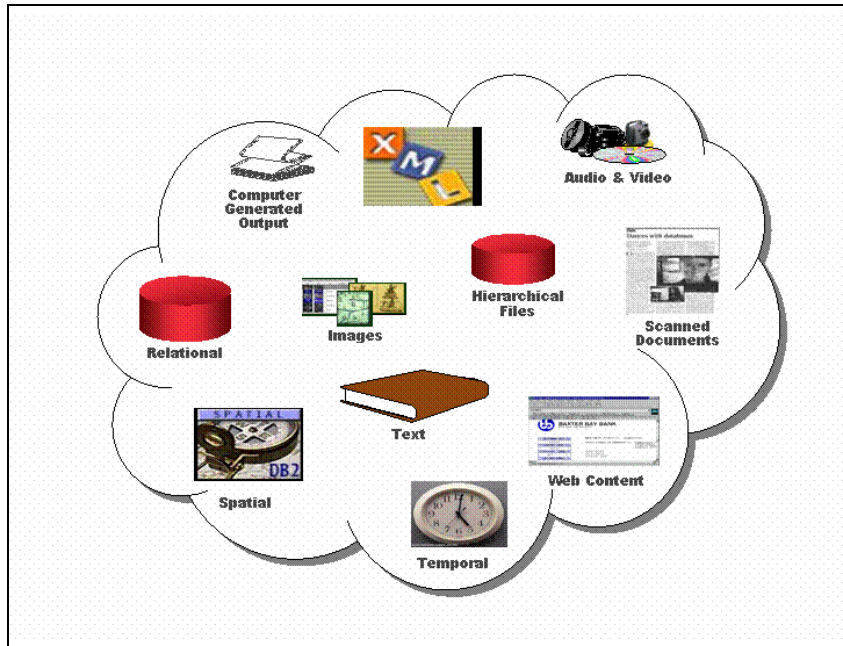


Figure 1-1 Data environment today

Information integration is a much discussed goal, but can be a very difficult goal to accomplish. When integration is attempted, it is most typically on a small scale and on an application-at-a-time basis. Therefore, it is not an architected enterprise data integration effort but rather integration to meet the requirements of a specific application or systems implementation. But, it is a small step forward.

Due to organizational, operational, or legacy data source constraints, any integration is typically a very expensive proposition—let alone enterprise information integration. In most organizations data is dispersed over a number of operating environments and data sources, both inside and outside the organization. Therefore it takes many different skilled resources to understand how to access, combine, and use the data from the many different types of data sources. Businesses continue to pursue that goal, but also look to other approaches to meet their requirements.

The vision of information integration is that users should be able to read and update all of the data they use as if it resided in a single source.

Information integration technology should shield the requester from all the complexities associated with accessing the data in its diverse locations and

formats and without regard to their accompanying semantics and access methods.

The challenge of information integration can be met using either of the two fundamental approaches below, or a combination of both:

- ▶ Data federation
- ▶ Data consolidation

We discuss these two approaches in more detail in the following sections.

Data federation

Data federation is not a new concept and has been around for some time. It is the ability to combine data from multiple heterogeneous data sources. It is a *logical* integration that typically takes place in real time. Although it is not full integration, many are finding that they can satisfy much of their data integration need with data federation.

Data federation refers to an architecture where a relational database management system enables access to heterogeneous data sources. This is depicted in Figure 1-2 on page 17. It enables access using the common language of the RDBMS, typically Structured Query Language (SQL). To do this some technology and mechanism must be used to translate the SQL into the language of the heterogeneous data source. Also, it must be able to perform such non-trivial tasks as handling the differences in data types that exist across these heterogeneous data sources.

Although data federation sounds relatively simple, there are many considerations and much care to be taken. To date, most data federation has been accomplished by custom programming. Now, however, there are products to help, such as the DB2 Information Integrator (DB2 II).

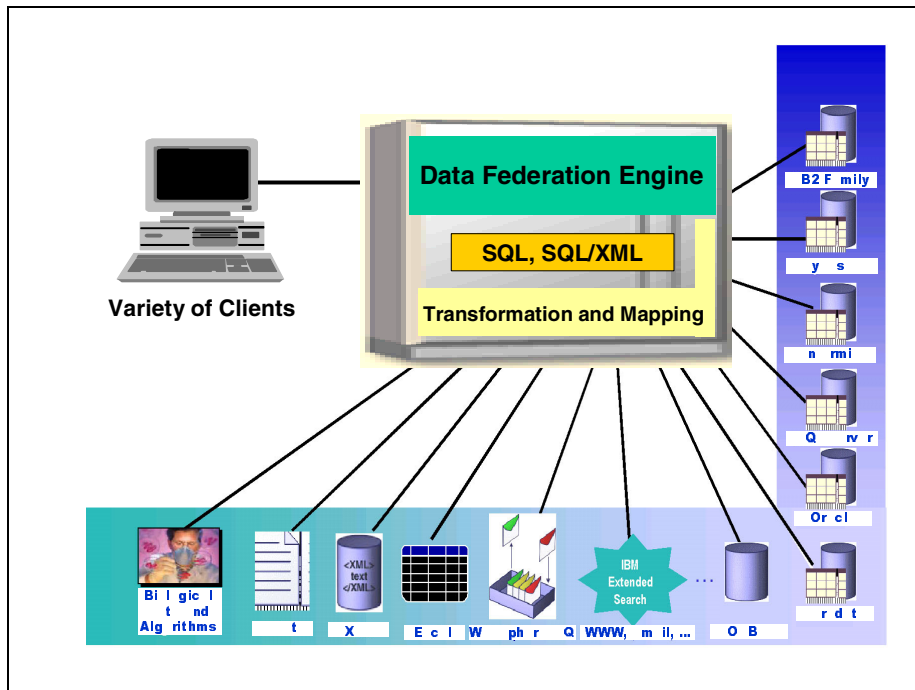


Figure 1-2 Data federation architecture

Some of the earlier federation attempts involved gateways, which allow a DBMS to route a query to another database system. Most accomplished this by using a standard access mechanism called Open Database Connectivity (ODBC). We will also demonstrate the use of a gateway in this publication. Informix has such a gateway, called the Informix Enterprise Gateway Manager (EGM), which we will use along with DB2 Information Integrator. Although EGM is a very robust gateway, you will find that DB2 II has a number of important capabilities that EGM does not—for satisfying the requirements of data federation.

The following are a number of capabilities that should be inherent in a product supporting data federation. A few are included in the following list:

- ▶ Conceal differences in the implementation and use of a heterogeneous data source. It must behave as if it were part of the single database environment.
- ▶ Support for many heterogeneous data sources.
- ▶ Ability to exploit unique capabilities of the heterogeneous data source.
- ▶ Ability to easily add support for new data sources. In addition, the federating DBMS should be able to extend its capabilities to operate on the

heterogeneous data source. This could include using the DBMS optimizer, for example.

For more detailed information on the topic of data federation, there is an excellent article in the IBM Systems Journal Volume 41, November 4, 2002. It is called "Data Integration Through Database Federation," by L.M. Haas, E.T. Lin, and M.A. Roth. Some of the above information was distilled from that article.

Gartner Group estimates that 70 percent of corporate application development budgets are allocated to accessing and federating disparate data.

Data federation (or distributed access) achieves the vision of information integration by giving the *appearance* that the various federated data sources exist in the same place. The creation of this illusion is the role of the *federated database server*. This is the only server that the end user or application will access directly, and it is in this way that the benefits of the single data source illusion are experienced.

Data consolidation

Data consolidation (or data placement) physically brings the source data together from a variety of locations into one place in advance, so that a user query does not need to be distributed. This approach typically uses either extract, transformation and load (ETL), or replication functionality. As with the federated approach above, the end user or application interacts only with this one physical *consolidated database server* to enjoy the single data source experience.

With data consolidation, both the remote data request and transfer must occur before the end user or application request is issued. It is logical therefore that the request to the remote data source is basically formulated one time only during data requirements definition, while the transfer of data typically occurs many times according to some defined cycle or trigger. Neither the data request nor the data transfer to and from the remote data source are directly related to the end user's request. Although, hopefully, there is some relationship with the end user request and the data architect's prediction of the types of queries to be serviced.

Data consolidation or placement is the traditional approach to integrating information and, in contrast to data federation, moves the data to the query. It has always been considered less complex than data federation, as data consolidation creates a second, local copy of the data, pre-processed as required, thus reducing the need for extensive data manipulation and remote access within the user query. Data consolidation, because it operates off the critical time path of the user's query, also allows for substantial and complex transformation of the data to address issues of cleanliness, semantic and temporal consistency, and so on. It therefore exhibits varying levels of complexity. At its simplest, it is a

manually initiated database unload followed by a load of the target system. At its most complex, it may involve the automated, real-time, multi-way synchronization of databases on a number of remote systems. In most cases today, it is somewhere in between.

A key consideration for data consolidation is the maximum latency that can be tolerated when transferring the data from source to target. Typically, business needs specify how up-to-date a copy of the data must be. In data warehouses, for example, the frequency required might be daily or weekly and the latency of data consolidation can easily extend to many hours. At the other extreme, the need for almost real-time data, such as in stock market systems, requires minimum latency in data consolidation.

Two of the most important factors determining the minimum latency possible in data consolidation are the complexity of the transformations required and the volumes of data to be transferred. These factors lead to two complementary approaches to consolidating data. ETL is optimized for larger data volumes and is often associated with more complex transformations, while data replication emphasizes the transfer of individual data records and is often restricted to simpler transformations.

1.2 To federate or not to federate

Data federation and data consolidation are actually similar concepts. Both involve requesting and receiving data that originally resides outside the physical confines of the database server with which the end users or applications interact. The key difference is in the timing of the data requests to, and transfers from, the remote data source and the central sever. With data federation, both the remote data request and transfer occur after the end user or application has issued the request to the federated database server.

But from the end users' points-of-view, or that of the applications acting on their behalf, data federation and data consolidation act in opposite ways. Data federation integrates the required information synchronously, directly from its original sources, acting only after the end user decides what information is required. It must therefore return the result in a time frame that is acceptable to the user or requesting application. Data consolidation operates in advance of the user query, allowing itself more time to perform the required processing. However, the data architect needs to make decisions in advance regarding what data will be required. Secondly, because data consolidation is creating a second copy of the data, it requires a larger quantity of permanent data storage than the data federation approach.

It is sometimes suggested that data federation is a technique that can be used to provide users direct, unplanned access to any data, anywhere, because a federated query allows data to be combined in real time. Although this could be a theoretically true statement, it is a potentially dangerous suggestion. Such unrestrained access could lead to significant performance problems that could impact the end user, the federated database server, and, perhaps more importantly, the source systems. Furthermore, data federation *can* actually demand even more rigorous analysis, modelling, control, and planning than data consolidation. This would happen, for example, to avoid significant performance and semantic problems for users if they try to combine data that is inconsistent in meaning and structure.

Consolidated data stores, such as data warehouses and operational data stores constructed through data placement, have the capability to avoid such issues using a range of techniques. For example, they could do so by using a consistent, point-in-time, snapshot approach. The data consolidation approach can still have advantages in some areas when compared to a federated database. However, data federation may eliminate the need for building a data mart. If the volume of queries is not large, and often can be satisfied with summary tables, a huge productivity boost can be realized by eliminating such things as the need for a data mart and the corresponding creation of a new server, and movement of significant quantities of data to populate it. Of course, for frequent complex queries that need access to the lowest level of detailed data, a data warehouse is the preferred solution.

Most queries require tables to be joined together, which can be resource intensive and time consuming. So, how can you speed up queries that require a join operation? Well, the fastest way to join two tables is not to join them at all. The Transaction Processing Council (TPC) TPC-D benchmark, now withdrawn, proved conclusively that if the answer to a query is pre-computed in a summary table, the query runs faster.

How can this summary table speed-up technique be applied to a federated environment? Because to really take advantage of summary tables they have to be used whenever possible. How can we make that happen? Let us give an example. A summary of data at a monthly interval can satisfy a query at a monthly, quarterly, or annual level of aggregation. If the query optimizer is smart enough to rewrite the query to operate on the summary table instead of seeking the detail, then we will get an advantage.

We tested this capability by adding a summary table, and it works. In DB2, the summary table is called a Materialized Query Table (MQT). A powerful feature of DB2 II is that it can recognize and understand the MQT. In our test, it recognized that the MQT could satisfy the query we submitted. So DB2 II intercepted the SQL for the query and rewrote it to access the MQT rather than the base tables.

This meant that there were no joins to perform, just a simple access to the MQT. That resulted in a significant reduction in processing resources and response time.

Arguments in favor of data federation

Data federation is likely to be the appropriate method of data integration when:

- ▶ Real-time or near real-time access to rapidly changing data is required.
Making copies of rapidly changing data can be costly, and there will always be some latency in the process. Through data federation, the original data is accessed directly and joined in the query. However, the performance, security, availability, and privacy aspects of accessing the original data must be considered.
- ▶ Direct immediate write access to the original data is required.
Working on a data copy is generally not advisable when there is a need to insert or update data, because data integrity issues between the original data and the copy may occur. Even if a two-way data consolidation tool is available, complex two-phase commit locking schemes are required. However, writing directly to the database of another application is generally prohibited. Therefore, it is generally recommended to call the owning application, via an application program interface (API) invoked by the federated database server, to request any updates.
- ▶ It is technically difficult to use copies of the source data.
When users require access to widely heterogeneous data and content, it may be difficult to bring all the structured and unstructured data together in a single local copy. Or, when the source data has a very specialized structure, or has dependencies on other data sources, it may not be possible to sensibly query a local copy of the data. In such cases, accessing the original source is recommended.
- ▶ The cost of copying the data exceeds that of accessing it remotely.
The performance impacts and network costs associated with querying remote data sets must be compared with the network, storage, and maintenance costs of storing multiple copies of data. In some cases, there will be a clear preference for a federation-based approach, such as when:
 - Data volumes of the original sources are too large to justify copying it.
 - Data is too seldom used to justify copying it.
 - A very small or unpredictable percentage of the data is ever used.
 - Data is accessed from many remote and distributed locations, which would imply multiple copies.

- ▶ It is illegal or forbidden to make copies of the source data.
 Creating a local copy of source data that is controlled by another organization or that resides on the Internet may be impractical, due to security, privacy, or licensing restrictions. However, federated access with discrete queries may be permitted. If access to the sensitive data is only of an *on-demand* nature, this access can be audited (if applicable) at the remote source using the normal auditing techniques applied at that data source. This auditing capability at the remote data source *to know who has accessed what* would be lost under the data consolidation approach.
- ▶ The users' needs are not known in advance.
 Allowing users immediate, ad hoc access to any enterprise data is an obvious argument in favor of data federation. However, care is needed with approach. The potential exists for users to create queries, accidentally or otherwise, that negatively impact both source systems and network performance, and that disappoint the user by yielding poor response times. In addition, because of the typical level of semantic inconsistencies across data stores within organizations (as a result of differing data latencies, formats, and data structures), there is a significant risk that such queries could return answers that are of little or no value.

Arguments against data federation

Not surprisingly, the arguments against the data federation approach are the very same as those in favor of data consolidation:

- ▶ Read-only access to reasonably stable data is required.
 The data federation approach will present the remote data in real time. This may not be advantageous to the end user or application, which would prefer to suffer some latency in the data in order to be insulated from the continuous flux of information in the remote operational data sources.
- ▶ Users need historical or trending data.
 Historical and trending data is seldom available in operational data sources, but requires a data consolidation approach to build up historical data over time. This is a very common data warehousing requirement.
- ▶ Data access performance or availability is overriding considerations.
 Users routinely want quick data access. Despite the best efforts of a well-designed federated server working in unison with the remote data sources, the volumes of data required may necessitate that a local, pre-processed copy of the data be made available. As seen in data warehousing environments, the queries to be serviced can be very complex or may require a multi-dimensional view of historical or trending data. As a result, data consolidation is a fundamental technique in data warehousing. However, data federation can still be used in conjunction with data

consolidation to provide support for the lowest drill-down level. In this case the data warehouse acts as both the consolidated database server and the federated database server.

- ▶ User needs are repeatable and can be predicted in advance.

When user queries are well-defined, repeated, and require access to only a known subset of the source data, it may be cheaper to create a copy of the data for local access and use. This approach also insulates the remote operational data sources from large, complex, or poorly structured queries.

- ▶ Data transformations or joins needed are complex or long-running.

In cases where significant data transformations are required or where joins are complex or long-running, it is inadvisable to have them run synchronously as part of a user query due to potentially poor performance and high costs. In such cases, creating a copy of the data through data consolidation would seem to be more advantageous.

Using both data federation and data consolidation

It is likely that there will be cases where a combination of data federation and data consolidation techniques is the best option.

One case is where a federated query can leverage data consolidation functionality transparently. This is because sometimes a federated query will not work. It could be because of network outages, for example. Here, for example, data federation can use data consolidation to create or manage cached data. On the other hand, data consolidation tools may be optimized of only a subset of available data sources. Using data federation along with it can expand that number, and allow pre-joining of data for a performance impact.

Later on in Chapter 7, “Optimization in a federated environment” on page 193, we consider exactly such a combination of techniques through the use of Materialized Query Tables (MQTs). MQTs are data stores of preprocessed data from the remote data sources stored at the local federated server. We consider a user query against the federated database where the underlying tables reside in one or more remote databases. We demonstrate how we can exploit the federated server to transform the query to instead reference the MQT residing locally at the federated server.

1.3 Data federation examples

The previous sections have discussed the pros and cons of data federation, and described it in general terms. In this section we present examples of data federation in environments where DB2 is the base, and where Informix is the base. These are simply to provide you with an idea of the capabilities for data

federation. In fact, either DB2 or Informix can be the base, or you can use a combination of both. Specific details on how to implement data federation, capabilities, and considerations are discussed in later chapters.

In the following sections, there are figures that depict the general environments of DB2 and Informix, when used as the base for a data federation. There is a wide range of support with both products that can provide significant benefits as you integrate the information in your enterprise. And, as we have discussed, there are differing ways to implement an integrated environment. The primary message here is that whether you have DB2, or Informix, or both, you have powerful tools and capabilities at your disposal that can save you time, effort, and resources as you integrate the information in your enterprise.

Data federation has become a necessity with the evolution and availability of numerous data management products. Between new technology advances and the normal evolution of business requirements, companies find themselves with quite a number of different, and incompatible, data sources. Most companies are using some form of data federation, even if calling it by another name. Also, it is typically a customized approach that is very expensive and resource intensive to support and maintain.

Even with the advent of relational technology and databases, companies find themselves with more than one relational DBMS. This could be for many reasons, but it became a reality and the norm rather than the exception. To minimize the skilled resources and extra work required in application development, as well as to satisfy the continued push for data integration, companies demanded additional capabilities from their vendors. In the beginning, that support was primarily for relational technology.

Vendors responded with architectures and products to give companies some relief. As examples, IBM introduced data replication technology for data movement, and Distributed Relational Database Architecture™ (DRDA®) to support standard interfaces for data exchange. A first step was to provide easy data access and interchange among IBM relational databases.

Data replication became a fast and reliable way to have multiple images of data in multiple locations, all kept in synchronization. The technology quickly moved beyond relational, for example, with support for the IBM Information Management System (IMS™) database technology.

Third-party products and tools became available to support inter-vendor data access and exchange, and the move to integration and data federation was on. The technology is still growing and becoming more sophisticated. The goal is simple: Access to any data source, anywhere, with the same programming language or tool.

1.3.1 Data federation with DB2

We show in Figure 1-3 an example of the data federation capabilities when using DB2 as the base. You can see that there are a number of tools to enable you to implement a robust federated data environment. There is access to relational and nonrelational databases, file systems, tools, and Web data sources. In most cases, the DB2 approach uses the native API for each data source for compatibility, breadth of support, and performance reasons.

From a programmer perspective, all data sources will appear to be the same. That is, they will all appear to be DB2 sources. Also, there is access to both structured and unstructured data (universal data) through the use of DB2 Extenders. Extenders are extensions to DB2 that enable additional capabilities. The primary product to be used for data federation is the DB2 Information Integrator, and it is discussed in great detail in subsequent chapters.

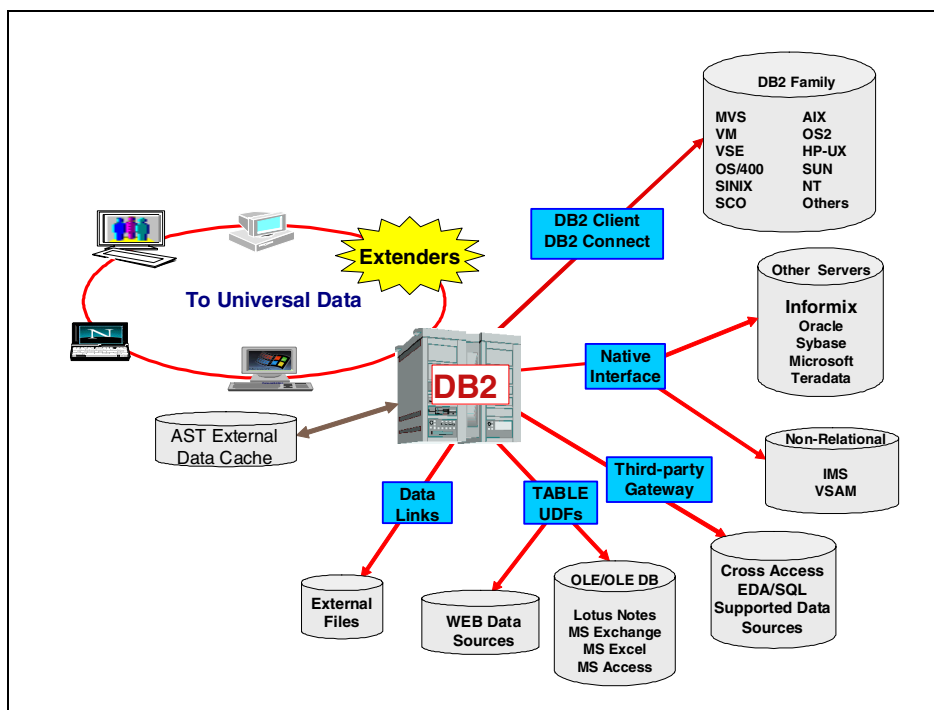


Figure 1-3 DB2 data federation

One of the primary advantages of DB2 Information Integrator is that it takes advantage of the DB2 optimizer. The optimizer examines the environment to determine the best way to access a data source. For example, it looks at the processors involved, network speeds, and statistics regarding the type, size, and

characteristics of the data source. It then develops the best access plan to be used for the data sources involved. This may include performing some of the work on the local processor and some on remote processors, whereas many older technologies simply bring all the data back to the local processor and work on it there. This is rarely an optimal approach. DB2 Information Integrator can then rewrite the SQL access commands to retrieve the data with the approach that results in the lowest resource cost. That is why the DB2 optimizer is categorized as a *cost-based* optimizer.

We present a number of examples, in later chapters, that result in the development of new query plans and the rewrite of SQL by the DB2 optimizer. These examples will clearly demonstrate the advantages gained and benefits received by this capability.

1.3.2 Data federation with Informix

Informix products can also be used as the base for data federation. They can provide a number of capabilities that are similar to those described in the DB2 data federation example. We depict a data federation environment, using Informix as the base, in Figure 1-4 on page 27. The primary product used for data federation in that environment is Informix Enterprise Gateway Manager. We briefly describe that approach here, but it will be discussed in greater detail in subsequent chapters.

To address data federation, a number of vendors developed what was referred to as *gateway* products. That is, they provided a gateway to other heterogeneous sources of data. For example, Informix introduced their Enterprise Gateway. The first version of this gateway used IBM DRDA to enable access to data on the IBM mainframe processors. At that time, the mainframe was where vast amounts of data was stored, so to enable mid-range solutions, vendors needed to give their customers access to that mainframe source of data.

Support was then expanded to enable access to other vendor data sources, regardless of the platform, but access was primarily through the use of ODBC rather than the native DBMS API. This approach does enable data access, but can bring with it performance issues. This is why DB2 has focused on using native APIs, and it is one of the differentiators of the DB2 approach.

Over time, gateways improved and began to address some of these issues. For example, as we discuss in later chapters, the Informix Enterprise Gateway Manager has the capability to change the data access plan as well. However, there are also some limitations.

In the Informix example, you will note that it can also enable access to both structured and unstructured data. Informix *DataBlades* are used for the interface

to unstructured data. These are extensions to the database that enable this capability, and a number of others.

Early on, Informix introduced I-STAR as a way to enable easy access and exchange of data between the different Informix relational databases. This is one of the ways Informix provided the capability to change access plans and distribute some of the query execution tasks. It was another step towards data federation. I-STAR began as a separate product, but over time was integrated into the Informix base technology.

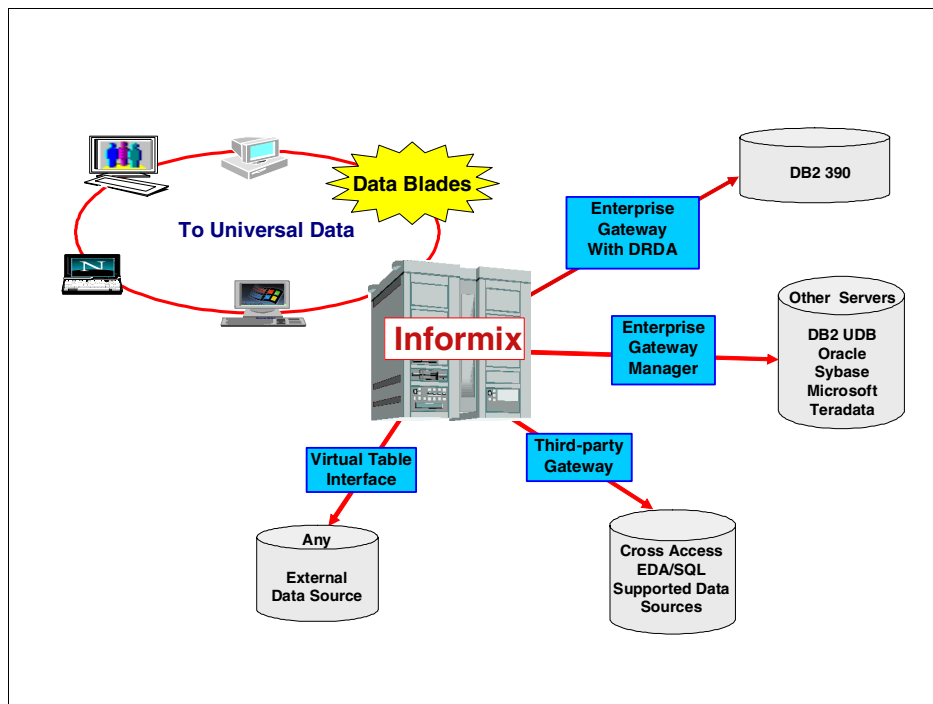


Figure 1-4 Informix data federation

Informix I-STAR Technology

In the early days of integration and data federation, Informix developed a technology called *I-STAR*. It was a separate product at that time, but has since been integrated into the Informix code base for Informix Dynamic Server and Informix Extended Parallel Server.

I-STAR allows users to query and update more than one database across multiple database servers, within a single transaction. The database servers can reside on a single host computer or on the same network. A two-phase commit protocol ensures that transactions are uniformly committed, or rolled back,

across multiple database servers. You can also use Informix database servers with Informix Enterprise Gateway products to manipulate data in non-Informix databases. The heterogeneous commit protocol ensures that updates to one or more Informix databases and one non-Informix database in a single transaction are uniformly committed or rolled back. You can also use I-STAR in a heterogeneous environment that conforms to X/Open.

There is more detailed information on the capabilities of both DB2 Information Integrator and Informix Enterprise Gateway Manager in the chapters that follow.

1.4 Considerations

In this section we discuss some individual techniques to be considered with data federation. In later chapters, we make specific recommendations based on the test environment we used.

1.4.1 Naming conventions

If you think naming conventions are not too important, we would like you to review that position now. In this project we found ourselves managing more discrete, but similar, data sources than ever before. If you are clear about which data source you are looking at, you will be able to federate your different data sources much more easily.

Good naming conventions are invaluable when interpreting query plans to analyze any potential performance problems you may encounter. As you will see when you read 2.1, “Environment and server configuration” on page 33, our database schema was not overly complex. However, we did purposely create quite a lot of complexity in the number and types of database servers. This was to more closely simulate an environment that might be more closely comparable with that in which you work.

In our environment, it is the multiplication factor provided by having the same schema distributed over a large number of database servers, that really demonstrates that naming conventions are critical.

In this context we are referring to the local names you allocate in the federated database server to the remote data sources. Of course the existing remote data sources will already have their own names and we are not suggesting that you embark on a campaign to rename all of those.

In our environment, we used a naming convention, the main purpose of which is to aid clarity for you. Consequently, once you understand the convention, you can easily determine the remote data source for a given federated database object.

Bear in mind that the names *you* allocate in the federated database server for the remote data sources are the names by which all *your* end users and applications will address these objects in the *single data source illusion* that the federated database server creates. Of course, you may choose to obscure the data source for the federated database objects from your end users and applications by applying aliases, synonyms, or views over the top of your other names.

You may, or may not, choose to follow something similar to our naming approach. After reading this publication, and hopefully getting a good understanding of what data federation is all about, we do at least request that you give some thought to naming conventions. It is much more cost effective to do this at the beginning of this data federation journey.

1.4.2 Data types

In general, the data types in your remote data sources will already be determined by the time you decide to federate your data, so any difficulties you face in this area will be the legacy of other system design decisions taken long before you arrive at this point.

In addition, certain higher value data types are either not supported or are supported with some limitations with the currently available data federation technology. Examples are certain binary large objects (BLOBs), character large objects (CLOBs), and some date/time data types on some platforms.

Fortunately, there are some weapons in the data federation arsenal to help you here. We discuss some of the problems we encountered and how we dealt with them for DB2 II in 4.4, “Considerations for use” on page 108, and for EGM in 5.3, “Considerations for use” on page 149.



The data federation project

In this chapter we introduce you to the environment and architecture used for our product implementations and testing. We used this environment to exercise the capabilities of the products, to validate specific functional scenarios, and to evaluate and document the results for inclusion in this publication.

In setting up the environment, our objectives were to:

- ▶ Create an environment that was complex enough to be a reasonable model for a real-life commercial data federation scenario.
- ▶ Encompass the primary database management systems (DBMSs) typically in use in Informix installations. We recognize that most Informix installations will perhaps not have all the DBMSs we used, but most will probably have enough of a subset to gain value from the information presented.
- ▶ Have an environment that would constitute a reasonable compromise between meeting the above objectives and still be manageable within our constraints of server availability and time.

We also introduce you to our case study, which we hope will give you something near a *touch and feel* experience of how you could use data federation within your own organization.

We have constructed the case study scenario centered around a fictitious company that we have based in the United States. This fictitious company has the states organized into seven regions. Each region in our case study uses the

same database schema, but has implemented it using a different database management system or operating system. Most businesses would probably be less complex than this. However, we wanted to demonstrate the robustness of the data federation capabilities that are available to you. We have used the Stores Demo database schema, which is a sample database supplied with Informix Dynamic Server (IDS) and Informix Extended Parallel Server (XPS). Most companies that have Informix installed are already familiar with this schema, which should aid the understanding of our case study.

As the starting point for the case study, we populated the databases in each region with significant quantities of meaningful test data. We then integrated the disparate databases from the regions into a single federated corporate database. To simulate daily operations, we constructed sample queries to run against our federated database. Sometimes we ran into problems with incompatible data types in the different DBMSs and had to overcome these issues. We analyzed access plans produced by the query optimizer for execution of our sample queries, and then investigated the effect on performance of altering various configuration settings.

In constructing this case study, we discovered many considerations for building a federated system. You can read about all of this in detail in Chapter 4, Chapter 5, and Chapter 6.

2.1 Environment and server configuration

Our operating environment consisted of a workstation client and three servers, shown as A, B, and C in Figure 2-1.

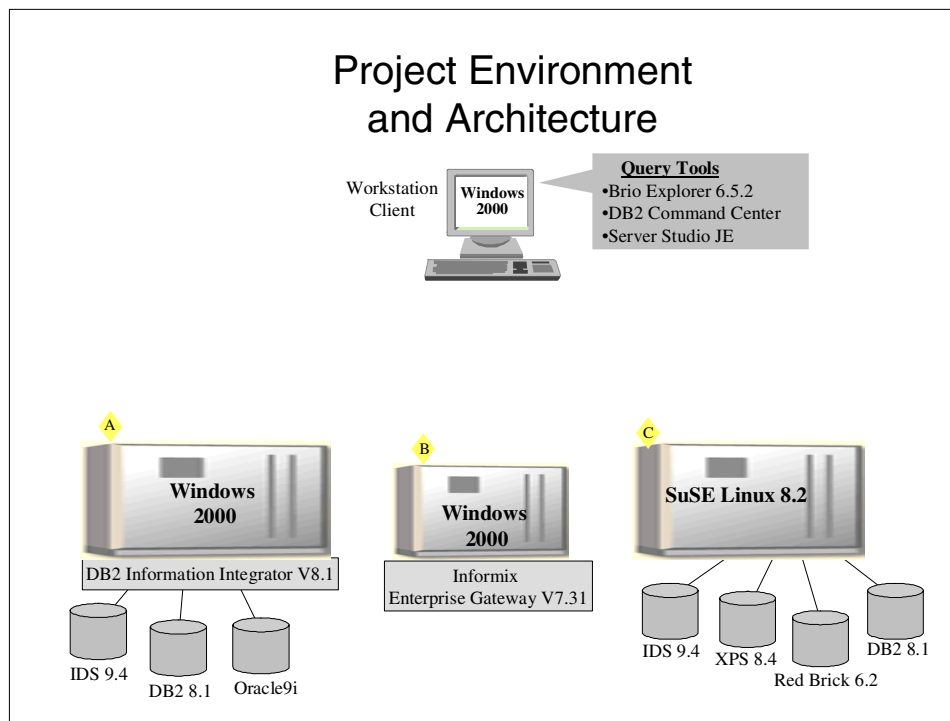


Figure 2-1 Project environment and architecture overview

The definitions of the environment components are as follows:

- ▶ Server A is an IBM NetVista™ Intel server running Windows/2000 Server SP3. This server hosted the DB2 instance in which we installed DB2 Information Integrator (DB2 II) V8.1 with FixPak 3. The name given to this server was winsvr.
- ▶ Server B is an IBM PC 300PL Intel desktop running Windows/2000 Professional SP3. On this server we installed Informix Enterprise Gateway Manager V7.31.TD1. In fact, we defined multiple gateway services for the Informix Enterprise Gateway Manager on server B. Multiple gateway services are required to reach the multiple different remote data sources. See “Multiple data sources within a single application” on page 151 for more details. The name given to this server was gtwsvr.

- ▶ Server C was an IBM NetVista Intel server running SuSE Linux V8.2 (Kernel V2.4.19). This server contained the Informix Dynamic Server (IDS) database instance, which was used as the federated server for the Informix federation solution. The name given to this server was linsvr.

Within server A (winsvr), we installed the following database management systems:

- ▶ Informix Dynamic Server V9.40.TC1
- ▶ DB2 UDB V8.1 FixPak 3
- ▶ Oracle9i V9.2.0.1.0

Within server C (linsvr), we installed the following database management systems:

- ▶ Informix Dynamic Server V9.40.UC1
- ▶ Informix Extended Parallel Server V8.40.UC1
- ▶ Red Brick Warehouse V6.20.TC4
- ▶ DB2 UDB V8.1(FixPak 0)

Each DBMS where we have stored data is referred to as a *data source*. When accessing a data source from another server or a different DBMS on the same server, the data source is known as a *remote data source*.

Having done that, we then proceeded to federate our complete distributed database twice using different methods. In the first case, we used DB2 Information Integrator, and in the second case we used Informix Dynamic Server in conjunction with Informix Enterprise Gateway Manager. We only federated our database twice for informational purposes so we could test them and document the results. However, even though you could also use both, typically you would federate a database using only one of these federation technologies.

2.1.1 Client configuration

We used Brio Explorer V6.5.2 for our client workstation, to investigate a typical end user tool view of our federated database.

In addition, when running the sample queries against the DB2 II, we used the DB2 Command Center, which is shipped with DB2 UDB. When issuing the same sample queries against the Informix federated database, we used Server Studio JE 3.00.JC1, which is shipped with Informix Dynamic Server. Both these GUI SQL clients operate in similar ways and can be used to produce a query plan. For more details, see Chapter 6. In our testing with the case study we defined, our sample queries accessed every data source, and because we had used consistent naming conventions, we were able to run basically the same queries using each of the two federated databases.

While Figure 2-1 shows the overall project environment and server architecture, this diagram does not attempt to illustrate the actual connectivity between the client workstation and the various servers. This is described as a *build process* in the following sections to aid in clarity.

2.1.2 Database connectivity using DB2 Information Integrator

This section describes the connectivity of the federated database with the other data sources in the project when using DB2 Information Integrator. Figure 2-2 shows the client view of the federated database and the actual connectivity with the DB2 II federation solution.

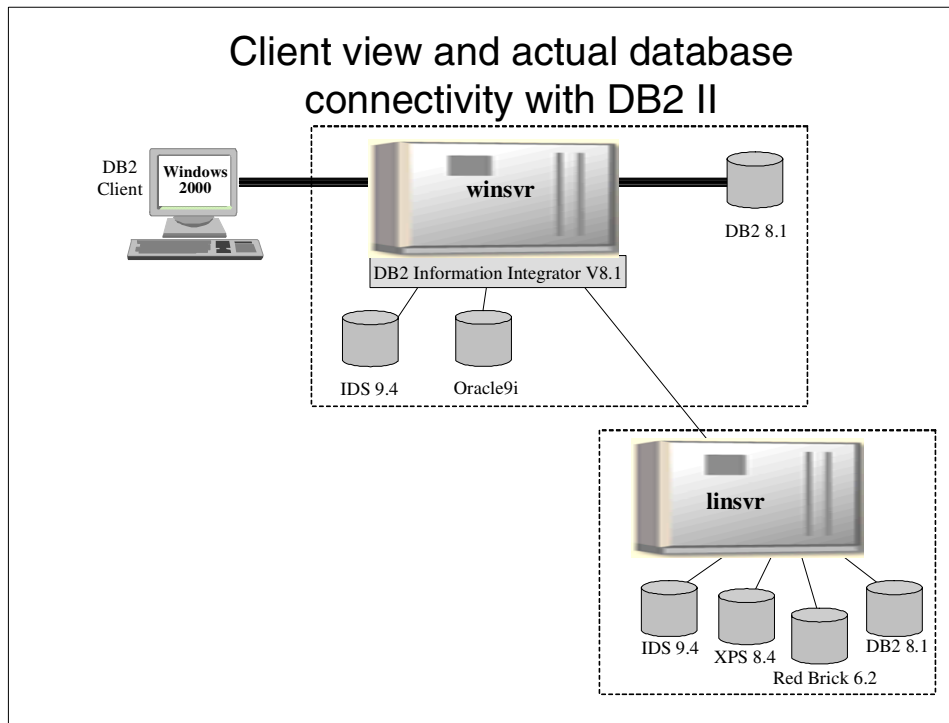


Figure 2-2 Client view of federated database and actual connectivity with DB2 II

The bold lines in Figure 2-2 show the client's perception of the federated database using DB2 II. From the client's perspective, there is only the one DB2 database instance at winsvr with which to connect and authenticate, and from which to request data. To the client, all the tables in each of the data sources appear as DB2 tables within the one database at the server winsvr. DB2 II has created this illusion over the *real* database connectivity diagram shown with the narrower lines in Figure 2-2

2.1.3 Database connectivity using Informix EGM

This section describes how we connected the federated database to the other data sources in the project when using Informix Enterprise Gateway Manager. Figure 2-3 shows the client view of the federated database and the actual connectivity with the Informix federation solution.

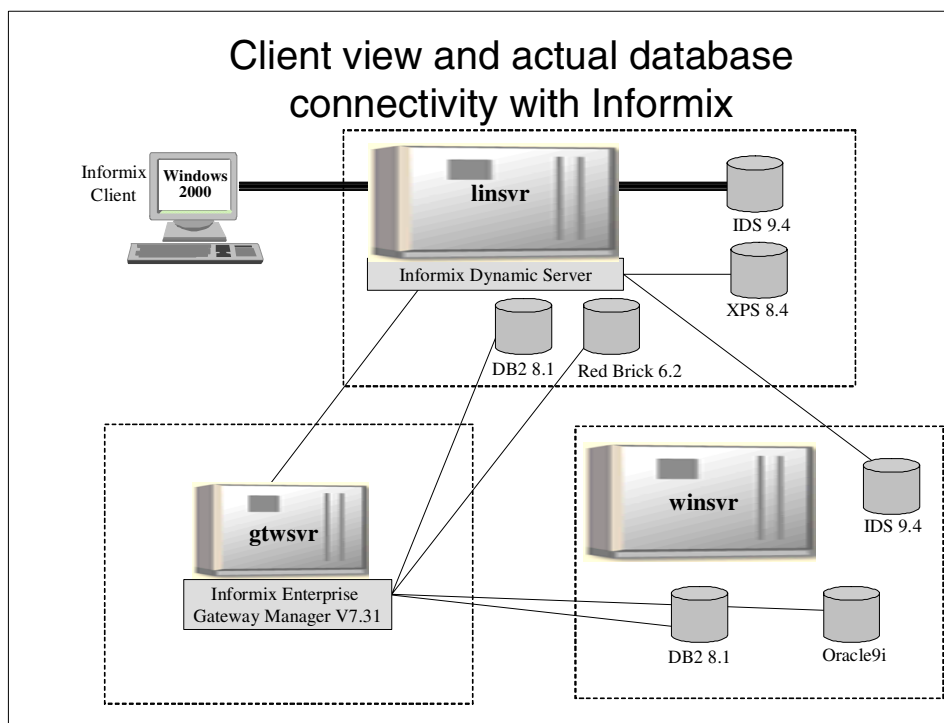


Figure 2-3 Client view of federated database and actual connectivity with Informix Database Server and Informix Enterprise Gateway Manager

The bold lines in Figure 2-3 show the client's perception of the federated database using Informix. As with DB2 II, from the client's perspective, there is only the one Informix database instance at server linsvr with which to connect and authenticate, and from which to request data. To the client, all the tables in each of the data sources appear as Informix tables within the IDS database at the server linsvr. The real database connectivity is shown with the narrower lines in Figure 2-3.

2.1.4 Other configuration options

The architecture described above in 2.1, "Environment and server configuration" on page 33, was partially mandated by the technical requirements and scope of

the products we were using. It was also partially chosen on the basis of what we considered to be reasonable. Let us look at what other options exist and what the real technical constraints are.

DB2 Information Integrator

DB2 Information Integrator (DB2 II) *must* be installed into an existing DB2 UDB instance. If you have a DB2 UDB license for the candidate server, or if you purchase DB2 II Advanced Edition, you may also store other DB2 application data in the same DB2 instance or in another DB2 instance at the same server.

We could have chosen to install DB2 II at any of the three servers in our operating environment. If we had chosen server B, we would have had to also install DB2 UDB on server B for this purpose.

For performance reasons, it may be preferable not to run DB2 II on the same server as other mission-critical DB2 operational systems. Bear in mind that some of the queries to be serviced in a federated environment could be extremely resource intensive.

Informix Enterprise Gateway Manager (EGM)

The Informix EGM could have been defined at either server B or C. It could not be installed at server A, as the EGM product is not currently available for the Linux operating system. Currently available platforms are Windows and Unix (AIX®, HP- UX, and Sun Solaris). We chose to locate the EGM at a server where no other Informix database instances exists, but this was simply by choice.

If you do decide to install EGM at the same server as another Informix instance, the EGM will be viewed by the Informix federated database server as another Informix instance at the local server.

In either architecture, a client cannot connect directly to the EGM. A client must always connect to an Informix database server, which may or may not be acting as a federated database server.

Informix federated server

We chose the IDS instance on server C (linsvr) as the *federated database server* for the Informix data federation solution. For the Informix data federation solution, we must connect our end-user query or application to an Informix Dynamic Server database. This database instance acts as the federated database server or *coordinating server*. We could equally well have chosen the other IDS instance on server C.

Note: XPS in its current version *cannot* be used as a federated database server.

We wanted to facilitate convenient access to the other remote data sources (both Informix and non-Informix) and to provide the highest possible level of portability between DB2 II and the Informix data federation solution for our sample queries. To achieve this, we defined local references at the IDS instance on server A, which points to the remote data sources. These local references can be created using views and/or synonyms. For details, see 5.3, “Considerations for use” on page 149.

Had we chosen to use the other Informix Dynamic Server instance on server A (winsvr) as our federated database server, we would have instead defined similar local references at that Informix instance.

2.2 Schema definition

To build a case study that appears realistic, we adopted the Stores Demo database schema, which is a sample database supplied with Informix Dynamic Server (IDS) and Informix Extended Parallel Server (XPS). A graphical representation of the Stores Demo schema is shown in Figure 2-4.

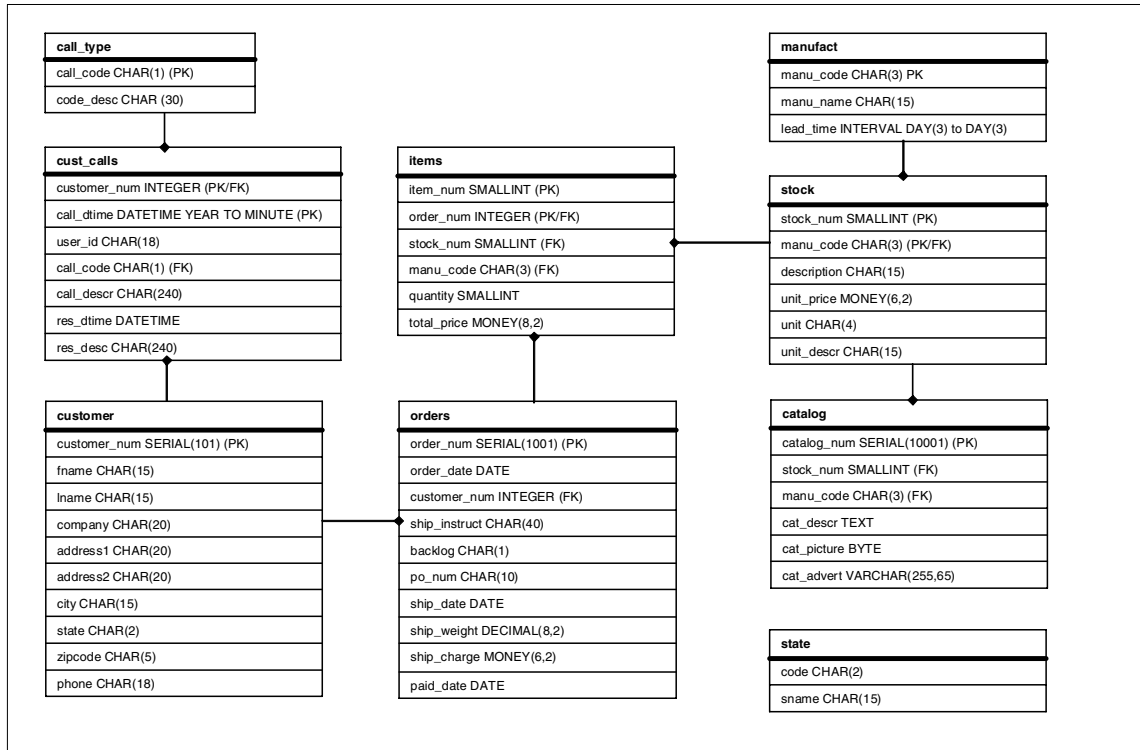


Figure 2-4 Stores Demo database schema for Informix databases

We then implemented this database schema into each of the DBMSs present in our operating environment. Some of the data types in this schema that are unique to Informix required alteration to facilitate implementation in the other non-Informix DBMSs, DB2, Red Brick, and Oracle. Two such data type examples are the MONEY and INTERVAL data types. This is illustrated for DB2 in Figure 2-5. Note that the Stores Demo database schema for DB2 databases has modifications made to a several column definitions (changed columns are shown underlined in bold italics).

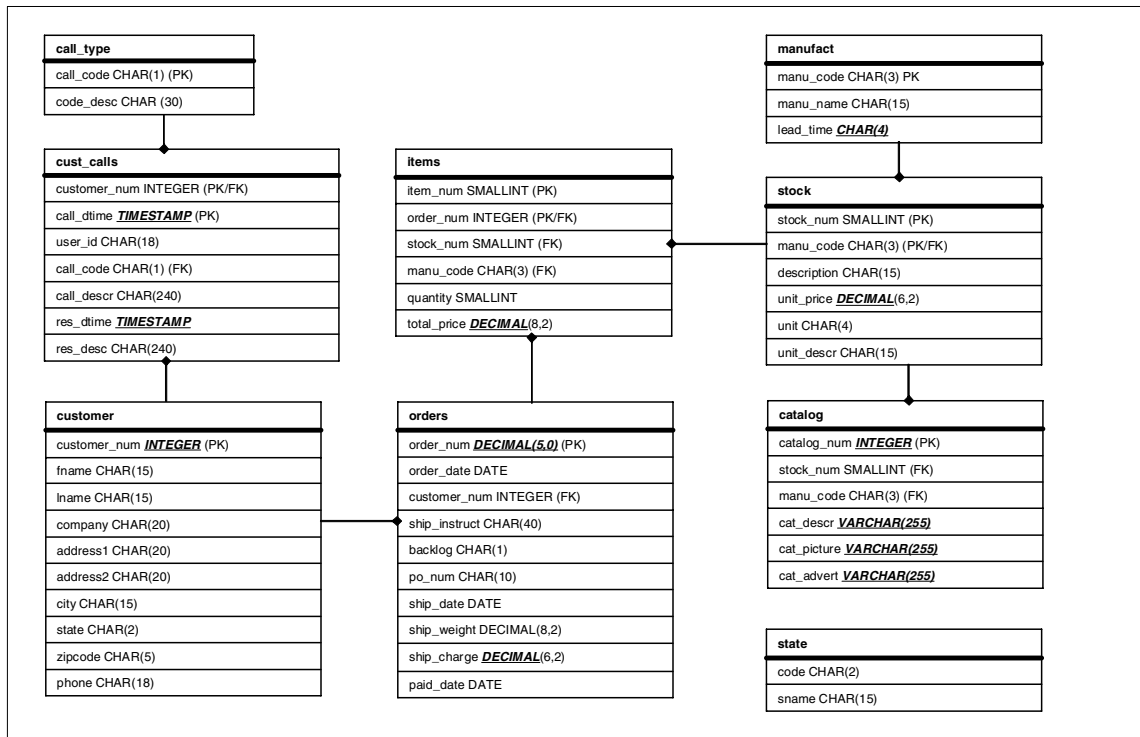


Figure 2-5 Stores Demo database schema for DB2 databases

Having implemented the Stores Demo schema in each of the DBMSs, we needed to populate the database tables. To do this we created a model for the fictitious corporation in our case study. We allocated each of the individual states in the United States to a region, where each region was using a different DBMS and/or operating system. Perhaps such a scenario could arise where the same custom application had been implemented regionally within the one corporation on the local hardware and/or operating system present in that region. Another possibility for this scenario might be where an application package had been implemented in different companies on different hardware and/or operating systems that had now been formed into one corporation as a result of merger or acquisition activity.

Another variation on this theme might be a situation where the schemas in the remote data sources are actually substantially different and some thorough data analysis is required before attempting to get meaningful results through the federation technology. But in fact this situation is no different from trying to join tables from different schemas in the same database. This is a data analysis problem, not something unique to data federation. Consequently, we have not pursued this subject further in this publication.

Within the scope of our case study, incompatibilities in the data types, which were mandated for us as a result of feature differences in the various DBMSs used, have given us adequate opportunity to test solutions for these types of issues even though we have used essentially an identical schema in each case. See 4.4, “Considerations for use” on page 108, and 5.3, “Considerations for use” on page 149.

The distribution of the states among the regions in our case study is shown in a graphical approximation in Figure 2-6.

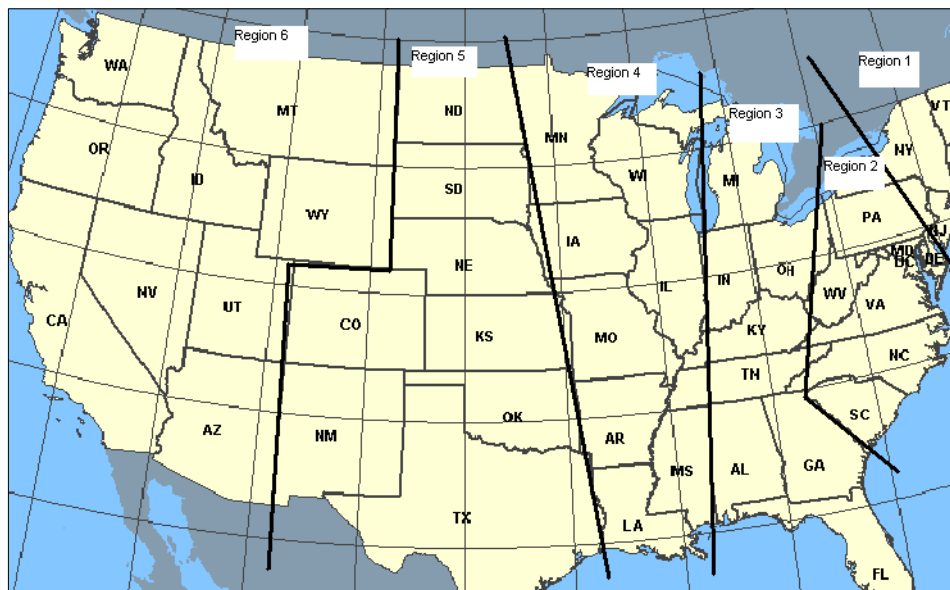


Figure 2-6 Map showing approximate distribution of states to regions

The actual distribution of states to regions for our case study is shown in Table 2-7 on page 42. This table shows for each state the region to which it is allocated and also the DBMS and operating system used to store that region’s data within the scope of our case study. For this purpose the following codes are used:

- ▶ ifxl - Informix Dynamic Server on Linux
- ▶ ifxw - Informix Dynamic Server on Windows
- ▶ oraw - Oracle on Windows
- ▶ xpsl - Informix Extended Parallel Server on Linux
- ▶ db2w - DB2 on Windows
- ▶ rbwl - Red Brick Warehouse on Linux
- ▶ db2l - DB2 on Linux

Note: Within our case study, none of the base schema definitions were altered as a result of the decision to federate the data. In other words, the region code has not been introduced as a column to any of the data source objects. This is important to understand when we get to Chapter 7.

Region	State Code	State Name	DBMS/Op Sys	Region	State Code	State Name	DBMS/Op Sys
Region 1	CT	Connecticut	ifxl	Region 4	AR	Arkansas	xpsl
Region 1	MA	Massachusetts	ifxl	Region 4	IA	Iowa	xpsl
Region 1	ME	Maine	ifxl	Region 4	IL	Illinois	xpsl
Region 1	NH	New Hampshire	ifxl	Region 4	LA	Louisiana	xpsl
Region 1	NJ	New Jersey	ifxl	Region 4	MN	Minnesota	xpsl
Region 1	NY	New York	ifxl	Region 4	MO	Missouri	xpsl
Region 1	RI	Rhode Island	ifxl	Region 4	MS	Mississippi	xpsl
Region 1	VT	Vermont	ifxl	Region 4	WI	Wisconsin	xpsl
Region 2	DC	D.C.	ifxw	Region 5	CO	Colorado	db2w
Region 2	DE	Delaware	ifxw	Region 5	KS	Kansas	db2w
Region 2	MD	Maryland	ifxw	Region 5	ND	North Dakota	db2w
Region 2	NC	North Carolina	ifxw	Region 5	NE	Nebraska	db2w
Region 2	PA	Pennsylvania	ifxw	Region 5	NM	New Mexico	db2w
Region 2	SC	South Carolina	ifxw	Region 5	OK	Oklahoma	db2w
Region 2	VA	Virginia	ifxw	Region 5	SD	South Dakota	db2w
Region 2	WV	West Virginia	ifxw	Region 5	TX	Texas	db2w
Region 3	AL	Alabama	oraw	Region 6	AZ	Arizona	rbwl
Region 3	FL	Florida	oraw	Region 6	CA	California	rbwl
Region 3	GA	Georgia	oraw	Region 6	ID	Idaho	rbwl
Region 3	IN	Indiana	oraw	Region 6	MT	Montana	rbwl
Region 3	KY	Kentucky	oraw	Region 6	NV	Nevada	rbwl
Region 3	MI	Michigan	oraw	Region 6	OR	Oregon	rbwl
Region 3	OH	Ohio	oraw	Region 6	UT	Utah	rbwl
Region 3	TN	Tennessee	oraw	Region 6	WA	Washington	rbwl
				Region 6	WY	Wyoming	rbwl
				Region 7	AK	Alaska	db2l
				Region 7	HI	Hawaii	db2l
				Region 7	PR	Puerto Rico	db2l

Figure 2-7 Distribution of states to regions for case study

Populating the databases

Within our case study we made design decisions regarding which data would be populated in the physical databases for each region. In our case, each region was represented by a combination of a DBMS instance and an operating system. This was done to satisfy our original intention to keep our case study as realistic as possible.

Since the `manufact`, `stock`, `catalog`, and `call_type` tables in the `Stores Demo` schema do not have any state-related information, it seems reasonable to populate these tables with the same data for each region. From a business viewpoint, this means that all regions within the corporation use the same list of

suppliers, offer the same inventory, produce nationwide catalogs, and use the same customer call classifications.

The state and customer tables in the Stores Demo schema both have columns containing the state code. Initially these tables were populated with data such that each region (DBMS instance) contained only rows for the state codes specified as belonging to that region in Table 2-7 on page 42. Consequently, when we populated the remaining orders and items tables with test data, we were mandated by the referential integrity within the schema to use only customers from that region.

In fact we used DB2 Information Integrator to populate all the databases in our case study. Initially we created an additional copy of the schema in a totally separate IDS database (not one used in our case study). Into that test IDS database we inserted all our test data for the case study. We then used it as the source from which to populate all the individual databases to be used for the case study. We did that by using a series of federated inserts of the form shown in Example 2-1. These federated inserts first select data from a remote data source (in our case the IDS test database) and inserts the data into a similar table in the schema but located on one of the remote regional data targets (for example, Oracle).

Example 2-1 A sample federated insert query

```
insert into source1_table1
      select * from source2_table1
```

We even used the same technique when inserting into a remote Informix table from an Informix table at the same remote data source. Refer also to “Query 3 - Consolidating the data” on page 167 for more details of federated inserts.

Maybe there will be situations where you decide not to federate your data for some of the reasons stated in “Arguments against data federation” on page 22, but *think* about the possibilities this capability could create for you in your data migration or data consolidation work. For example, now *you* may be able to accomplish most or all of *your* data migration requirements in a simple set of SQL statements written in the one DB2 SQL dialect, thus minimizing the level of skill required with the different source system DBMSs in your organization.

2.3 Naming conventions

In order to progress further with the understanding of our case study, it is now necessary to explain the naming conventions used in this publication. The conventions we used were mainly chosen to aid clarity for you in understanding

our rather complex environment with the same database schema implemented in seven different DBMS instances.

2.3.1 Database servers

First we defined names for our two database servers (refer to Figure 2-1):

- ▶ winsvr: Our Windows/2000 Server FP3 server (server A in Figure 2-1)
- ▶ linsvr: Our SuSE Linux V8.2 Server (server C in Figure 2-1)

2.3.2 Database identifiers

If you recall, for each region in our case study we stored the data in a separate combination of a DBMS and server. So as the major basis of our naming convention, we allocated a unique four-character identifier to each DBMS/server combination within our environment. Within this four-character database identifier, the first three characters indicate the DBMS used and the fourth character indicates the server on which it resides.

The database identifiers allocated within our naming convention are shown in Table 2-1.

Table 2-1 Database identifiers used in our naming convention

Region	DBMS	Server	Database Identifier
Region 1	IDS	linsvr	ifxl
Region 2	IDS	winsvr	ifxw
Region 3	Oracle9i	winsvr	oraw
Region 4	Informix XPS	linsvr	xpsl
Region 5	DB2	winsvr	db2w
Region 6	Red Brick	linsvr	rbwl
Region 7	DB2 UDB	linsvr	db2l

Tip: The use of this database identifier is not a prerequisite for database federation, but more as an aid to clarity. We do however recommend that you give consideration as to how you will identify your remote data sources within the one single federated database before you set up your federated environment.

2.3.3 References to remote tables within the federated database

In Chapter 4 and Chapter 5, we will see references to tables in the federated database. These are either nicknames, as is the case for DB2 II, or synonyms, and/or views, on remote tables, as is the case for Informix. Details are shown in the referenced chapters. In each case we have used identical naming conventions for these references to tables in the remote data sources.

For each such reference to a remote table in the federated database, we have prefixed the remote table name with the database identifier from Table 2-1 on page 44. Thus the stock table in the Red Brick Warehouse DBMS on the server linsvr is known as rbwl_stock in both the DB2 II and Informix federated databases.

By choosing the same naming convention for both federated databases, we were able to use basically the same SQL statements against each.

Note: We do not anticipate that you would necessarily need to federate the same tables within two (or more) federated databases. We federated our data twice for informational purposes so we could test the two environments and document the results.

For convenience and consistency we also created synonyms for the local tables in each federated database according to the same naming convention. For example, in the DB2 instance on winsvr, we defined the synonym db2w_stock for the local table stock.

Note: Later on in 6.2, “Testing with IBM Information Integrator” on page 94, and 6.3, “Testing with Informix Enterprise Gateway Manager” on page 95, when we examine the access plans, the local tables will be shown by the optimizer as their physical names instead of their synonyms. Other than this, the use of synonyms to make the local tables look similar to the remote tables is useful for our purposes in this publication.

2.3.4 User IDs

In order to fully explore both the possibilities and the necessities of different authorization scenarios with the use of a federated database, we established different user IDs for use with each database identifier.

For each database identifier, the database tables are owned by a different user ID. Within each remote data source we have granted authorization to a different end-user user ID. The naming convention for user IDs contains the same elements as those in the database identifier, DBMS, and server. In addition, the

user ID naming convention also indicates whether the user has been authorized in the remote source as a database administrator (DBA) or as an end user. In our case study, the end user specific to that database identifier has been specifically granted SELECT access to the tables at that remote data source. Authorizations to PUBLIC have not been used. This approach has allowed us to investigate exactly what the security and authentication requirements are when using each of the federated technologies.

Table 2-2 shows the user IDs used for creating tables and those users authorized to access tables in each of the data sources.

Table 2-2 User IDs used in each of the data sources

Region	DBMS	Server	Database identifier	DBA user ID	End user
Region 1	IDS	linsvr	ifxl	ifxdbal	ifxusr1
Region 2	IDS	winsvr	ifxw	ifxdbaw	ifxusrw
Region 3	Oracle9i	winsvr	oraw	oradbaw	orausrw
Region 4	Informix XPS	linsvr	xpsl	xpsdbal	xpsusr1
Region 5	DB2	winsvr	db2w	db2dbaw	db2usr1
Region 6	Red Brick	linsvr	rbwl	rbwdbal	rbwusr1
Region 7	DB2	linsvr	db2l	db2dbal	db2usr1

The user ID used to run the queries for our case study was the local end user for the relevant federated database. So for DB2 II, the end user db2dbaw was used. When running the same queries through the Informix federated database, the user ifxusr1 was used.



Overview of project data sources

With data federation we exploit the capability to access, manipulate, and use many heterogeneous data sources as if they were all the same data source. That is, we access them and perform operations on them, with a common SQL syntax.

This capability extends beyond access to relational data sources to nonrelational data sources and file structures of any kind. There are many benefits associated with this capability, including programmer productivity, minimization of skilled development resources, easier application development, added flexibility when using multiple heterogeneous data sources, ease of introduction of new data sources, and data source migrations.

In this chapter we give a brief overview of the data sources used in the development of this publication. Since the primary audience for this publication is customers who currently use Informix database products, we focused on the Informix product set and other data sources that would typically be used by those customers. The following are the data sources used in this publication:

- ▶ Informix Dynamic Server
- ▶ Informix Extended Parallel Server
- ▶ IBM Red Brick Warehouse
- ▶ IBM DB2 UDB
- ▶ Oracle9i

▶ Microsoft Excel

You will no doubt be familiar with one or more of these data sources. Some are focused more on OnLine Transaction Processing (OLTP) applications, others on data manipulation, and still others on query, analysis, and data warehousing applications—called Decision Support Systems (DSS).

Data source selection and usage is determined by a number of criteria, a few of which are:

- ▶ Performance
- ▶ Cost
- ▶ Availability of packaged applications
- ▶ Reliability and availability
- ▶ Data manipulation functions
- ▶ Availability of skilled resources
- ▶ Level of support

Our objective in providing these overviews is not for comparative purposes or to determine the relative importance or value of the data sources. It is simply to provide information on their high-level functions and features for those who may not be familiar with a particular data source. These brief descriptions will aid your understanding as you read this publication. They will hopefully provide you with enough information to give you a level of understanding of a particular DBMS product to eliminate the need for you to search out other documents in order to get that information. You hopefully have all the information you need in this one publication.

3.1 DB2 UDB

This section provides a brief overview of some of the basic functions and features of DB2 UDB. It is not intended to be an in-depth product description, but just some basic information to enable you to understand terms and discussions presented in this publication.

DB2 is IBM's premier relational database software, and is the worldwide market share leader in the industry. It is a complete multimedia, Web-ready, relational database management system. Also, it is proven to be flexible enough to meet the requirements of companies large and small. There are more than 60 million DB2 users from 400,000 companies worldwide.

3.1.1 DB2 UDB architecture

In Figure 3-1 on page 51 we depict the DB2 UDB architecture. From a client-server point of view, DB2 UDB separates the client code and the server code into different address spaces. The application code runs in the client process while the server code runs in separate processes. Each client application links with the DB2 client library, and communicates with the DB2 server using shared memory for local clients and a communication protocol, such as TCP/IP, for remote clients.

The following is a brief description of some of the components in the DB2 UDB architecture:

- ▶ DB2 agents

DB2 agents include coordinator agents and subagents, and are the most common type of DB2 processes that carry out the bulk of SQL processing on behalf of applications. DB2 assigns a coordinator agent with an application, and this agent coordinates the communication and processing for this application.

If intra-partition parallelism is disabled (this is the default), then the coordinator agent performs all the application's requests. If intra-partition parallelism is enabled, then DB2 assigns a set of subagents to the application to work on processing the application requests.

- ▶ Buffer pool

This is an area of memory where user table data, index data, and catalog data are temporarily moved from disk storage. DB2 agents read and modify data pages in the buffer pool. The buffer pool impacts performance, because data can be accessed much faster from memory than from a disk. Buffer pools can be defined with varying page sizes including 4 k, 8 K, 16 K, and 32 K.

- ▶ Prefetchers

These are designed to improve performance by retrieving data from disk and moving it into the buffer pool—*before* the application needs the data. For example, applications needing to scan through large volumes of data would have to wait for data to be moved from disk into the buffer pool if there were no data prefetchers.

- ▶ Page cleaners

These are present to make room in the buffer pool, before agents and prefetchers read pages from disk storage and move them into the buffer pool. For example, if an application has updated a large amount of data in a table, many of the updated data pages in the buffer pool may not yet have been written on to disk storage—such pages are called dirty pages. Since prefetchers cannot use dirty pages in the buffer pool, these dirty pages must first be flushed to disk storage and become “clean” pages, so they can again be used.

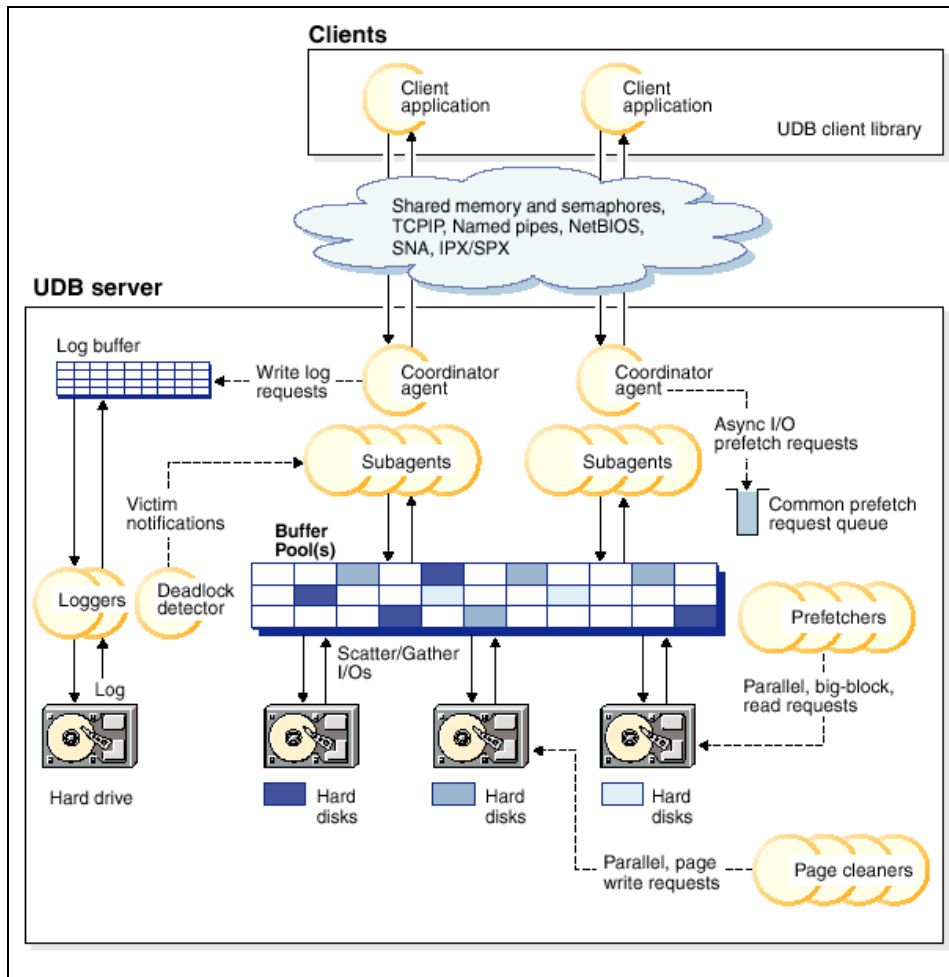


Figure 3-1 DB2 UDB architecture

- ▶ **Logs**
 Changes to data pages in the buffer pool are written to a special disk and logged there by agent processes. These logs are important for recovery and maintaining data integrity.
- ▶ **Deadlock**
 A deadlock occurs when one or more applications require access to a resource that is currently locked by another application(s). This can result in interminable waits by all the applications involved in the deadlock. To avoid such a situation, DB2 uses a background process called the *deadlock detector* to identify and resolve these deadlocks.

For more information on DB2 architecture, please refer to the *IBM DB2 UDB Administration Guide: Performance V8*, SC09-4821.

3.1.2 DB2 UDB functions and features

DB2 is highly scalable and highly extensible, yet very easy-to-use and manage. The following is a condensed list of some of the high level features of DB2 UDB:

- ▶ Scalability
DB2 UDB scales from a laptop to massively parallel systems with terabytes of data and/or thousands of users. This enables you to grow with the same RDBMS as your requirements grow. That brings with it the added benefits of minimizing costs for skilled resources and the ability to satisfy requirements for your application needs, be they OLTP or Decision Support.
- ▶ Multimedia extensibility
This feature includes the ability to support more advanced applications involving multimedia data such as documents, images, audio, and video. You can now develop applications that use this technology and other such technologies as XML, spatial and high-speed searching extenders, and object-relational extensions to the SQL language.
- ▶ Web-enablement
DB2 UDB is integrated with Web technology and positioned to enable you to build the e-business or Internet applications that are so critical in today's environment.
- ▶ Business intelligence support
DB2 UDB has strong offerings in support of business intelligence, such as data warehousing and online analytical processing (OLAP). DB2 provides parallel database technology and query optimization that can save on your processing costs and give you excellent performance.
- ▶ Ease of use and management
DB2 UDB is an easy database to set up, use, and manage. It includes a complete suite of GUI administration tools that allow for easy installation, administration, and remote operation.
- ▶ Reliability
DB2 UDB sets the standard for quality and reliability in the client/server database industry. The high level of reliability and availability provides the confidence you need when building mission-critical applications.
- ▶ Information Integration
DB2 provides a strong foundation of information integration technologies, including federation, consolidation, replication, Web services, and XML

support. For example, with DB2's built-in capabilities you can query, update, and replicate data across DB2 and IDS data sources.

- ▶ Enriched data type support

DB2 provides rich type support for spatial data, text data, and flat files, via DB2 Spatial Extender, DB2 Net Search Extender, and DB2 Data Links Manager, respectively.

- ▶ Connection concentrator

Reduces memory consumption on the database server by allowing transactions from remote clients to be concentrated or multiplexed across a small number of persistent database connections.

- ▶ DB2 Warehouse Manager

DB2 Warehouse Manager is a foundation element for business intelligence (BI) and customer relationship management. DB2 Warehouse Manager provides an infrastructure that helps you build, manage, and access the data warehouses that form the backbone of your BI solution. You can then integrate other BI tools and applications with this framework to ensure that you get the timely information need to make your business decisions.

This section has given you a very brief overview of some of the DB2 capabilities, functions, and features. The architecture section should give you some familiarity with DB2 UDB and how it functions. It is meant to be a basis to facilitate understanding of the sections on data federation later in this publication.

3.2 Informix Dynamic Server (IDS)

This section provides a brief overview of Informix Dynamic Server (IDS). It is not intended to be an in-depth product description, and does not describe all of the functions and features of IDS. Rather, it presents some of the basic concepts, architecture, and functions and features that contribute to the powerful capabilities of the product. It will enable you to better understand the terms and discussions presented in this publication. Since it primarily supports OLTP applications, many of the features discussed are centered around the ability to support high availability, fast data loads, fast backup and restore, and high performance. However, it also provides excellent support for DSS and BI systems.

IDS actually is not a traditional Relational DataBase Management System (RDBMS), rather it uses an object-relational model, so it should really be referred to as an Object-Relational DataBase Management System (ORDBMS).

RDBMS support consists of simple data types and pre-existing functionality defined in the database server. The data types provided by a traditional RDBMS are adequate to describe most business models. However, there are more complex models that work with more complex data, such as large arrays of data, points in a grid, maps, and images. An ORDBMS provides support for these complex data types. The functionality provided by a robust RDBMS such as scalability, security, transaction recovery, and online backup and restore are also required and provided in an ORDBMS. An ORDBMS takes advantage of the object-oriented concepts, for example, that enable the more specific definition of a business model.

As an ORDBMS, IDS provides both relational and object-oriented capabilities. For example, IDS enables you to extend your database by defining new data types, and user-defined routines (UDRs) to perform operations on those new data types. You can create UDRs in stored procedure language (SPL), C, or Java to extend the functionality of the database.

DataBlade™ modules are a means of packaging data-type definitions and the functions that operate on them. IBM Informix and third-party vendors package some data types and their access methods into DataBlade modules, or shared-class libraries. DataBlade modules allow you to store and manipulate complex data without having to create all the definitions and routines. They often plug directly into the components of the Dynamic Server for immediate use.

In Figure 3-2 we have depicted the IDS architecture as an implementation of an ORDBMS.

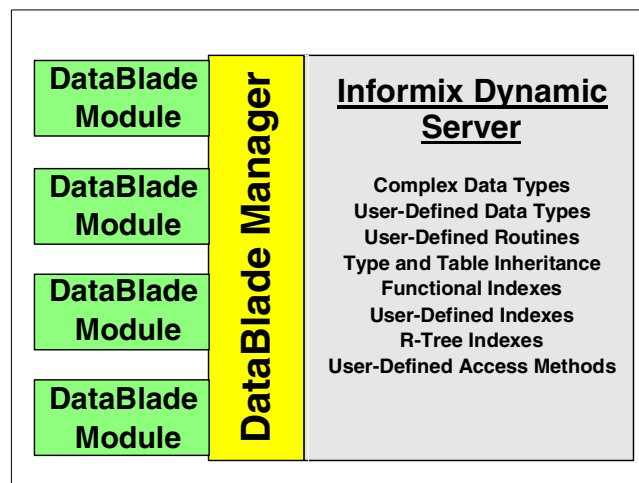


Figure 3-2 IDS ORDBMS architecture

IDS is the Informix flagship database server. It is the best-of-breed, general-purpose, but mission-critical, OLTP database for e-business. It delivers the performance, reliability, scalability, and high-availability needed for today's global, e-business enterprises.

Key IDS features include:

- ▶ Data partitioning
- ▶ High Availability Data Replication (HDR)
- ▶ Enterprise Replication (ER)
- ▶ Parallel queries
- ▶ Fast loads and backups
- ▶ Support for industry standards such as ODBC, JDBC, and OLE/DB

IDS has two primary product lines: Version 7.x and Version 9.x. The IDS architecture, and most of the features, are supported on both families. However, there are some extended features only supported by Version 9.x.

IDS is a fast and powerful OLTP relational database management system based on the multi-threaded Dynamic Scalable Architecture (DSA) developed by Informix. This architecture requires fewer processes to execute multiple database activities. DSA was designed to provide efficient resource utilization so you need less hardware to support your growing business needs. That means that as more users are added to the system, required support resources can be added dynamically.

3.2.1 IDS architecture

The IDS architecture is depicted as an overview in Figure 3-3 on page 56. It is a general high-level view of the major components of the architecture and is discussed in this section.

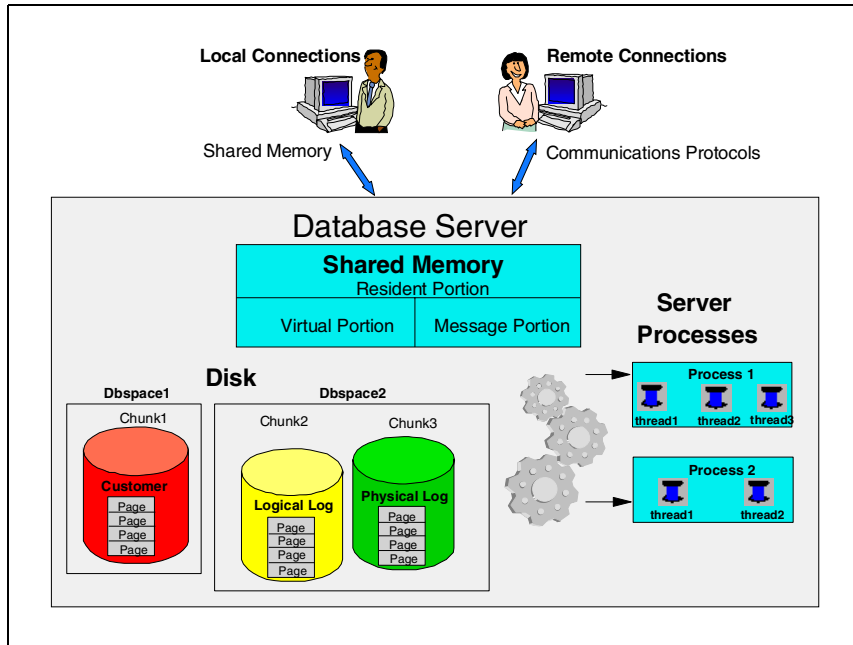


Figure 3-3 Informix Dynamic Server overview

The IDS architecture consists of three primary components:

► Multi-threaded Implementation

IDS uses a multi-threaded architecture. A thread is a set of instructions that are executed in a program, and these threads run in processes. Starting and stopping processes takes time and processor resources, so enabling many threads to reuse a process can result in significant time and resource savings. This is called multithreading.

► Database server

The database server consists of three major components, and each plays an important part in determining the overall server performance, workload capacity, response time, availability, reliability, and resource consumption. Those three components are:

- Process: This is the mechanism that schedules the Central Processing Unit (CPU) to execute the activities performed by the threads.
- Shared memory: All threads are executed in the memory of the CPU. Since each CPU can execute multiple processes, the memory needs to be shared among them. This is because it may take multiple iterations, or steps, before a process is completed, so memory must be saved while the other processes are being executed in the same CPU.

- Database server: The processes require a place to permanently store the results of their executions. Physical disk storage is used for this purpose. This disk space resides on the database server and is assigned to the specific processes.
- ▶ Client-server connectivity
This component enables the application, which is executed by application threads in the processes, to connect to the database server. There are a number of methods to connect, depending on whether the client is local or remote. For example, a local connection might be through shared memory and a remote connection through a communications protocol such as TCP/IP.

3.2.2 Functions and features

The usefulness of a DBMS depends, in addition to performance, on the functions and features provided to make it easy, simple, and cost effective to provide the capabilities required. Here are just a few of the IDS functions and features:

- ▶ Table and Index Partitioning (Fragmentation)
Data stored by applications in a relational DBMS are stored in tables. To better utilize the physical disk space, the DBMS will use different methods. One of those is called fragmentation. Fragmentation enables and manages the distribution of data from one table across many physical disks. The segments of the table are stored in areas called *dbspaces*. This is primarily for better performance.
- ▶ High Availability Data Replication (HDR)
HDR provides synchronous and asynchronous replication (making of a copy) of an entire database server on separate servers that, in some cases, can be located in different sites. This enables the data to still be available to applications, even if the primary database server becomes unavailable. Or it can be used to provide improved performance, for example, to applications executed at other remote sites. This enables local data to be used rather than relying on the performance of a communications link to a remote primary database server.
- ▶ Enterprise Replication (ER)
This component enables data to be copied to one or more separate local or remote database servers. It can be for the purpose of backup, or to improve performance at remote locations. Data at all locations is kept synchronized based on a predetermined time schedule. The primary difference between ER and HDR is that HDR replicates the entire database instance to a remote site, while ER replicates at a table level. In addition, HDR supports both synchronous (real-time) and asynchronous (scheduled or delayed update) data replication, whereas ER only supports asynchronous.

► Parallel Backup and Restore

IDS performs parallel backup and restore through ON-Bar, which is one of its Backup and Restore utilities. The other is *ontape*, which performs backup and restore serially.

ON-Bar can be used to make a backup copy of your database server data and logical logs to be used as insurance against lost or corrupted data. Data might be lost or corrupted for reasons that range from a program error to a disk failure to a disaster that damages the facility in which your computer resides. To recover data, restore the database in two steps: First restore the backup copy of the data, and then restore the logical logs to bring data as close as possible to the most recent state.

ON-Bar works in conjunction with another software layer called Storage Manager. They communicate with each other using the X/Open Backup Services Application Programmer's Interface (XBSA). IDS comes with a simple storage manager called Informix Storage Manager (ISM), but ON-Bar can also work along with IBM Tivoli® Storage Manager (TSM), and also other third-party storage managers, such as HP Omniback.

For speed and efficiency, ON-Bar can perform parallel backup and restore. For example, ON-Bar can back up multiple storage spaces concurrently. However, you can also configure ON-Bar to perform this task serially.

ON-Bar performs parallel backup and restore based on a configuration parameter, `BAR_MAX_BACKUP`. When ON-Bar receives a request, it determines how many objects are involved. If the request involves more than one object, ON-Bar creates a new `onbar_d` process for each object up to the limit that you specified in the `BAR_MAX_BACKUP` configuration parameter. Each new instance of ON-Bar creates a new XBSA session.

For more information about ON-Bar and parallel backup and restore, see the *IBM Informix Backup and Restore Guide, Version 9.4*, G251-1240.

► High Performance Loader (HPL)

The HPL is a feature of the database server that allows you to load and unload large quantities of data efficiently to or from a database. The HPL lets you exchange data with tapes, data files, and programs, and converts data from these sources into a format compatible with Informix databases. The HPL also allows you to manipulate and filter the data as you perform load and unload operations. HPL is much faster than other load/unload utilities, such as SQL load/unload commands, *onload/onunload*, and *dbload*.

HPL uses a client-server architecture, and is composed of these three utilities:

- *pload* utility: This is the graphical user interface, where you create load and unload jobs. These jobs are stored in the *onpload* database.

- onpload utility: The onpload utility performs the actual activity of converting and moving data. The onpload utility uses information from the onpload database to run the load or unload, and to convert the data.
- onpload database: The onpload database contains information that the onpload utility requires to perform data loads and unloads.

For more information about HPL, see the *IBM Informix High-Performance Loader User's Guide, Version 9.4*, G251-1255.

▶ User-defined routines (UDRs)

A routine is a collection of program statements that perform a specific task, and are used to extend the database capabilities. It can be invoked in an SQL statement (such as a SELECT) by the Database Server or by another UDR. UDRs are also supported on IDS 7.3, but they can only be written using the Stored Procedure Language (SPL). With IDS 9.x the UDRs can also be written in the C and Java programming languages.

▶ User-defined data types and complex data types

With IDS 7.x only built-in data types are available for use. Those data types include such types as CHAR, VARCHAR, INTEGER, and DECIMAL. With IDS 9.x you can use all the built-in data types, plus a new set of Extended data types. You can now create your own data type to satisfy all your requirements. The data types supported in IDS 9.x are depicted in Figure 3-4 on page 60

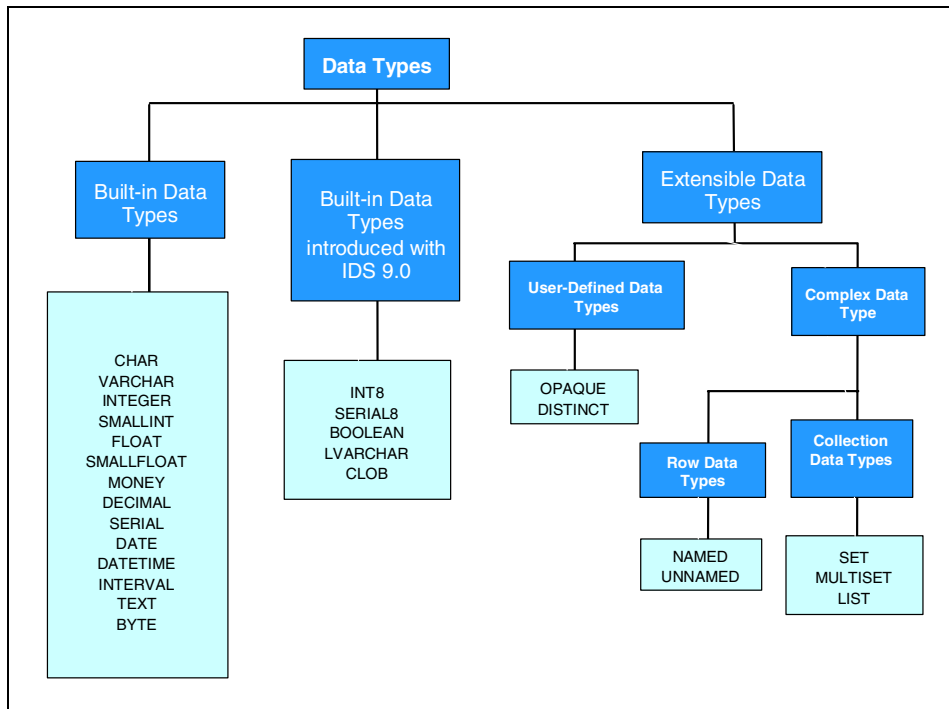


Figure 3-4 Data types

► SQL Statement Cache

In earlier versions of IDS (such as IDS 7.x) each session generated and stored its own SQL statement information. With Version 9.4, the database server allows sessions to share the SQL statement information that is generated. This allows multiple sessions that are executing identical statements to share the information stored in internal data structures. The SQL statement cache feature reduces memory utilization at the session level and eliminates the need to reparse and reoptimize statements that have been executed by other sessions. The database server maintains the SQL statement cache in the virtual portion of shared memory. As SQL statements are executed by various sessions, they are parsed, optimized, and stored in this cache. Only data manipulation language (DML) statements (SELECT, INSERT, UPDATE, and DELETE) are cached. The primary purpose of the SQL Statement Cache is to reduce the usage of memory, but some performance gains might also be realized.

► Fuzzy Checkpoints

The checkpoint is a major event on the database server. It can be defined as a point of synchronization between memory and disk, and is required to

maintain data integrity. Pages of data in memory are written to disk and then flushed from memory. During the checkpoint operation the system is frozen and no other database activity can occur. The flush of these pages is very costly and, depending on the number of pages in memory, the system can be unavailable for a critical period of time.

To reduce the checkpoint duration IBM introduced with IDS 9.2 a new feature called Fuzzy Checkpoints. The original checkpoint method still exists, but now it is referred as *full* or *sync* checkpoint. The intention is to avoid performance issues by writing fewer pages to disk during a checkpoint. To accomplish this, a certain subset of database operations has been designated as *fuzzy*. The buffers modified by these fuzzy operations are flagged, and are not written to disk when a checkpoint occurs. Fuzzy operations are inserts, updates, and deletes. However, they are only flagged as fuzzy operations if they are executed against a database that has logging and the buffered page to be changed is not considered old (as determined by an internal timestamp comparison).

This has been a very brief overview of IDS and its functions and features. It is a very robust DBMS that is primarily oriented to OLTP applications, but also works well, and is widely used, for data warehousing, business intelligence, and decision support query operations.

3.3 Informix Extended Parallel Server (XPS)

This section provides a brief overview of Informix Extended Parallel Server (XPS). It is not intended to be an in-depth product description, and does not describe all of the functions and features of XPS. Rather, it presents some of the basic concepts, architecture, and functions and features that contribute to the powerful capabilities of the product. It will enable you to more easily understand terms used and discussions presented in this publication. XPS is to primarily support data warehousing, business intelligence, and decision support types of applications. Many of the features discussed are centered around the ability to scan, filter, join, and organize response sets for large volumes of data, even that reside across a number of database servers.

XPS satisfies the complex needs of data warehouses and DSS applications. Designed specifically to handle the demands of very large databases (VLDBs), Informix Extended Parallel Server 8.3 offers a shared-nothing database engine, complementing the shared-nothing hardware systems. This highly optimized data engine utilizes all available hardware resources including CPU, memory, and disks, delivering mainframe-caliber scalability, manageability, and availability, while requiring minimal operating system overhead.

3.3.1 XPS architecture

XPS was architected from the beginning to support large data volumes and an ad-hoc decision support query environment that services large numbers of users. Much of this capability is enabled by use of the shared nothing architecture. It is implemented through the use of co-servers, as depicted in Figure 3-5 on page 63.

This approach to managing data greatly minimizes operating system overhead and reduces network I/O. To achieve this level of independence, each node runs its own instance of XPS that consists of basic services for managing its own logging, recovery, locking, and buffer management. This instance of the database is called a *co-server*. Each co-server *owns* a set of disks and the partitions of the database that reside on these disks.

To optimize performance, the system catalog containing information about the way in which the data is distributed can be cached across the nodes. Additionally, smaller, frequently accessed tables can be replicated across the nodes to further enhance performance. Although multiple instances of XPS are running, all of the co-servers cooperate to provide the image of a single, large server. This single-system image is provided to both general database users and also to system and database administrators.

Different clients can connect to different co-servers. The co-server that the client connects to is called the connection co-server. The connection co-server determines if a client request can be satisfied with data that resides on the co-server alone, or whether it requires data that resides on other co-servers. If the co-server requires data that resides on another co-server, it interacts and coordinates activities with participating co-servers. For example, client A connects to co-server 1. After determining that it does not have the necessary data to complete the request, co-server 1 (the connection co-server) calls co-server 2, which becomes the participating co-server with co-server 1, to complete the request (see Figure 3-5 on page 63).

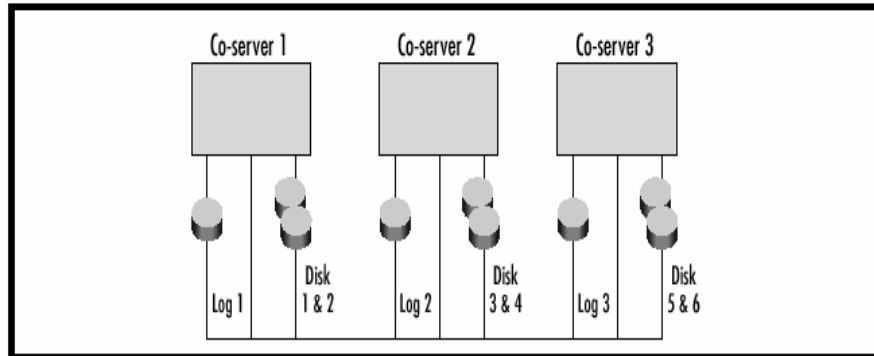


Figure 3-5 XPS architecture

Informix Extended Parallel Server can achieve this level of coordination by making intelligent decisions about how to divide the query, and where to send the SQL operations (distributed on different nodes) to be performed. The server modules responsible for making such decisions include the request manager, query optimizer, metadata manager, and the scheduler. These services modules are tightly integrated with the core kernel of Informix Extended Parallel Server.

The query optimizer is responsible for determining the best way to perform a specific query. It is cost-based, meaning the optimizer generates multiple query plans, computes a cost for each plan, then chooses the lowest-cost plan. A query plan is simply a distinct method of executing a query that takes into account the order in which tables are read, how they are read (by index or sequentially), and how they are joined with other tables in the query. Using pertinent information provided by the metadata manager, the query optimizer determines the degree of parallelism of the request, and sends the execution plan to the scheduler for distribution.

3.3.2 Functions and features

In addition to a shared-nothing database engine, Informix Extended Parallel Server 8.3 offers a complete set of features to further enhance the performance of decision support queries. These features include enhanced SQL extensions and join methods to improve the execution of large decision support queries, new indexing methods to accelerate access to data, and a graphical administration environment to ease management of the large volumes of data.

The following is a list of some of the newer features in XPS:

- ▶ Long identifiers

Provides longer identifiers for SQL objects. User names are expanded to 32 characters and SQL objects to 128 characters. This feature simplifies the use

of XPS by making it possible to use more natural naming schemes and to use the same naming scheme that information may have in other databases.

- ▶ I-STAR distributed query participant

There are two types of I-STAR participation. The first is that XPS can participate in an IDS-coordinated I-STAR transaction, but certain restrictions will exist. For example, binary large objects (BLOBs) will not be accessible. The second feature is that XPS can coordinate and select only the I-STAR transaction. That is, XPS will be able to select information from Informix Dynamic Server (IDS) databases.

- ▶ Roles and default roles

XPS implements the IDS version of roles. This feature allows roles to be defined and users to be assigned certain roles. This feature simplifies security planning for XPS databases. One specific part of this work is the ability to set a default role for users who have no other role assigned.

- ▶ Snowflake schema enhancements

This feature enables better recognition of snowflake schemas and has significantly improved the performance of queries run against this type of schema.

- ▶ External table support for greater than 2 GB files

This feature allows users to load and unload data, on 64-bit platforms, to files larger than 2 GB in size, facilitating the handling of large amounts of data.

- ▶ Rapid, efficient, fully parallel query processing

Makes full use of all available hardware resources to deliver mainframe-caliber capability, manageability, and performance, while requiring minimal operating system overhead. The smart IBM Informix XPS optimizer determines the best query plan and can combine several methods of joining tables in a single query plan for the most efficient use of memory and processing power.

- ▶ Fast, easy expandability

The dynamic co-server management (DCM) feature lets you add nodes to your system to expand database server capacity, either temporarily or permanently. When end-of-month processing causes a temporary system overload, one or more specific-purpose co-servers can distribute the processing load to enable parallel processing tasks to be accomplished while normal operations continue. When data requirements push the limits of your current database capacity, it is easy to add permanent co-servers to the database server to contain the additional tables or table fragments.

- ▶ Flexible database design

XPS provides table fragmentation methods that are appropriate for normalized or denormalized relational database schemas. You can choose the database schema that best fits your data and queries, and then determine the fragmentation method to distribute data across co-servers for optimum performance—even on specific queries.
- ▶ Rapid data loading and unloading

The IBM Informix XPS parallel data loader quickly adds new data for immediate use, checking constraints as it loads. The parallel loader is also a parallel unloader that enables quick downloads of data from your data warehouse to data marts or other special-purpose data stores.
- ▶ Ease of management

With IBM Informix Server Administrator, you can manage your IBM Informix XPS database from any computer that has a Web browser.

3.4 IBM Red Brick Warehouse

This section provides a brief overview of IBM Red Brick Warehouse (Red Brick or RBW). It is not intended to be an in-depth product description, and does not describe all of the functions and features of Red Brick. Rather, it presents some of the basic concepts, architecture, and functions and features that contribute to the powerful capabilities of the product. It will enable you to more easily understand terms used and discussions presented in this publication. Since it primarily supports Decision Support applications, many of the features discussed are centered around the ability to scan, filter, join, and organize response sets for large volumes of data. Though it is an RDBMS, it is architected to primarily support star data schemas. These types of schemas can enable very fast queries against large volumes of data.

Red Brick is a database server designed to meet the specialized requirements for business-critical, high-demand data analysis. It provides a robust and scalable platform for developing star schema-based decision support applications, through the application of innovative technology. Designed to be practical, cost-effective, and scalable, Red Brick enables more users to analyze more data and make better decisions faster than with a general-purpose database. Highlights include:

- ▶ High-speed, high-volume query performance
- ▶ High-performance data management
- ▶ Linear scalability for growth
- ▶ Low cost of ownership

Red Brick provides efficient processing of large data sets and delivers outstanding performance. It is designed to be a *hands off* database engine. As such, it reduces deployment time as well as hardware and maintenance costs.

3.4.1 Red Brick architecture

The Red Brick environment is depicted in Figure 3-6. It has all the functionality required to retrieve data from heterogeneous data sources and load it into the Red Brick Warehouse. The data is managed by using Red Brick and business partner administration tools. Also, data can be aggregated for faster query performance. Clients and client applications can then access and analyze the data with ease, through mechanisms such as ODBC and JDBC.

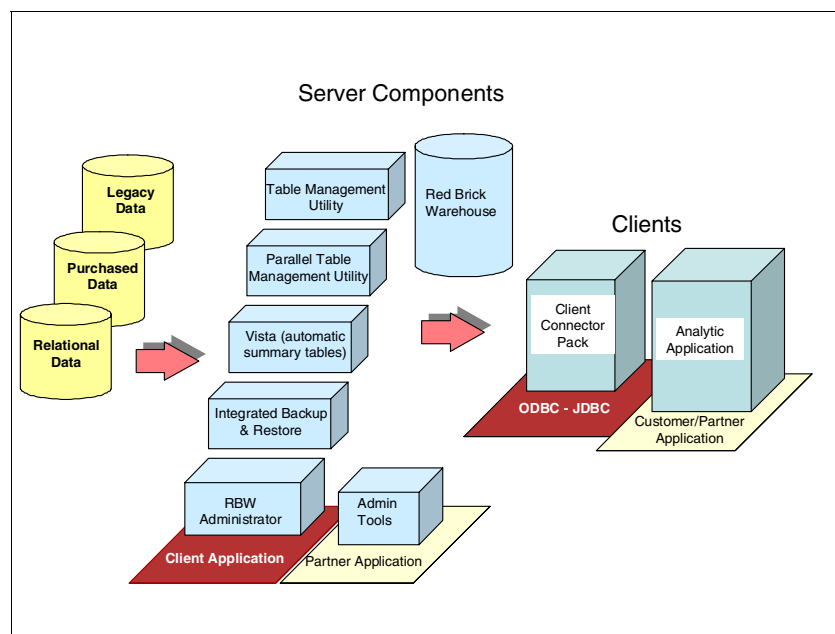


Figure 3-6 Red Brick Warehouse environment

Red Brick enables very fast query performance by means of specialized join technologies. These are depicted in Figure 3-7 on page 67.

These join technologies are enabled by unique indexing technologies. For example, the binary join is based on indexing schemes such as B-tree and hybrid hash. It is rarely used in decision support applications because the join performance is typically not fast enough.

However, there are other types of indexing that offer better join performance. For example, the STARindex relates rows in the various tables of a database that is based on a star schema. That is, it can index across tables. This is opposed to a traditional schema based on normalized relational tables where an index is generated only on a single table.

STARjoin is a critical Red Brick performance technology. With STARjoin, multiple relational tables are joined at one time in a fast, single-pass operation. Red Brick is able to process joins from three tables up through 20-plus tables in a single step with orders of magnitude performance benefits. In contrast, many join technologies must join tables pair-by-pair.

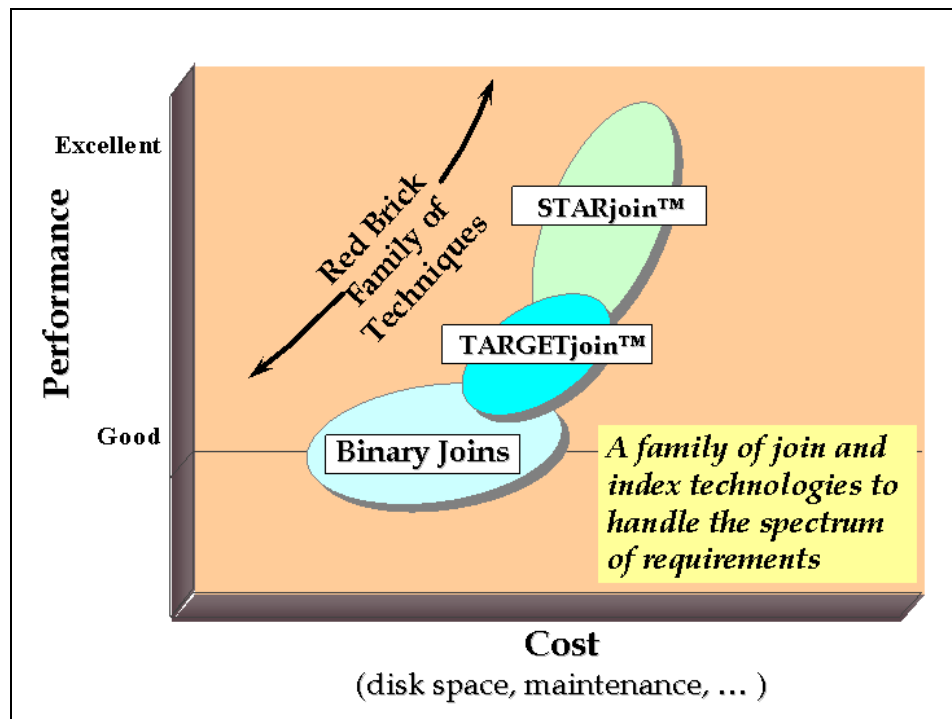


Figure 3-7 Red Brick join techniques

We show in Figure 3-8 on page 68 an example of star indexing. It demonstrates how this indexing scheme can provide the performance improvements in join operations.

As you can see with the traditional indexing scheme, there is a separate index for each table. Therefore to join the three tables X, Y, and Z, you must join the three indexes. Many times they must be joined two at a time, whereas the STARindex is built across all three tables. Once this is done, queries can be executed with a

single pass of the STARindex. This can give a significant performance improvement.

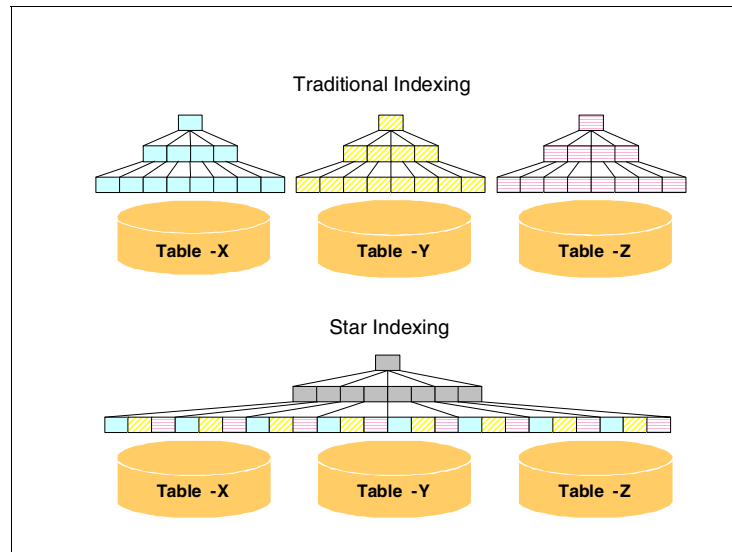


Figure 3-8 Red Brick star indexing

For the TARGETjoin technology, TARGETindexes are used. This is the Red Brick implementation of a bitmap index. A bit-mapped index can dramatically reduce query response time by greatly reducing the number of read operations required. In addition, the Red Brick implementation can handle all ranges of cardinality. It automatically selects the appropriate cardinality type based on the values in the column of data.

These unique capabilities make Red Brick a great choice for data marts and data warehouses, and decision support applications. In addition, there are many other functions and features we can discuss with Red Brick.

3.4.2 Functions and features

Red Brick Warehouse comes with many functions and features that you can leverage for your data management needs. The following list briefly summarizes some of those features:

- ▶ Query performance

Red Brick provides high-speed, high-volume query performance in analytic data warehouse environments through several key technology features. These include highly optimized indexing, aggregation, and parallelism technologies.

- ▶ High-performance data management

Red Brick server supports high-speed, high-volume data loading within a single, integrated process. The superior data-loading process of the server provides maximum efficiency and automation, enabling easier and quicker availability of query-ready data.
- ▶ Linear scalability for unlimited growth

The linear scalability of Red Brick Warehouse, and its ability to support increasing numbers of users without impacting performance, enables it to easily adapt to large volumes of data.
- ▶ Aggregate maintenance

Red Brick Vista®™ is a comprehensive, server-integrated solution for aggregate computation and management. When detail tables are modified either through server-based DML operations or through the bulk data loader, the modifications are automatically propagated to the aggregates. This server-integrated solution provides an automatic and efficient way of keeping aggregates in sync with detail data, simplifying the administration tasks associated with aggregates while maximizing their usability.
- ▶ Random sampling of data

This is useful for applications such as data mining. It allows users to efficiently produce approximate results based on sampled data sets. Sampling can be performed on any table or sub-select in a query, and can be block based or row based.
- ▶ Data loading

This includes optimizations and support of parallelism for loading and for modifications to existing data. The bulk-loading program is called the Table Management Utility (TMU). It performs inserts and updates, checks referential integrity, builds indexes, and performs aggregate maintenance. Indexes and aggregates are built in a single pass, reading the input source data only once.
- ▶ Query optimizations

These include improved plan selection for queries involving multi-table joins.
- ▶ Query plan transformations

These enable better utilization of join techniques, and indexes, for more efficient processing of queries containing multiple blocks.
- ▶ Database connectivity

The Informix Red Brick JDBC™ and ODBC Drivers now support JDBC V2.0 and ODBC V3.5, respectively.

- ▶ Backup and restore
Simplifies administration, speeds data discovery for restores, provides online backups, and adds more device and third-party tools support with Xopen Backup and Restore APIs (XBSA).
- ▶ Enhanced XML support
Enables you to load large amounts of relational data exchanged through XML. This feature also allows high-speed export of data to XML format.
- ▶ Locally segmented indexes
Allows a B-TREE or TARGETindex to be segmented like the table even though the segmenting column of the table does not appear in the index. Most useful for creating the indexes for TARGETjoin processing.
- ▶ Segment DDL improvements
Simplifies specification of STARindex segmentation with new syntax.
- ▶ Remote Load Table Management Utility (TMU)
Allows you to use the TMU to load data from a local machine to a remote database.
- ▶ SQL/OLAP
Implements elementary OLAP operators. SQL/OLAP consists of a number of language extensions that facilitate analytical processing.
- ▶ Vista Query Rewrite Extensions
The existing rewriter allows rewrite of queries that involve COUNT DISTINCT, MIN, and MAX aggregations, provided that columns containing these aggregations are present in the corresponding pre-computed views.

3.5 Oracle9i and Microsoft Excel

Oracle9i and Microsoft Excel were also used in the development of this publication. The intent was simply to demonstrate that DB2 Information Integrator could access, join, and manipulate data from a variety of heterogeneous relational and nonrelational data sources, along with Informix and IBM relational data sources. Also, that this can be done using the same SQL. This enables you to access and manipulate data from all these sources, including Microsoft Excel, as if they were all part of the same database.

Both Oracle9i and Microsoft Excel are data sources prevalent in the marketplace, and in use by many companies. We did not feel it appropriate or necessary to provide product overviews for those products in the publication. It is sufficient to know that we can include them in our federated environment. We do provide

considerations for using them, in the following chapters, as appropriate for a better understanding of data federation.



DB2 Information Integrator

DB2 Information Integrator V8.1 is a product designed to access structured and unstructured information across and beyond the enterprise. It enables optimized access to, and federation of, data from numerous heterogeneous data sources.

Components of DB2 II include a federated data server and a replication server to integrate diverse data types on demand in real time. Applications that use SQL or tools that generate SQL (for example, integrated development environments or reporting and analytical tools) can access, integrate, and manipulate distributed and diverse data through a federated system server as if it were a single data source.

In this chapter we provide an overview of DB2 Information Integrator V8.1. We guide you through the installation and setup process, and then discuss some of the considerations when using this product in a federated data environment.

4.1 Product overview

DB2 Information Integrator allows you to access different types of data sources through a single federated server. With applications that use SQL, or that generate SQL, you can perform the following tasks:

- ▶ Access traditional forms of data and emerging data sources.
- ▶ Use data that is structured, semi-structured, and unstructured.
- ▶ Retrieve, update, transform, and replicate information from diverse distributed sources.

In the following section, we describe the architecture and the basic components of a federated system that is based on DB2 Information Integrator.

4.1.1 Systems environment

A DB2 federated system is a special type of distributed database management system (DBMS). The federated system consists of a DB2 instance that operates as a federated server, a database that acts as the federated database, one or more data sources, and clients (users and applications) that access the database and data sources. With a federated system, you can send distributed requests to multiple data sources within a single SQL statement. For example, you can join data that is located in a DB2 Universal Database™ table, an Informix table, and an XML tagged file in a single SQL statement. Figure 4-1 shows the components of a federated system and the data sources you can access.

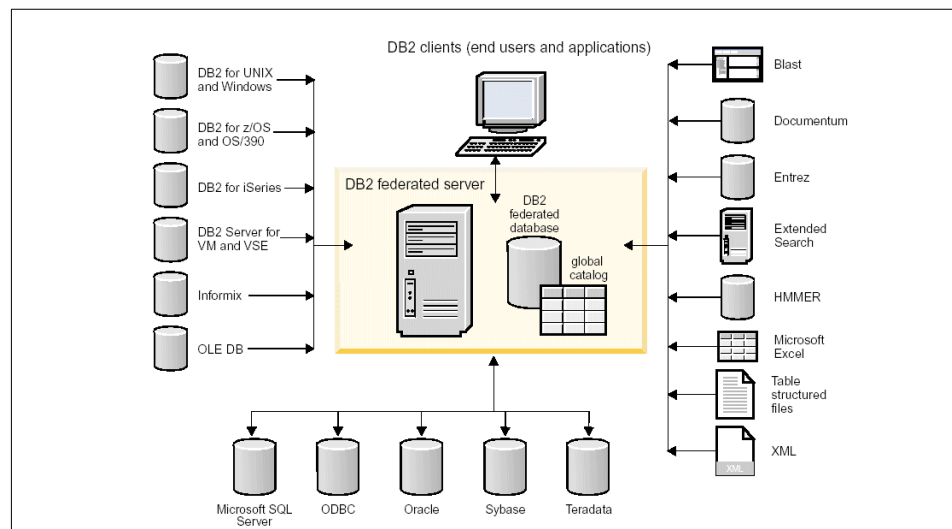


Figure 4-1 The components of a federated system and the supported data sources

The power of a federated server is the ability to:

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

Federated server

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

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

Data sources

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

The method, or protocol, used to access a data source depends on the type of data source. For example, DRDA is used to access DB2 for z/OS™ and OS/390® data sources, and the Informix Client API is used to access Informix data sources. Although there are different protocols and access methods, you still access the data sources through the federated server using native DB2 SQL.

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

Table 4-1 shows the currently supported data sources.

Table 4-1 Supported data sources

Type	Data source
Relational data source	DB2, Informix, Oracle, Sybase, Teradata, MS SQL Server, ODBC, OLEDB
Nonrelational data sources	BLAST, Documentum, Entrez, HMMER, IBM Lotus Extended Search, Microsoft Excel, Table-structured files, XML

Wrappers and wrapper modules

Wrappers are mechanisms by which the federated server interacts with data sources. The federated server uses routines stored in a library called a *wrapper module* to implement a wrapper. These routines allow the federated server to perform operations such as connecting to a data source and retrieving data.

Usually you create only one wrapper for each type of data source. For example, suppose that you want to access three DB2 for z/OS database tables, one for DB2 for iSeries™ table, two Informix tables, and one Informix view. You need to create only two wrappers: One for the DB2 data source objects and one for the Informix data source objects. Once these wrappers are registered in the federated database, you can use these wrappers to access other objects from those data sources.

Server definitions and server options

After wrappers are created for the different data sources, the federated instance owner defines the data sources to the federated database. For relational data sources you need to know the connection-specific information. For example, a connection to a specific Informix source is specified through the name of the remote Informix server and the name of the remote database.

Some of the information within a server definition is stored as server options. When you create server definitions, it is important to understand the options that you can specify about the server. Some server options configure the wrapper and some affect the way DB2 uses the wrapper.

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

See further details in 4.3, “Creating and using wrappers and nicknames” on page 97

User mapping and security

In a federated environment the user or application is only connected to and authenticated at the federated server. When the federated server needs to push down a request to a remote data source, the server must first establish a connection to the data source.

For most of the data sources, the federated server does this by using a valid user ID and password to that remote data source. When a user ID and password are required to connect to a data source, you must define an association between the federated server user ID and password and the data source user ID and password. This association must be created for each user ID that will be using the federated system to send distributed requests. This association is called *user mapping*.

Nicknames and data source objects

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

Nicknames are different from aliases that exist in DB2. They are pointers by which the federated server references these objects. Nicknames are typically defined with the CREATE NICKNAME statement. The wrapper provides a default mapping between the data types that are used in the data source and the data types that are available in DB2.

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

Suppose that you define the nickname DEPT to represent an Informix database table called NFX1.PERSON. The statement SELECT * FROM DEPT is allowed from the federated server. However, the statement SELECT * FROM NFX1.PERSON is not allowed from the federated server (except in a pass-through session—see “Pass-through sessions” on page 81).

Column options

You can supply the federated database server with additional metadata information about the nicknamed object. This metadata describes values in certain columns of the data source object. You assign this metadata to parameters that are called *column options*. The column options tell the wrapper

to handle the data in a column differently from the default behavior. The SQL compiler and query optimizer use the metadata to develop better plans for accessing the data.

Suppose that a data source has a collating sequence that differs from the federated database collating sequence. The federated server typically would not sort any columns containing character data at the data source. It would return the data to the federated database and perform the sort locally. However, suppose that the column is a character data type (CHAR or VARCHAR) and contains only numeric characters ('0','1',..., '9'). You can indicate this by assigning a value of 'Y' to the NUMERIC_STRING column option. This gives the DB2 query optimizer the option of performing the sort at the data source. If the sort is performed remotely, you can avoid the overhead of porting the data to the federated server and performing the sort locally.

Data type mapping

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

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

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

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

For relational data sources, you can override the default data type mappings, or create mappings when there is no default. For example, you can create a type mapping when a new built-in type is available at the data source, or when there is a user-defined type at the data source that you want to map to.

Function mapping

For the federated server to recognize a data source function, the function must be mapped against an existing counterpart function in DB2 UDB. DB2 Information Integrator supplies default mappings between existing built-in data source functions and built-in DB2 counterpart functions. For most data sources, the default function mappings are in the wrappers.

For relational data sources, you can create a function mapping when you want to use a data source function that the federated server does not recognize. The mapping that you create is between the data source function and a DB2 counterpart function at the federated database. Function mappings are typically used when a new built-in function or a new user-defined function becomes available at the data source. Function mappings are also used when a DB2 counterpart function does not exist.

Index specifications

When you create a nickname for a data source table, information about any indexes that the data source table has is added to the global catalog. The query optimizer uses this information to expedite the processing of distributed requests. The catalog information about a data source index is a set of metadata, and is called an *index specification*.

Significant changes at the data source object, such as when you add or drop indexes, should be reflected in the index specification in DB2. Currently the federated instance owner/DBA is responsible for updating this information. For more details see “The federated database system catalog” on page 79.

Federated database

To end users and client applications, data sources appear as a single collective database in DB2. Users and applications interface with the *federated database* managed by the federated server. The federated database contains a system catalog that contains entries that identify data sources and their characteristics. The federated server consults the information stored in the federated database system catalog and the data source wrapper to determine the best plan for processing SQL statements.

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

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

The federated database system catalog

The federated database system catalog contains information about the objects in the federated database and information about objects at the data sources. The catalog in a federated database is called the global catalog because it contains information about the entire federated system. DB2 query optimizer uses the

information in the global catalog and the data source wrapper to plan the best way to process SQL statements. The information stored in the global catalog includes remote and local information, such as column names, column data types, column default values, and index information.

The query optimizer

As part of the SQL compiler process, the query optimizer analyzes a query. The compiler develops alternative strategies, called *access plans*, for processing the query. Access plans might call for the query to be processed:

- ▶ By the data sources
- ▶ By the federated server
- ▶ Partly by the data sources and partly by the federated server

DB2 evaluates the access plans primarily on the basis of information about the data source capabilities and the data. The wrapper and the global catalog contain this information. DB2 decomposes the query into segments that are called *query fragments*. Typically it is more efficient to *push down* a query fragment to a data source, if the data source can process the fragment. However, the query optimizer takes into account other factors such as the:

- ▶ Amount of data that needs to be processed
- ▶ Processing speed of the data source
- ▶ Amount of data that the fragment will return
- ▶ Communication bandwidth

The query optimizer generates local and remote access plans for processing a query fragment, based on resource cost. DB2 then chooses the plan it believes will process the query with the least resource cost.

If any of the fragments are to be processed by data sources, DB2 submits these fragments to the data sources. After the data sources have processed the fragments, the results are retrieved and returned to DB2. If DB2 performed any part of the processing, it combines its results with the results retrieved from the data source. DB2 then returns all results to the client.

Compensation

The DB2 federated server does not push down a query fragment if the data source cannot process it, or if the federated server can process it faster than the data source can process it. For example, suppose that the SQL dialect of a data source does not support a CUBE grouping in the GROUP BY clause. A query that contains a CUBE grouping and references a table in that data source is submitted to the federated server. DB2 does not push down the CUBE grouping to the data source, but processes the CUBE itself. The ability by DB2 to process SQL that is not supported by a data source is called *compensation*.

The federated server compensates for lack of functionality at the data source in two ways:

- ▶ It can ask the data source to use one or more operations that are equivalent to the DB2 function stated in the query. Suppose a data source does not support the cotangent (COT(x)) function, but supports the tangent (TAN(x)) function. DB2 can ask the data source to perform the calculation (1/TAN(x)), which is equivalent to the cotangent (COT(x)) function.
- ▶ It can return the set of data to the federated server, and perform the function locally.

With compensation, the federated server can support the full DB2 SQL dialect for queries against data sources. Even data sources with weak SQL support or no SQL support will benefit from compensation. You must use the DB2 SQL dialect with a federated system, except in a pass-through session.

Pass-through sessions

You can submit SQL statements directly to data sources by using a special mode called *pass-through*. You submit SQL statements in the SQL dialect used by the data source. Use a pass-through session when you want to perform an operation that is not possible with the DB2 SQL/API. For example, use a pass-through session to create a procedure, create an index, or perform queries in the native dialect of the data source.

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

4.1.2 Components and packaging

DB2 Information Integrator is available for IBM AIX, Linux for Intel, Microsoft Windows, Hewlett Packard HP-UX, and Sun Solaris operating environments.

The DB2 Information Integrator product is a set of the following products:

- ▶ DB2 UDB Enterprise Server Edition V8.1
- ▶ Relational and nonrelational wrappers, formerly available as DB2 Relational Connect and DB2 Life Sciences Data Connect
- ▶ DB2 Net Search Extender

Information Integrator editions

DB2 Information Integrator is available in the following editions:

- ▶ Replication

- ▶ Standard
- ▶ Advanced and Developer

For more details, please refer to the packaging information in Table 4-2.

Table 4-2 DB2 Information Integrator packaging

DB2 Information Integrator components	Replication Edition	Standard Edition	Advanced Edition	Developer Edition
DB2 ESE	X	X	X	X
Relational wrappers	X	X	X	X
Nonrelational wrappers		X	X	X
Net Search Extender		X	X	X

4.2 Installation and configuration

The installation of DB2 Information Integrator we discuss here is an example using Microsoft Windows 2000 Server, with DB2 UDB ESE V8.1 FP2 already installed. For installation on different operating systems and different versions of DB2 UDB ESE please refer to the Installation Guide for DB2 Information Integrator.

4.2.1 Pre-installation requirements

The pre-installation requirements are:

- ▶ Hardware
 - Memory and disk space sufficient to fulfill DB2 minimum requirements
 - Approximately 30 MB of free disk space for the relational and nonrelational wrappers
- ▶ Software
 - Windows 2000 Server or Workstation (SP3)
 - DB2 UDB ESE V8.1 FP2 installed
 - DBMS client software for the data sources installed and properly configured

4.2.2 Installation instructions

In this section we show how to install the DB2 Information Integrator nonrelational wrappers for DB2, Informix, Oracle, and ODBC data sources.

1. Log in as the Administrator user if your Windows system is configured for multiple user accounts.
2. If you install from CD, it normally autostarts the setup program. If not, run the setup program manually by selecting **Start -> Run** and then entering the command:

```
cd cdriveletter:iissetup
```

If you install from another location, locate the directory where the setup files reside and start the setup program from there by selecting **Start -> Run** and then entering the command:

```
<dirpath>\iissetup
```

The setup program shows you the Installation Launchpad window (Figure 4-2).

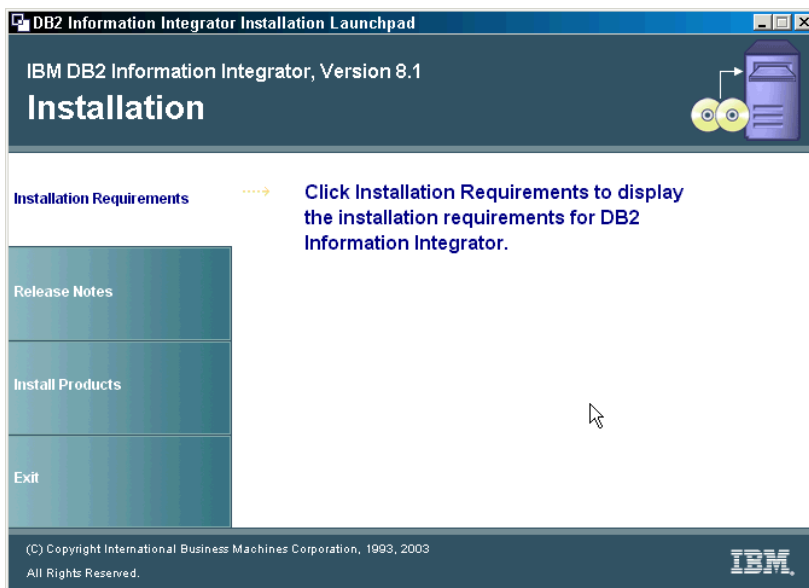


Figure 4-2 Installation Launchpad window

Click **Install Products**. InstallShield shows you the license agreement for DB2 Information Integrator Standard Edition, as shown in Figure 4-3 on page 84.

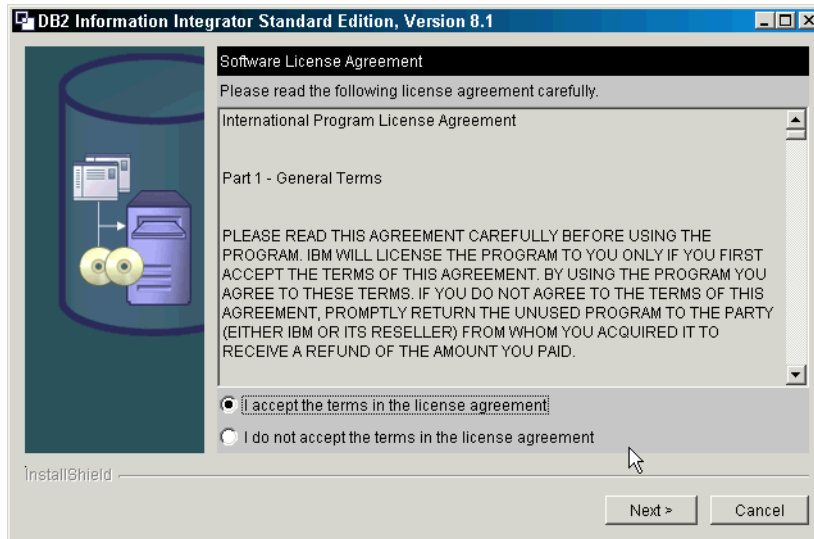


Figure 4-3 License agreement

You must click the radio button to accept the license agreement and then click **Next**. InstallShield then asks you to select the type of wrappers you want to install, as in Figure 4-4.

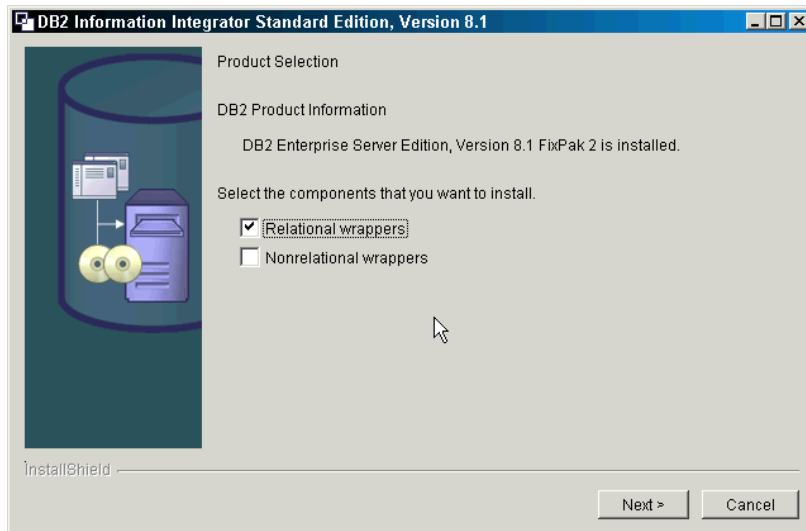


Figure 4-4 Wrapper selection

Select relational wrappers and click **Next**. In the following window (Figure 4-5 on page 85) check to see if the installer location is correct, then click **Next**.

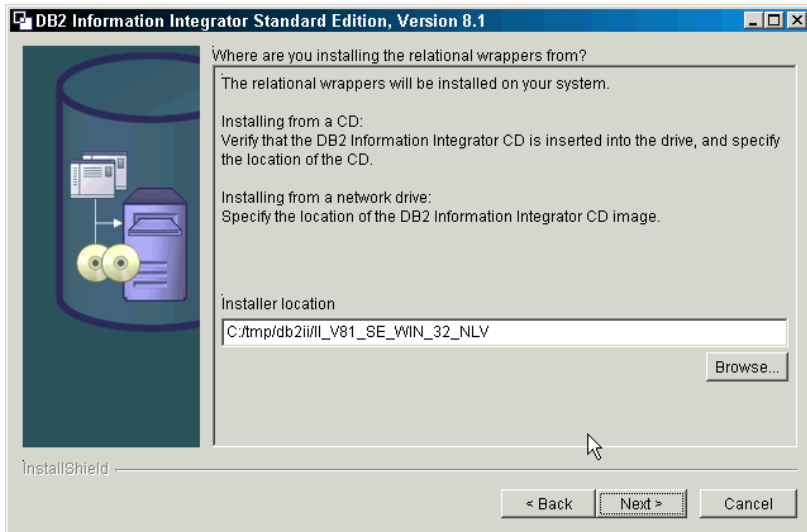


Figure 4-5 Installer location

Next you will see Figure 4-6. There you will click **Install Product** to start the installation process.



Figure 4-6 Relational wrapper installer welcome window

On the next two screens (Figure 4-7 and Figure 4-8) accept the defaults and click **Next**.

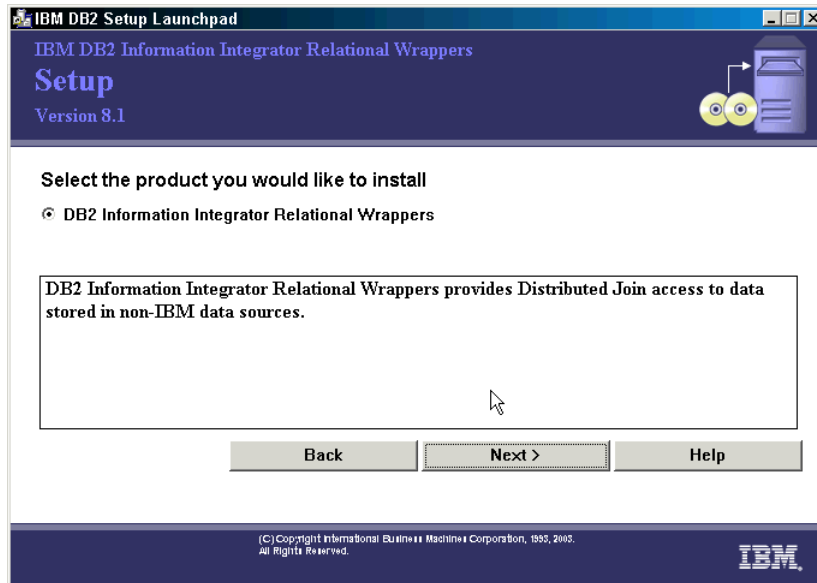


Figure 4-7 Relational wrapper selection window

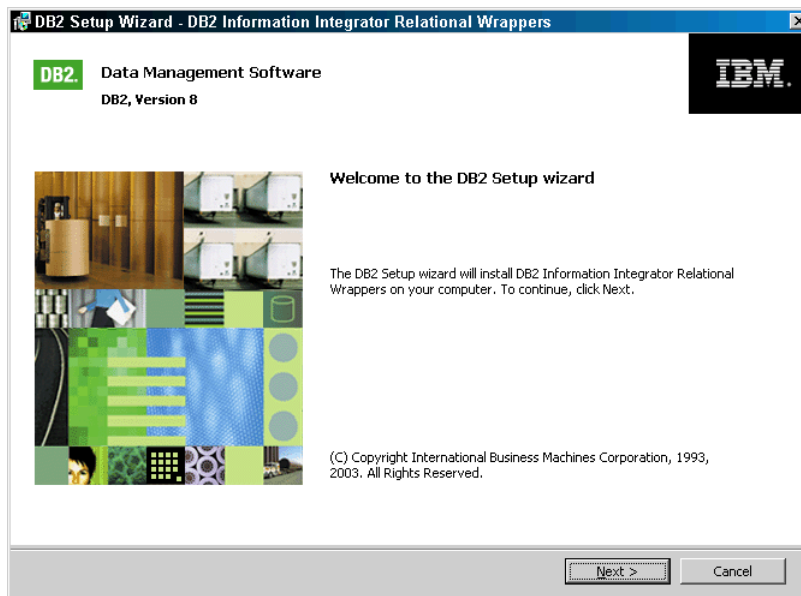


Figure 4-8 InstallShield Wizard for relational wrappers

InstallShield then prompts you again to accept the license agreement, as shown in Figure 4-9.

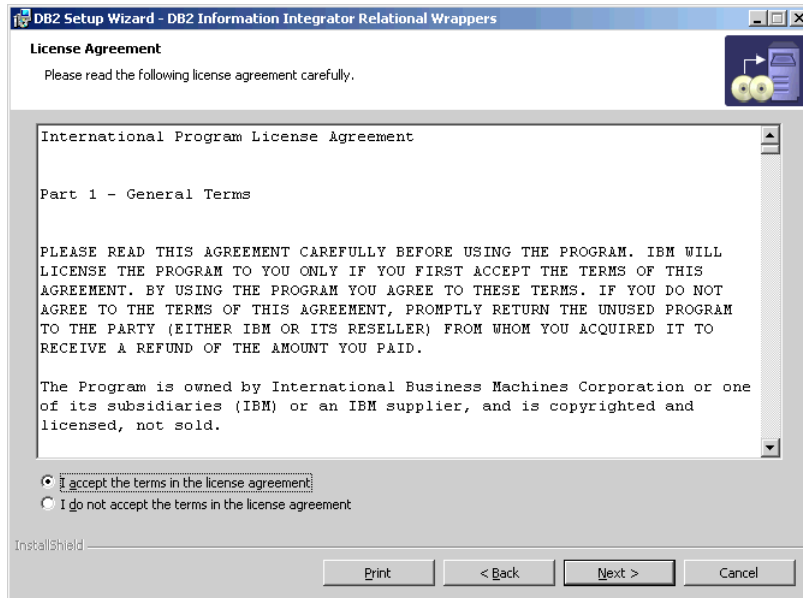


Figure 4-9 License agreement

The default button indicates that you will accept the license agreement. Click **Next** to accept the license agreement. Again you have the choice to install the wrappers and/or to create a response file for later installation (shown in Figure 4-10 on page 88).

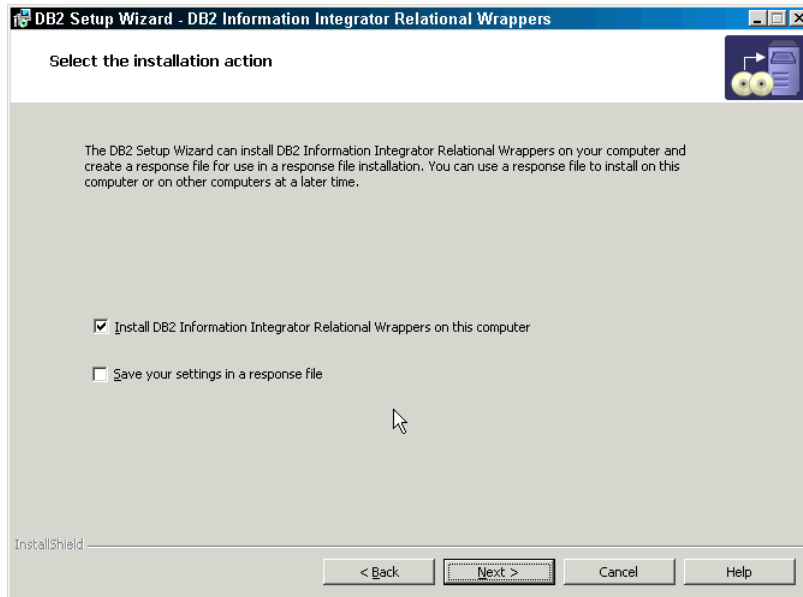


Figure 4-10 Select action window

Make your selection to install the wrappers and click **Next**. The following window (Figure 4-11 on page 89) allows you to select the wrappers you want to use in your federated server.

Select the different wrappers you want use in your federated system and click **Next**.

Note: The selection in Figure 4-11 shows all the relational wrappers except the wrapper for Informix data sources. This wrapper is included in DB2 UDB ESE V8.1 and must be installed from the DB2 CD (see “Installing the Informix wrapper” on page 90).

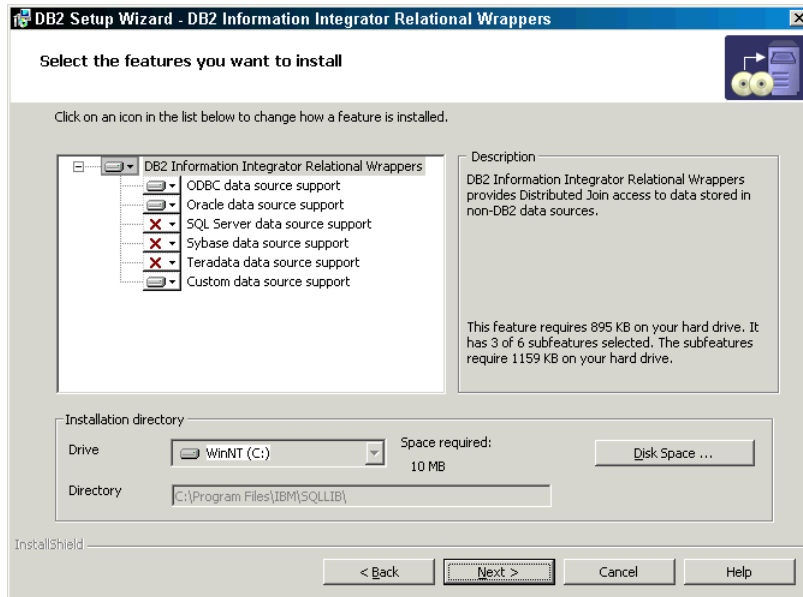


Figure 4-11 Feature selection window

In the next window (Figure 4-12) select the languages you want to install for the graphical tools, user interface, and product messages. Click **Next** to continue.

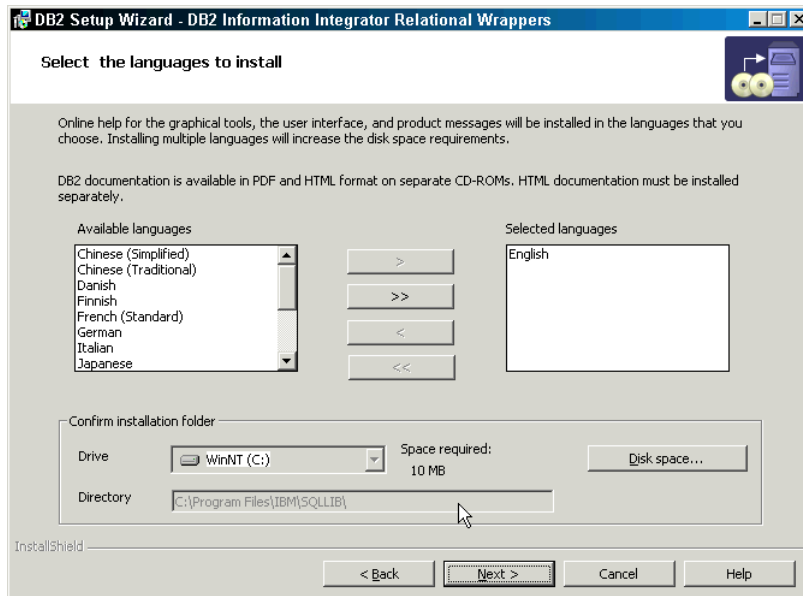


Figure 4-12 Language selection window

The next window is Figure 4-13, which shows you the selected components. Click **Install**, and the installation process will begin.

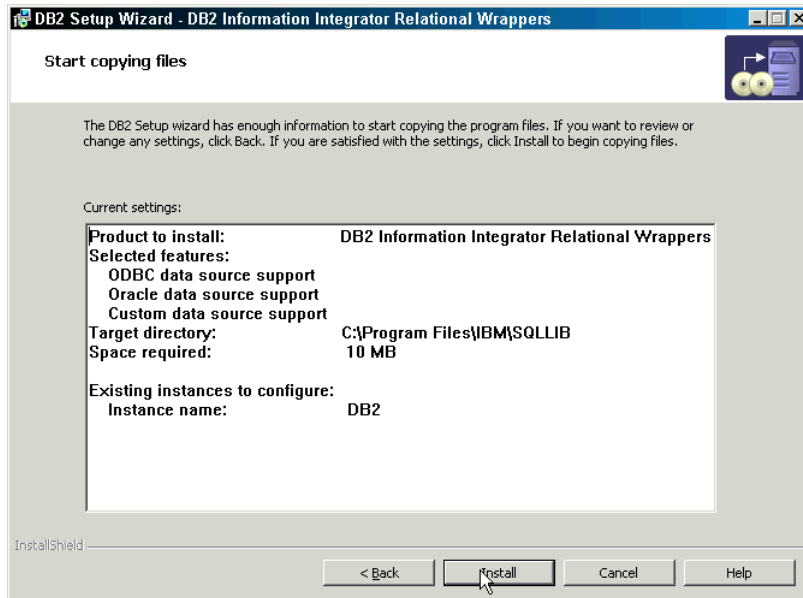


Figure 4-13 Selected product/features

After the installation finishes, there will be more screens that will simply guide you through the completion of the process. Just follow the instructions to finish the installation process.

Installing the Informix wrapper

As previously mentioned, the Informix wrapper is not part of the DB2 Information Integrator product. This wrapper is already included in all DB2 UDB editions starting with version 8. To install this wrapper, step through the following instructions:

1. Stop DB2.
2. Go to the Control Panel, as shown in Figure 4-14 on page 91, then double-click **Add/Remove Programs**.

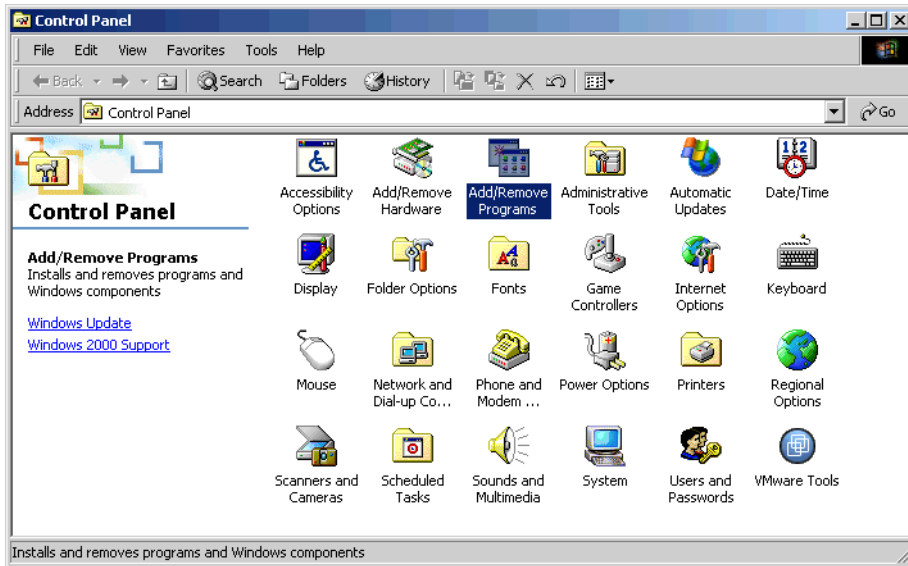


Figure 4-14 Windows 2000 - Control panel

3. In the next panel (Figure 4-15) select **DB2 Enterprise Server Edition** and click **Change**.

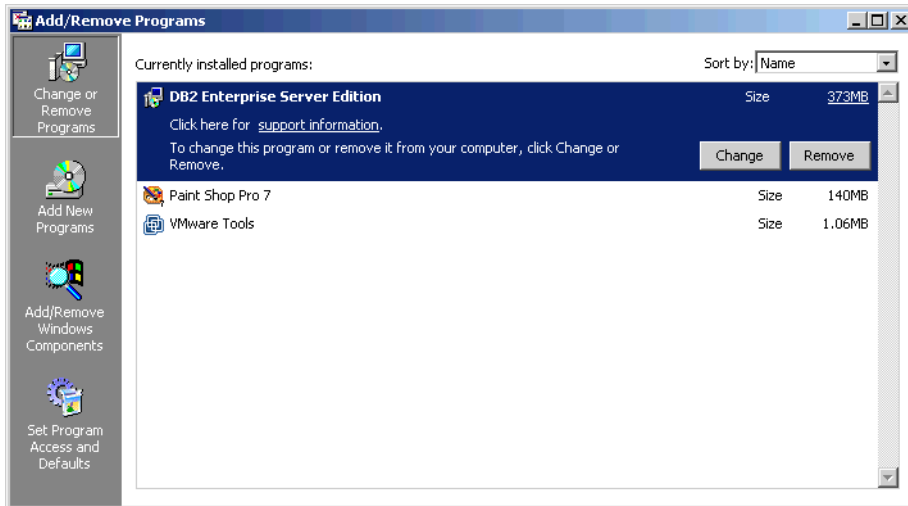


Figure 4-15 Add/Remove Programs panel

4. Then click **Next** to start the installation process, as shown in Figure 4-16 on page 92.

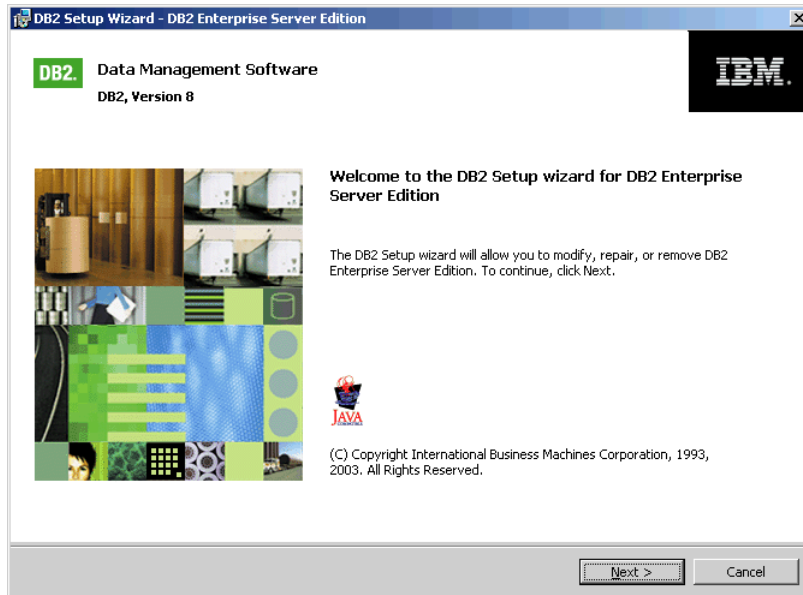


Figure 4-16 DB2 Setup Wizard

5. In the next window (Figure 4-17) select the **Modify** radio button and click **Next**.

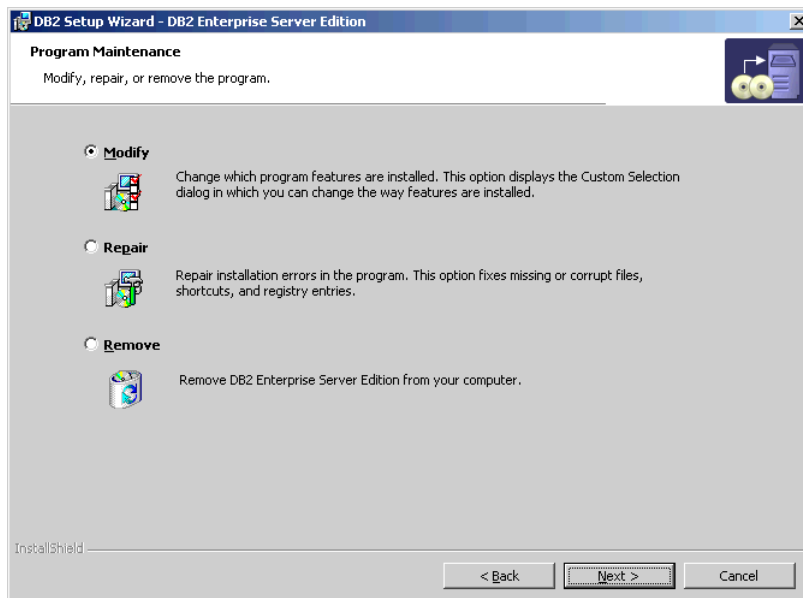


Figure 4-17 Program Maintenance window

6. In the feature selection window (shown in Figure 4-18) you can change the components you want to add or remove.

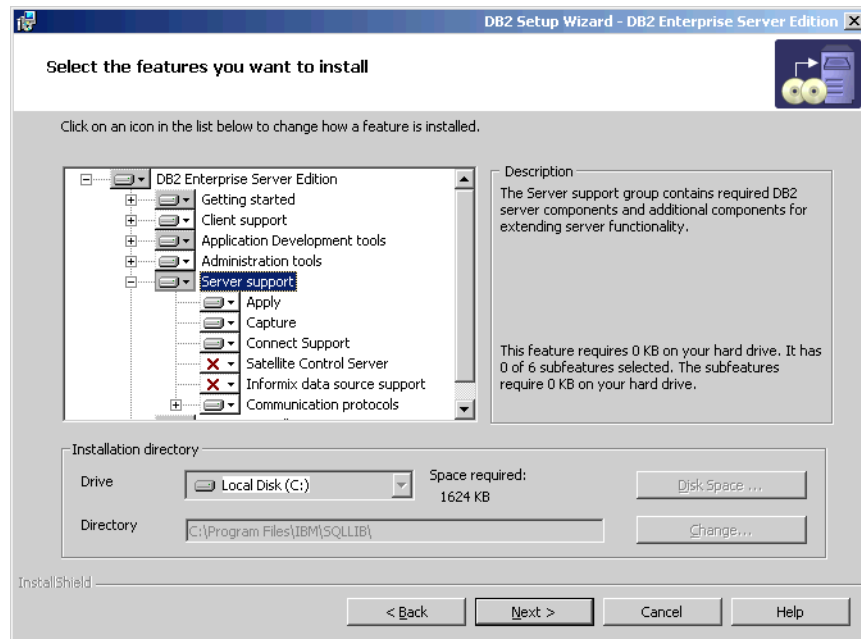


Figure 4-18 Feature selection window

7. The Informix wrapper is one of the options in the server support group. Select the Informix data source support to be installed and continue the installation process. There are no special instructions needed here. Simply follow the instructions until the installation is completed.

Note: Running setup.exe from your DB2 distribution CD will not allow you to change the installed components. It only re-installs the previous selected components.

4.2.3 Post-installation tasks

After the federated server is setup, you can avoid potential problems by checking the key settings discussed in the following sections.

Check the data source environment variables

When you set up the federated server, the installation process attempts to set the environment variables for the Documentum, Informix, Oracle, Microsoft SQL Server, Sybase, and Teradata sources.

- ▶ On UNIX you need to check the environment variables for Microsoft SQL Server, Sybase, and Teradata data sources.
- ▶ For Oracle, Informix, and Documentum data sources, you need to check the environment variables on both UNIX and Windows operating systems.

Verify that the environment variables for the data sources you want to access are set in the `sqllib\cfg\db2dj.ini` file. The `sqllib\cfg\` directory is located in the directory where you installed DB2.

The valid environment variables for each data source are listed in Table 4-3.

Table 4-3 Valid environment variables

Data source	Valid environment variables
Informix	INFORMIXDIR INFORMIXSERVER INFORMIXSQLHOSTS (optional) CLIENT_LOCALE (optional) DB_LOCALE (optional) DBNLS (optional)
Oracle	ORACLE_HOME ORACLE_BASE ORA_NLS (optional) TNS_ADMIN (optional)
Microsoft SQL Server	DJXODBCTRACE DJX_ODBC_LIBRARY_PATH ODBCINI DB2LIBPATH DB2ENVLIST
Sybase	SYBASE SYBASE_OCS
Teradata	COPERR COPLIB

Important: Only one valid INFORMIXSERVER (defined through `setnet32`) and INFORMIXDIR value needs to be specified in the `db2dj.ini` file. Any other INFORMIXSERVER defined through `setnet32` will also be accessible through the federated server.

The data source environment variables will not be set in the `sqllib\cfg\db2dj.ini` file if you:

- ▶ Install the data source client software after the DB2 federated server is set up.
- ▶ Have not installed the data source client software.

Note: If you change any settings in the `db2dj.ini` file, *recycle* the DB2 instance to ensure that your changes are reflected.

Example 4-1 shows a `db2dj.ini` for use with Informix and Oracle data sources.

Example 4-1 db2dj.ini-file

```
ORACLE_HOME=d:\oracle\ora92
INFORMIXSERVER=ifx_winsvr_tcp
INFORMIXDIR=c:\program files\informix
```

Check the FEDERATED parameter

The FEDERATED parameter must be set to YES to enable access to the data sources. It is possible that this parameter was set when you created the DB2 instance. However, it is important to make certain that the FEDERATED parameter is set to YES.

To check/change the parameter with DB2 Control Center:

1. Start the DB2 Control Center.
2. Right click the federated server instance and click **Configure Parameters**.
3. Scroll to Environment; look for FEDERATED.
4. Set the parameter to YES, or see whether it is already in pending state to YES.
5. Recycle DB2:

```
db2stop
db2start
```

Figure 4-19 on page 96 and Figure 4-20 on page 96 show the configuration with the DB2 Control Center.

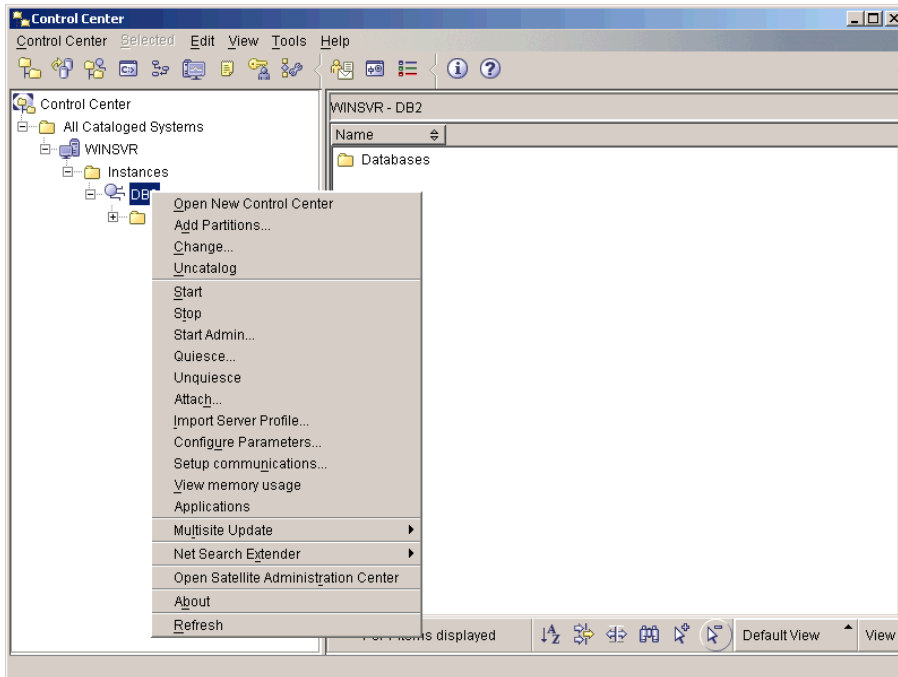


Figure 4-19 Using Control Center

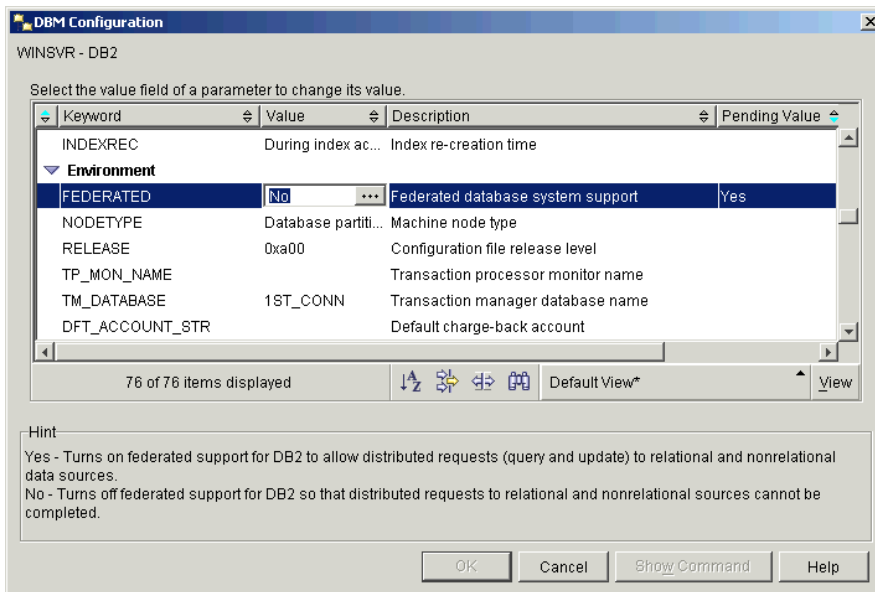


Figure 4-20 Database manager configuration window

Note: The installation process checks the FEDERATED parameter and sets the value to YES. If the change was made through the installation process, recycle the DB2 instance to ensure that the new value takes affect.

To check/change the parameter with a DB2 command window:

1. Open a DB2 command window.
2. Issue the command:
`DB2 GET DATABASE MANAGER CONFIGURATION`
3. Check the value for the parameter FEDERATED.
4. If it is set to NO issue the command:
`DB2 UPDATE DATABASE MANAGER CONFIGURATION USING FEDERATED YES`
5. Recycle DB2:
`db2stop`
`db2start`

For more information, please see the Installation Guide for DB2 Information Integrator.

4.3 Creating and using wrappers and nicknames

After you install the wrappers, you need to set up a federated database. You can either create a new federated database or use an existing database in your DB2 instance. The following steps are necessary to get access to a remote data source:

1. Create a wrapper.
2. Define a server.
3. Define user mapping (recommended).
4. Create and authorize nicknames.

All of the steps mentioned above can be performed either through the DB2 Control Center or, for example, through SQL in a DB2 command-line processor window. We give instructions for performing these tasks in the sections that follow.

Note: Step 3 above may not be necessary under certain circumstances when the authorization ID and password are identical on your federated server and the data source you want to access. If you intend to use these methods the password has to be provided when you connect to the federated server.

4.3.1 Create a wrapper

Usually you create a wrapper for each type of data source that you want to access. Through this wrapper you can access tables in different databases and data sources of the same type. We create a wrapper in our federated database *stores* for Informix data sources and name it INFORMIX.

Here are the steps through the DB2 Control Center:

- ▶ Right click **Federated Database Objects** for database stores and click **Create Wrapper**, as shown in Figure 4-21.
- ▶ Choose data source type and enter a name for the wrapper (as depicted in Figure 4-22 on page 99), and click **OK**.

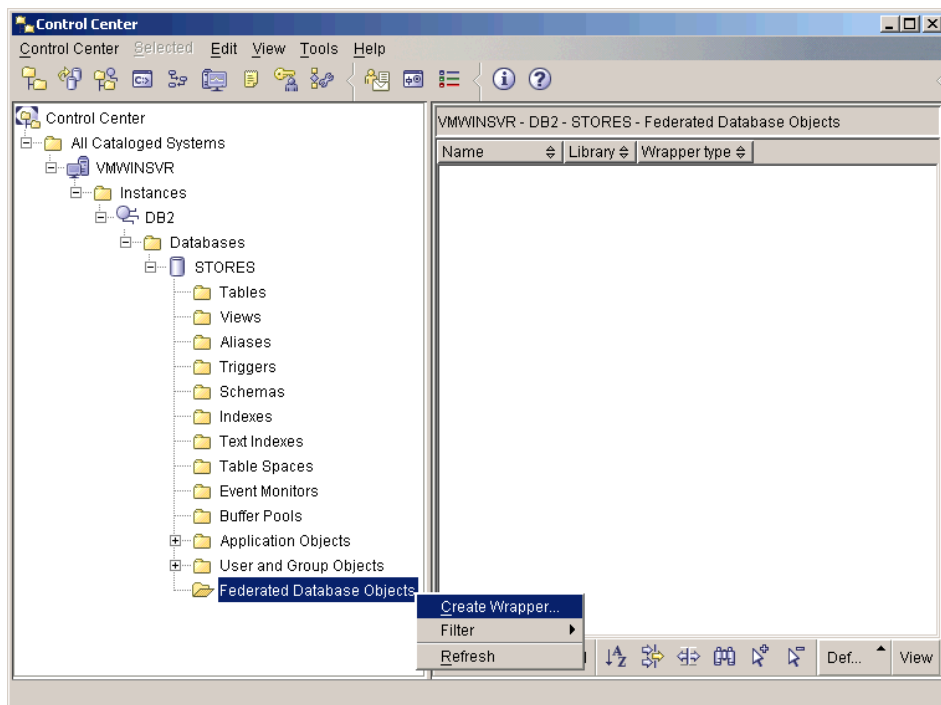


Figure 4-21 Create a wrapper in DB2 Control Center

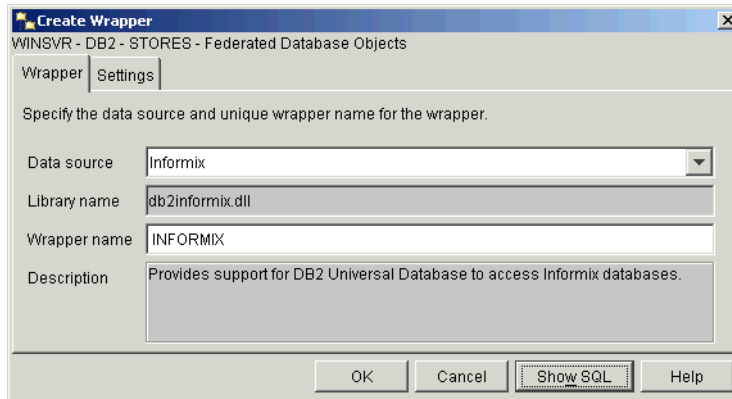


Figure 4-22 Choose data source type and wrapper name

Example 4-2 shows the SQL command that creates the same wrapper. This can be seen by clicking the **Show SQL** button.

Example 4-2 Create wrapper for Informix data sources -SQL

```
CONNECT TO stores;
CREATE WRAPPER "INFORMIX" LIBRARY 'db2informix.dll';
```

4.3.2 Define a server

Now we have to define a specific server where the tables you want to access reside. We want to get access to `ifx_winsvr_tcp` (INFORMIXSERVER value) with a database called stores. Therefore we define a server called IFXWIN.

Here are the steps with DB2 Control Center:

1. Go to the previously defined wrapper INFORMIX, right-click **Servers**, and click **Create**, as depicted in Figure 4-23 on page 100.
2. Type in IFXWIN as the name of the server and select as version 9, then go to the Settings tab and type in `ifx_winsvr_tcp` as the NODE and stores as DBNAME and continue by clicking **OK**. These steps are depicted in Figure 4-24 on page 100 and Figure 4-25 on page 101.

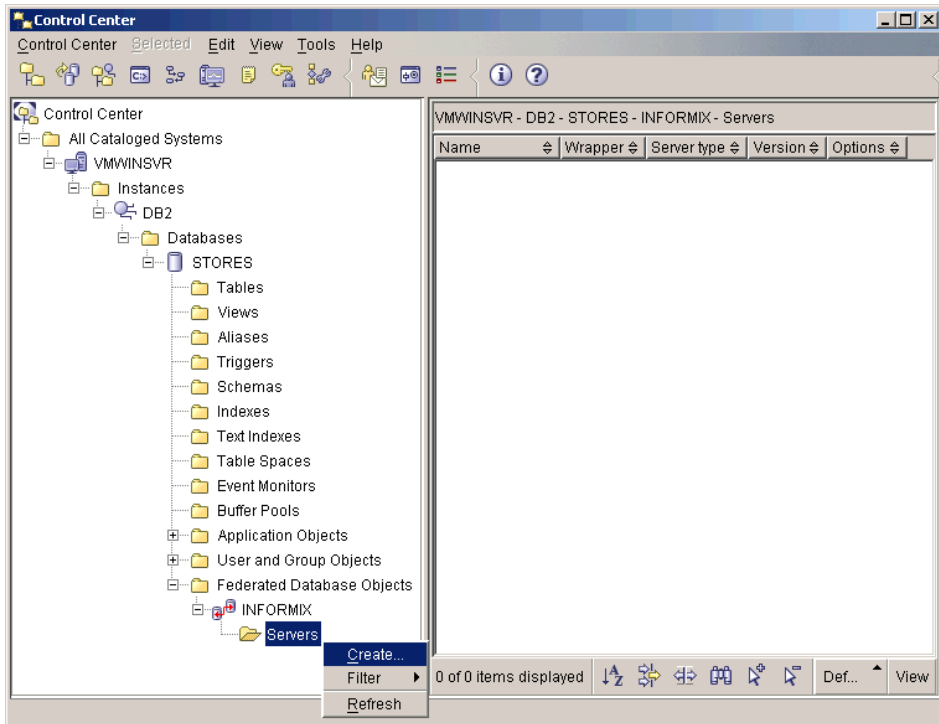


Figure 4-23 Create server

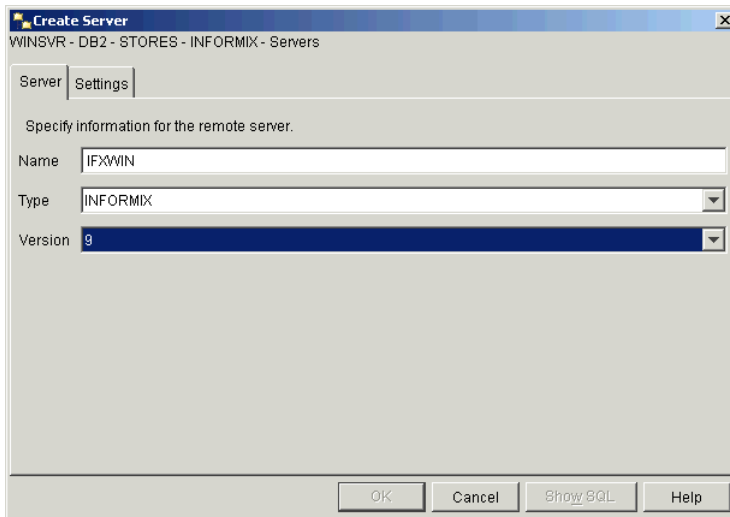


Figure 4-24 Choose server name and version

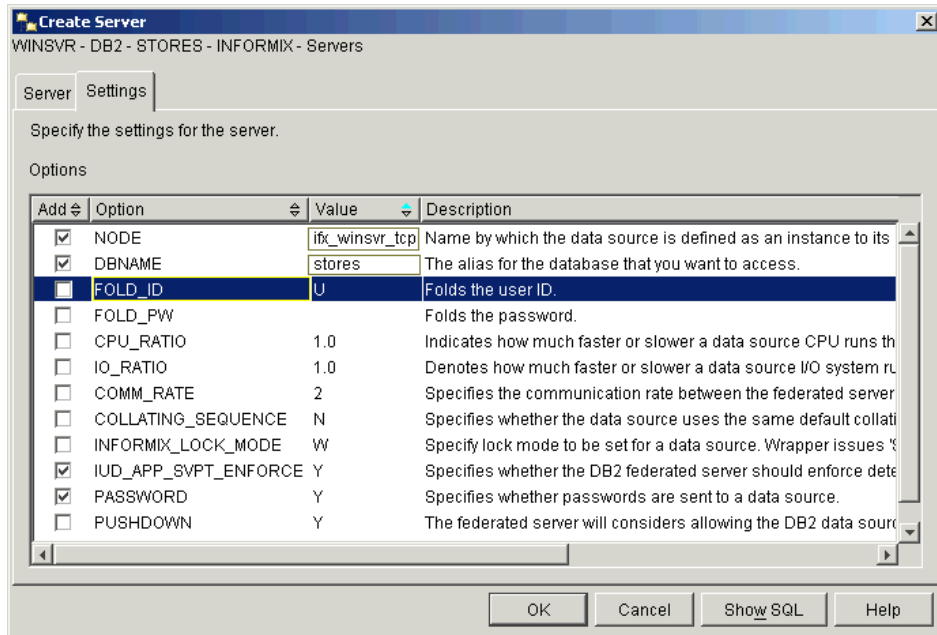


Figure 4-25 Server options in the Settings tab

In the Settings tab you find various options for the server. You can define the options here and alter them later. These settings can also be overridden for a specific session at the application level. Some of these options will be discussed in Chapter 7, “Optimization in a federated environment” on page 193.

Important: A server can only be defined after all the red boxes in the Settings tab have been set.

The SQL commands shown in Example 4-3 represent exactly the same server definition, including the options, as selected in Figure 4-25.

Example 4-3 Define a server - SQL

```
CONNECT to stores;
CREATE SERVER IFXWIN TYPE INFORMIX VERSION '9' WRAPPER "INFORMIX" OPTIONS(ADD
NODE 'ifx_winsvr_tcp', DBNAME 'stores', PASSWORD 'Y', IUD_APP_SVPT_ENFORCE
'Y');
```

4.3.3 Define user mapping

A federated system user is only connected to a federated server. The user name and password are specific to the federated DB2 server instance. A user who needs access to a different data source also has to be authenticated as a valid user at that data source. The federated server allows you to create a user mapping to a data source. For each federated user, you can create a user mapping to a particular server.

Assume there is a user in our federated server db2usrw. This user wants to connect to our Informix source IFXWIN. At the source, a user ifxusrw exists. Defining the user mapping with the DB2 Control Center requires the following steps:

1. Go to the previously defined server IFXWIN, right-click **User mappings** (below the server), and click **Create**, as shown in Figure 4-26.

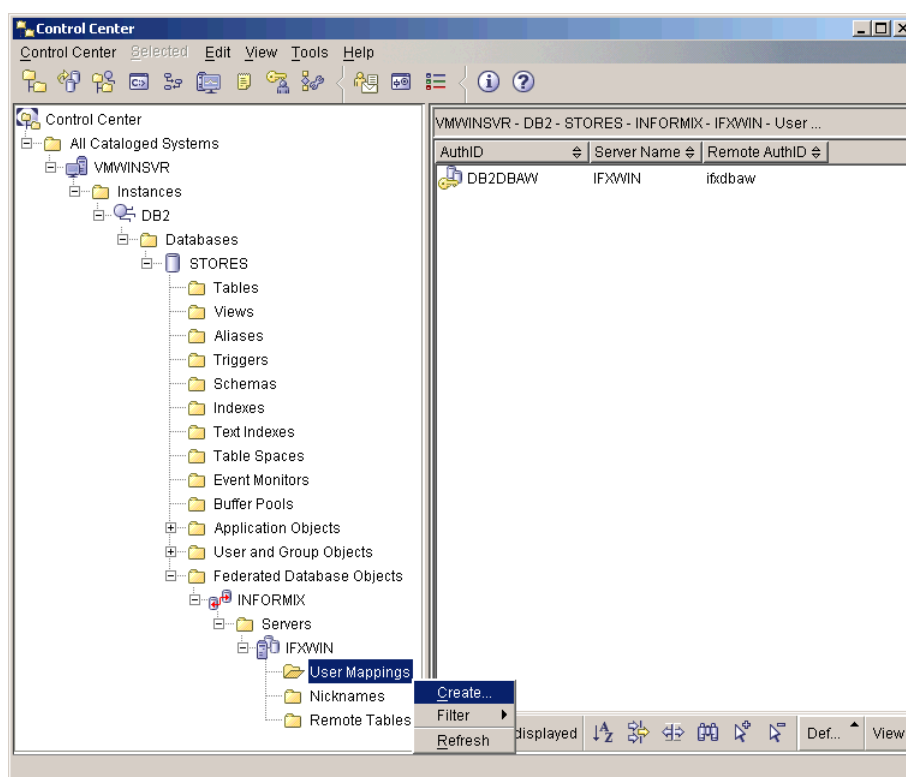


Figure 4-26 Create user mappings

2. Select the user ID **db2usrw** as shown in Figure 4-27 on page 103.

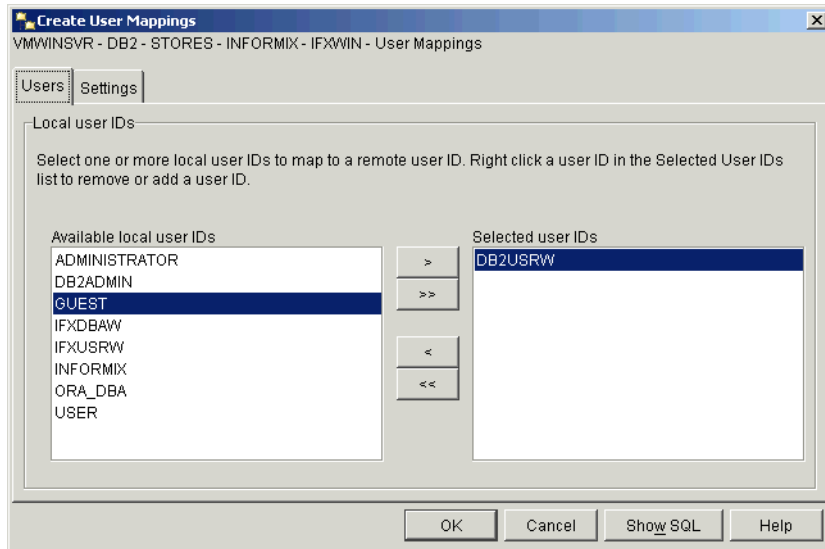


Figure 4-27 Select local user IDs

- Now go to the Settings tab, shown in Figure 4-28. There specify the remote user ID ifxusrw and the remote password password (password is hidden), and click **OK**.

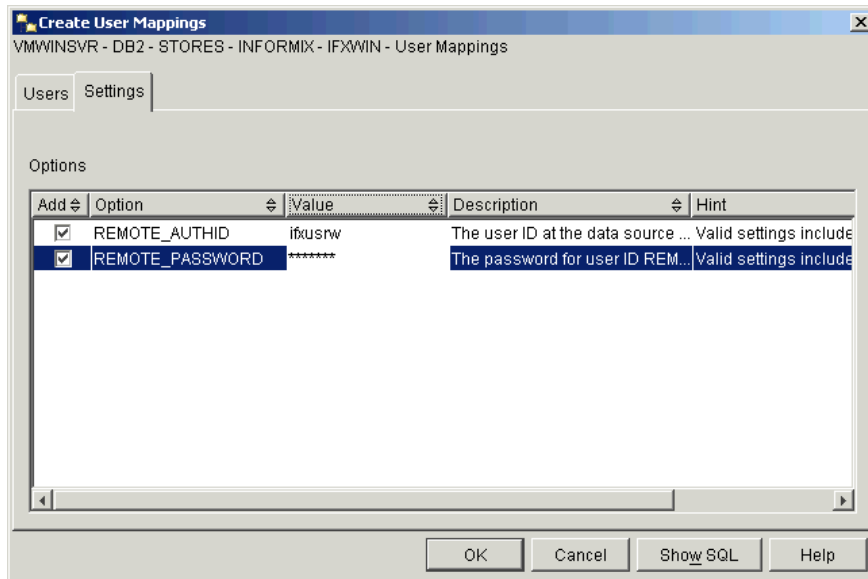


Figure 4-28 Specify remote user ID and remote password

Tip: You can change the user mapping at any time without dropping the mapping for a specific user.

Example 4-4 shows the equivalent SQL commands for creating the same user mapping.

Example 4-4 Create user mapping - SQL

```
CONNECT to stores;  
CREATE USER MAPPING FOR "DB2USRW" SERVER "IFXWIN" OPTIONS (ADD REMOTE_AUTHID  
'ifxusrw', ADD REMOTE_PASSWORD 'password');
```

Note: User mapping with DB2 Information Integrator is very powerful. You can define 1:1 or many:1 mappings from the federated server to a data source.

4.3.4 Create and authorize nicknames

A nickname in the federated database represents a table in a data source. It is an entry in the federated system catalog (not the same as aliases) that maps the remote table and data types to a local name and DB2 data types. For all the different data sources a default data type mapping exists that is provided through the wrapper. You are allowed to modify the default mapping or only modify the audiotape mapping for a specific nickname.

We want to access the customer table in the stores database, so we create a nickname called ifxw_customer. Why do we use the prefix ifxw_? The prefix tells us in which source the table is located. In this case it is the Informix server on Windows. Secondly, a nickname in a federated database has to be unique, so this is how we fulfill that requirement.

Here again are the steps to do in the DB2 Control Center:

1. Go again to the server IFXWIN as shown in Figure 4-29 on page 105, then right-click **Nicknames** and click **Create**.
2. Add a nickname or discover the remote schema, as depicted in Figure 4-30 on page 105.
3. Enter the required fields using the screen shown in Figure 4-31 on page 106, and click **OK**.
4. Click **OK** again, as shown in Figure 4-32 on page 106, and the nickname will be created.

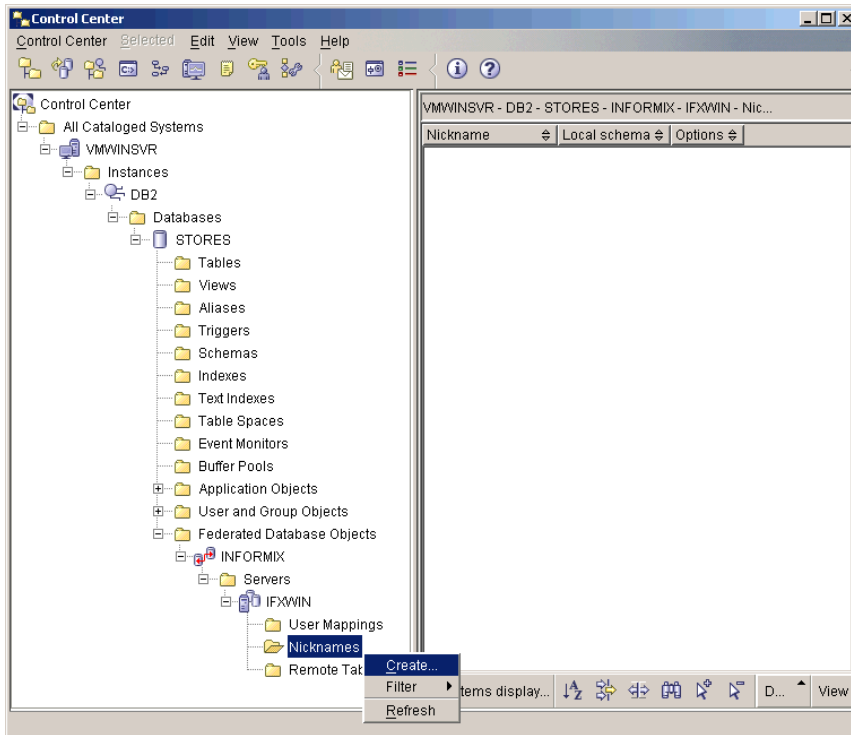


Figure 4-29 Create nickname

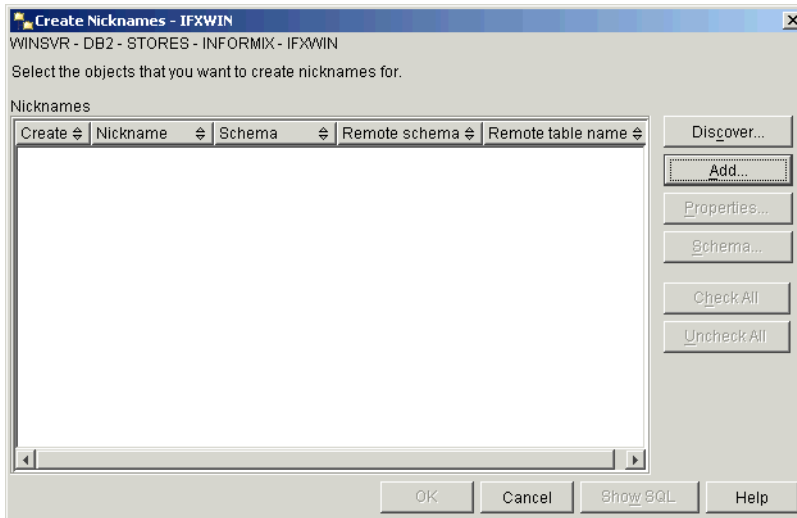


Figure 4-30 Add nickname or discover remote schema

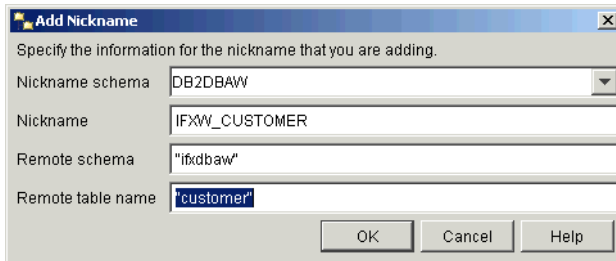


Figure 4-31 Add nickname IFXW_CUSTOMER

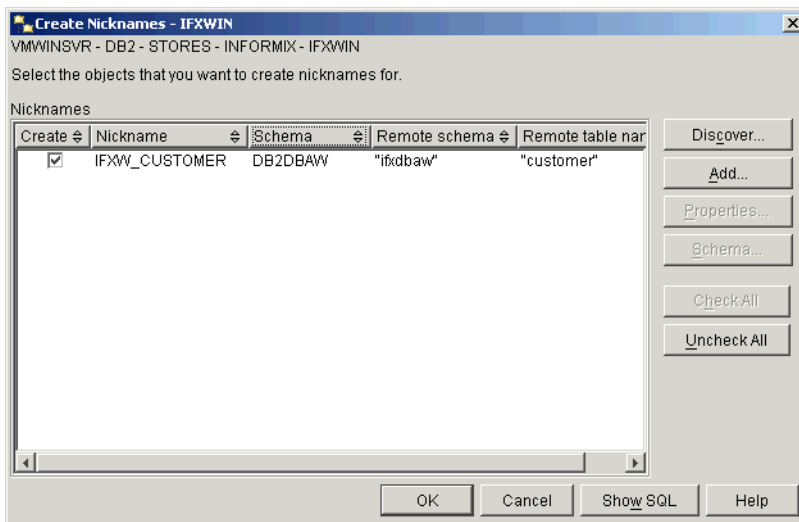


Figure 4-32 Execute add nickname

Note: Depending on the remote source, the schema name and the remote table name are required to be case-sensitive.

Of course, the nickname creation can be done through the SQL statements that we show in Example 4-5.

Example 4-5 Create nickname - SQL

```
CONNECT TO stores;
CREATE NICKNAME DB2DBAW.IFXW_CUSTOMER FOR IFXWIN."ifxdbaw"."customer";
```

Important: You will also have to authorize access to nicknames in the same way as you have to authorize access to local tables.

All these steps must be done for all of the remote tables you want to access.

The definitions can be created, altered, or dropped. If you drop an object that has dependent objects, all the dependent objects will also be dropped. Note that this may also affect other objects.

4.3.5 Testing the connection to a data source

After you have created all the necessary objects for a connection to a remote table, you can test the connection simply by using DB2 Control Center. Right-click the nickname you want to access and select **Sample Contents**. This is depicted in Figure 4-33. If the connection is successful you should see a window with a set of rows from the remote table, as shown in Figure 4-34 on page 108.

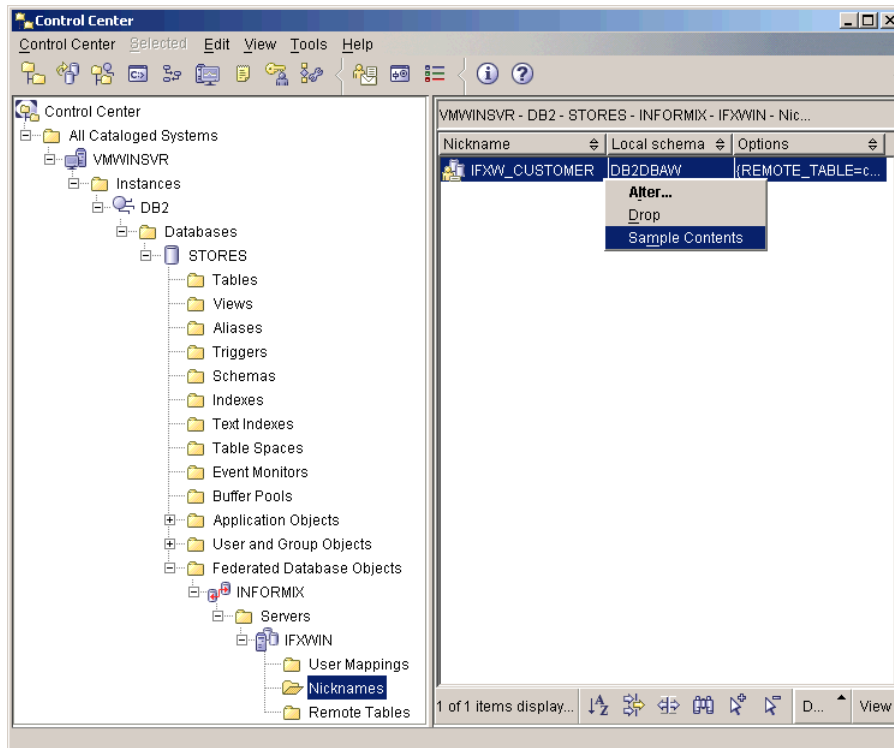


Figure 4-33 Show sample content

CUSTOME...	FNAME	LNAME	COMPANY	ADDRESS1	ADDRESS2	CITY	STATE	ZIPCODE	PHONE
101	Ludwig	Pauli	All Sports S...	213 Erstwil...		Sunnyvale ...	CA	94086	408-789-8...
102	Carole	Sadler	Sports Spo...	785 Geary ...		San Franci...	CA	94117	415-822-1...
103	Phillip	Currie	Phil's Sport...	654 Poplar ...	P. O. Box 3...	Palo Alto	CA	94303	415-328-4...
104	Anthony	Higgins	Play Ball!	East Shop...	422 Bay Ro...	Redwood ...	CA	94026	415-368-1...
105	Raymond	Vector	Los Altos S...	1899 La Lo...		Los Altos	CA	94022	415-776-3...
106	George	Watson	Watson & ...	1143 Carve...		Mountain Vi...	CA	94063	415-389-8...
107	Charles	Ream	Athletic Su...	41 Jordan ...		Palo Alto	CA	94304	415-356-9...
108	Donald	Quinn	Quinn's Sp...	587 Alvara...		Redwood ...	CA	94063	415-544-8...
109	Jane	Miller	Sport Stuff	Mayfair Mar...	7345 Ross...	Sunnyvale ...	CA	94086	408-723-8...
110	Roy	Jaeger	AA Athletics	520 Topaz ...		Redwood ...	CA	94062	415-743-3...
111	Frances	Keyes	Sports Cen...	3199 Sterli...		Sunnyvale ...	CA	94085	408-277-7...
112	Margaret	Lawson	Runners & ...	234 Wyand...		Los Altos	CA	94022	415-887-7...
113	Lana	Beatty	Sportstown...	654 Oak Gr...		Menlo Park...	CA	94025	415-356-9...
114	Frank	Albertson	Sporting Pl...	947 Waverl...		Redwood ...	CA	94062	415-886-6...
115	Alfred	Grant	Gold Medal...	776 Gary A...		Menlo Park...	CA	94025	415-356-1...
116	Jean	Parmelee	Olympic Cit...	1104 Spino...		Mountain Vi...	CA	94040	415-534-8...
117	Arnold	Sipes	Kids Korme...	850 Lytton ...		Redwood ...	CA	94063	415-245-4...
118	Dick	Baxter	Blue Ribbo...	5427 Colle...		Oakland	CA	94609	415-655-0...
119	Bob	Shorter	The Triathl...	2405 Kings...		Cherry Hill	NJ	08002	609-663-6...
120	Fred	Jewell	Century Pr...	6627 N. 17L...		Phoenix	AZ	85016	602-265-8...

Figure 4-34 Sample data from a remote table

4.4 Considerations for use

You are now in a position to begin using the federated system. However, there are a number of things to consider and address before you do so. These are things that can impact performance, and perhaps even data integrity,

4.4.1 Statistics and index specifications

One of the most important factors determining the federated query performance is the query execution plan produced at the federated server. Among other things, the optimizer evaluates the best query plan based on statistics and index information for the nicknames. It then decides, for example, if a join should be executed at the data source or the federated server.

When a nickname is created, DB2 Information Integrator automatically polls information on table statistics and indexes from the data source and populates the catalog information for the nickname. That is, it does not generate the statistics on the data source. Instead, it queries the data source and retrieves the information. Index and statistics changes are *not automatically propagated* to the federated server. If there is a change in the volume of data at the source or new indexes are added or removed, you must update the statistics in the federated server to reflect the changes so that the optimizer can make its decisions based on more accurate information.

However, the **runstats** utility is not supported on nicknames. So one approach you may take is to use the **getstats** utilities that can collect these statistics for you and update the information on the nicknames in the catalog. There are two utilities that have been made available for download at:

<http://www.ibm.com/software/data/integration/db2ii/support.html>

They are:

- ▶ **get_stats** for relational and some nonrelational nicknames, including DB2 UDB, Oracle, Sybase, IBM Informix, Microsoft SQL Server, Teradata, ODBC, table-structured files, Excel, XML, and BioRs
- ▶ **get_stats_nr** for nonrelational nicknames including flat files, Excel, BioRs, Documentum, Blast, Hmmer, Extended Search, Entrez, and XML

See Example 4-6 for an example of using the **get_stats** utility.

Example 4-6 get_stats syntax

To calculate statistics of all nicknames under a schema:

```
get_stats { <user_id> <password> } <dbname> <schema>
```

To calculate statistics of a single nickname:

```
get_stats { <user_id> <password> } <dbname> <schema>.<nickname>
```

Consider the nickname `ifxl_customer`. In Figure 4-35 on page 110 we show the statistics on `SYSSTAT.COLUMNS` for `ifx_items`, before and after **get_stats** execution.

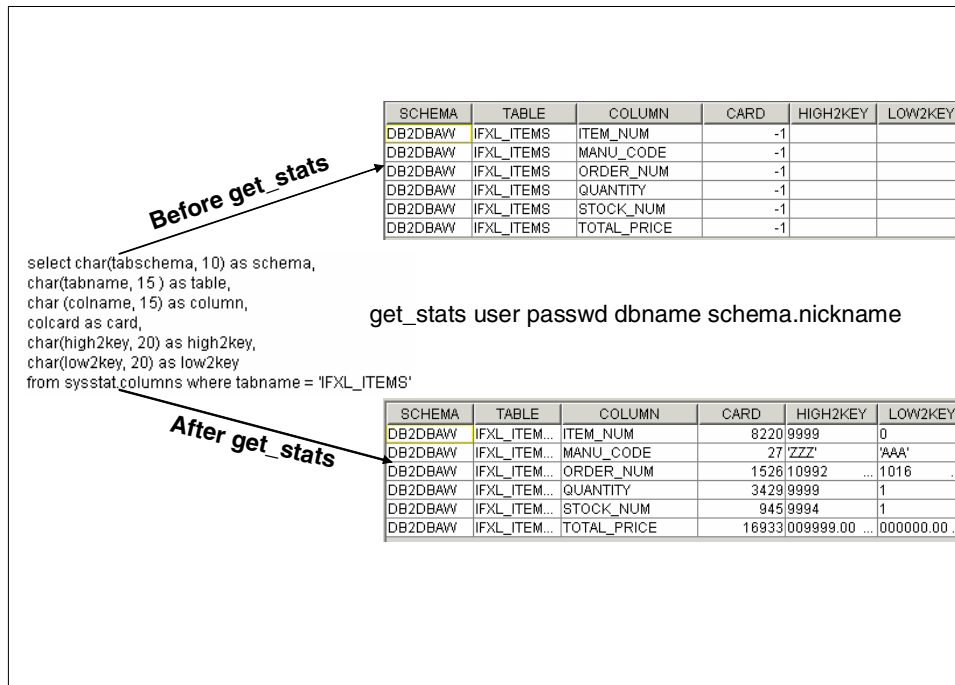


Figure 4-35 SYSSTAT.COLUMNS before and after get_stats execution

Another technique that can be used is to create a new nickname. You could drop and recreate the nickname so that new statistics are collected. However, you should consider the implications of doing this. For example, permissions on the nicknames would need to be granted again. Dependent views and packages would be marked as invalid. One way to avoid these complications is to *create a second nickname* for the same table in the data source. DB2 Information Integrator collects statistics from the data source table for the newly created nickname. You can then create a new nickname just to have the statistics collected, and then update the catalog tables in the federated server for an existing nickname based on the new statistics. For example, given an existing nickname *ifxl_items*, create a second nickname *ifxl_items_stats*. Next, query the catalog tables for tablename *ifxl_items_stats* and update the corresponding columns for the original table *ifxl_items*. Statistics are kept in two tables (SYSSTAT.TABLES, SYSSTAT.INDEXES, and SYSSTAT.COLUMNS).

Note: In our testing environment we saw dramatic changes on the query plans produced before and after the collection of statistics, especially for Red Brick. Because there are no statistic tools with Red Brick, at nickname creation, the statistics for the Red Brick nicknames in the federated server were misleading. A much better plan was chosen by the optimizer after we updated the statistics.

The CREATE INDEX with SPECIFICATION ONLY clause is also very useful for cases where a new index is added to a table in the data source after the creation of nicknames. It creates an index specification (metadata) that indicates to the query optimizer that a data source object has an index. No actual index is created in the federated server. Only the specification is created.

4.4.2 Informix synonyms and I-STAR

In an Informix environment it is common to use synonyms. These synonyms can either refer to a local table in a local database or a table in a remote database. With DB2 Information Integrator you can define nicknames on synonyms as long as they refer to another view or table in the same database. From the DB2 Information Integrator perspective these synonyms are seen as local tables within one remote data source. In this case, joins on these nicknames in this data source can be pushed down to the data source.

Informix customers often use the I-STAR capability of IDS to join tables between different Informix databases. It is common to use synonyms that refer to these remote tables. If you try to create a nickname in DB2 Information Integrator on an Informix synonym whose target table resides in a different Informix database from the synonym itself, you will get the error message shown in Figure 4-36. DB2 Information Integrator needs the column definitions for the Informix synonym, but this information is only kept in the database that the synonym refers to.

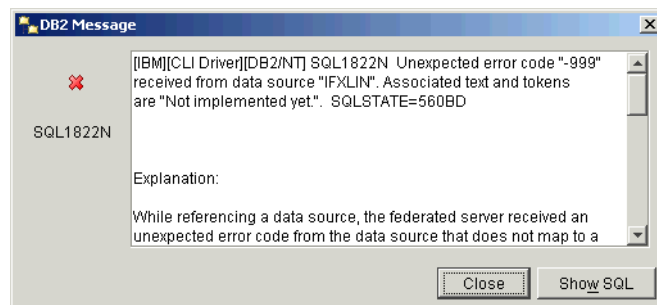


Figure 4-36 Error creating a nickname on an Informix synonym

Assume that we want to do a join between two different Informix databases with one table each and a synonym in one of the Informix databases that refers to the table in the other Informix database. To access both tables through DB2 Information Integrator we have two choices:

- ▶ We define two servers in our federated database, connecting to the two different Informix databases, shown in Figure 4-37. Each server has a nickname on one table. If we join these two tables we have to access two different data sources and a pushdown for the join will not be possible.

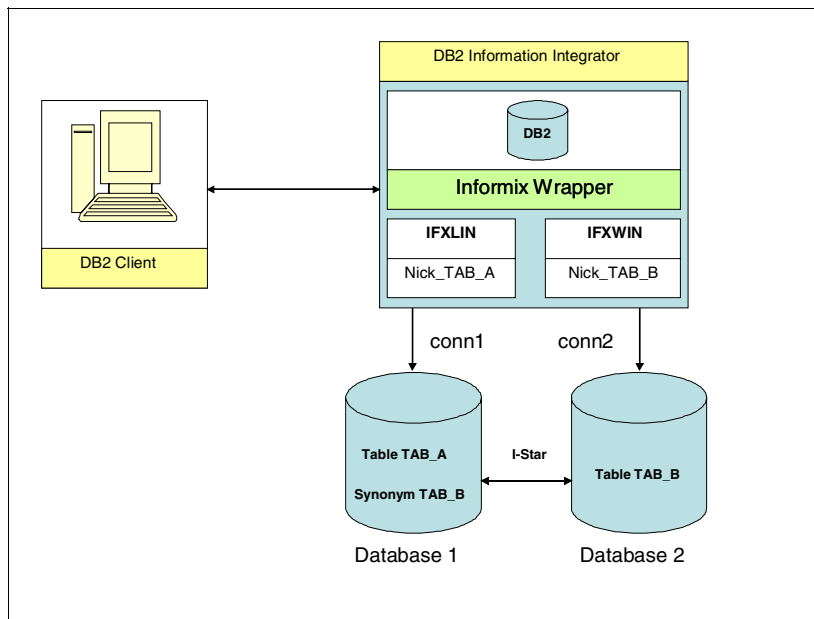


Figure 4-37 Synonyms - Scenario 1

- ▶ We replace the Informix synonym through a view that represents the table in the remote Informix database, shown in Figure 4-38 on page 113. In this case the necessary catalog information can be retrieved from the data source with the table and the view defined. A join between the table and the view can be done through a single data source and a pushdown for the join is possible. The I-STAR capability provides DB2 with a single data source although there are two different Informix databases used in this join.

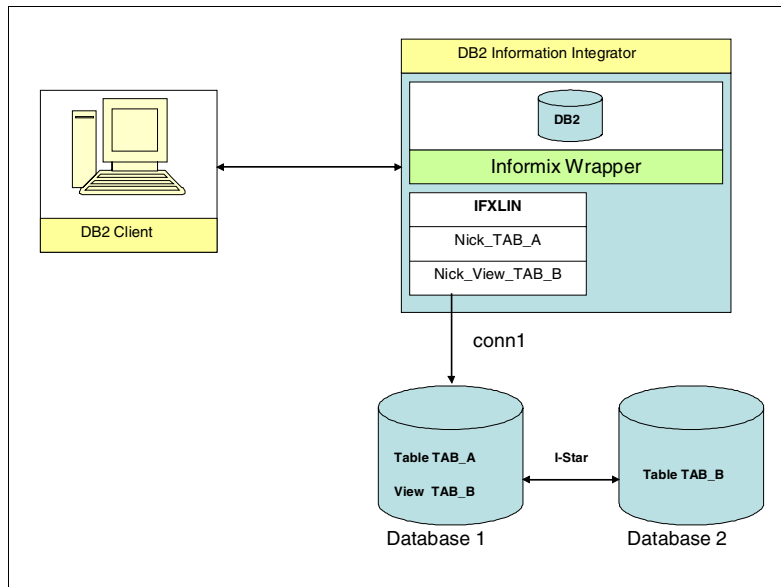


Figure 4-38 Synonyms - Scenario 2

4.4.3 Transaction support

DB2 Information Integrator supports one-phase commit operations. This includes:

- ▶ A non-distributed unit of work.
- ▶ A distributed unit of work that involves reading from one or more data sources but updating data at only one data source. This means that either the local DB2 database or one data source can be updated within one unit of work.

Important: Two-phase commit is currently not supported with DB2 II.

4.4.4 User mapping authentication for Informix data sources

When you define the user mapping for a particular user, you must declare the REMOTE_AUTHID and REMOTE_PASSWORD user options. If you do not declare them, the authentication at the remote server will be attempted using the current user and password. These options are case sensitive unless you set the FOLD_ID and FOLD_PW server options to U, L, or N (meaning Uppercase, Lowercase, or do Not fold) in the CREATE SERVER statement. If you set it to N, you must specify the user ID and password in the correct case.

4.4.5 Pass-through

As discussed in “Pass-through sessions” on page 81, pass-through sessions allow you to submit SQL statements *directly* to a data source. This functionality is useful when you want to perform an operation that is not possible with DB2 SQL. For example, to create a stored procedure or to run a particular administrative task, such as statistics update or index creation. In a pass-through session, SQL native to the data source is used, thus bypassing the DB2 SQL. The server name is provided in the SET PASSTHRU command, indicating where to send the subsequent commands. The pass-through session ends when SET PASSTHRU RESET is issued. The commands are shown in Example 4-7.

Example 4-7 Pass-through command example

```
SET PASSTHRU ifxlin;  
UPDATE STATISTICS LOW;  
SET PASSTHRU RESET;
```

4.4.6 Data type mapping

After creating a nickname, you may want to make adjustments on a column data type of the nickname in order to standardize the data types for the application. For example, you may want to have the column `stock_num` declared as *integer* on all nicknames regardless of what the data type is declared on its native form at the data source. In Figure 4-39 on page 115 you see a nickname for table *stock* created through the *discovery process* using the Control Center. The data type for column `stock_num` is mapped from number to *double*. The same column from a different data source may have been mapped to *smallint* or some other data type.

Although in most cases this is not a problem for DB2 Information Integrator, you may want to change your nickname definitions to a common data type in order to keep consistency throughout your application and data definitions in your federated environment.

The local data type of a nickname column can also affect the number of possibilities in a joining sequence evaluated by the optimizer. Ensure that the local data type of a column does not prevent a predicate from being evaluated at the data source. The default data type mappings are provided to avoid any overflow. However, a joining predicate between two columns of different lengths might not be considered at the data source whose joining column is shorter, depending on how DB2 binds the longer column. This situation can affect the number of possibilities in a joining sequence by the optimizer. Here's a classic example: Oracle data source columns created using the INTEGER and INT data type are given the type NUMBER(38). A nickname column for this Oracle data

type is from 2^{31} to $(-2^{31})-1$, which is roughly equal to NUMBER(9). In this case, joins between a DB2 integer column and an Oracle integer column cannot take place at the DB2 data source because of the shorter join column. However, if the domain of this Oracle integer column can be accommodated by the DB2 INTEGER data type, change its local data type so that the join can take place at the DB2 data source.

Name	Datatype	Size	Scale	Nulls?	Default Value
STOCK_NUM	NUMBER				
MANU_CODE	CHAR	3			
DESCRIPTION	CHAR	15		✓	
UNIT_PRICE	NUMBER	6	2	✓	
UNIT	CHAR	4		✓	
UNIT_DESCR	CHAR	15		✓	

Notice the data type for column stock_num on table stock is mapped from *number* on its native form to *double* in the newly created nickname

Column name	Data type	Length	Precision	Scale
STOCK_NUM	DOUBLE	-	-	-
MANU_CODE	CHARACTER	3	-	-
DESCRIPTION	CHARACTER	15	-	-
UNIT_PRICE	DECIMAL	6	6	2
UNIT	CHARACTER	4	-	-
UNIT_DESCR	CHARACTER	15	-	-

Figure 4-39 Data type mapping on newly created nickname

You can redefine the column data type by altering the nickname with the Control Center (as shown in Figure 4-40 on page 116) or by running the ALTER NICKNAME statement (as shown in the Example 4-8).

Example 4-8 Alter nickname statement

```
ALTER NICKNAME "DB2DBAW"."ORAW_STOCK"
ALTER COLUMN stock_num STOCK_NUM LOCAL TYPE INTEGER;
```

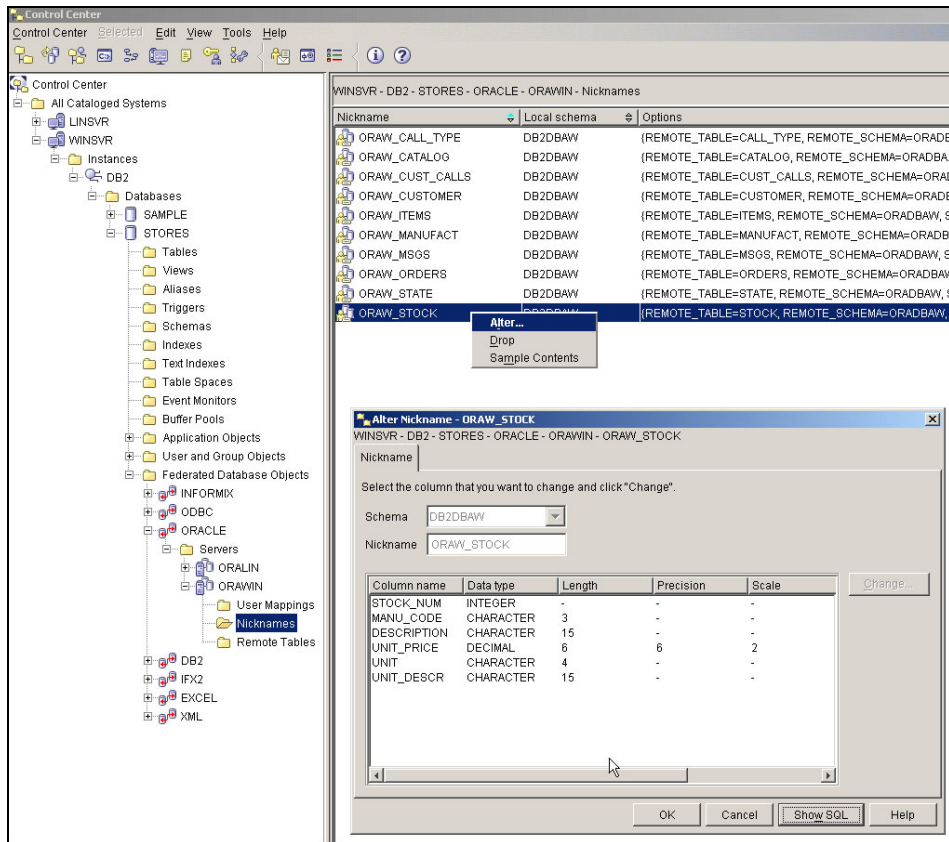


Figure 4-40 Altering column data type definition for nickname via the Command Center

4.4.7 Server options

Some of the information within a server definition is stored as *server options*. Server options are used with the CREATE SERVER statement to describe a data source server. Some server options configure the wrapper and some affect the way DB2 uses the wrapper. Data integrity, location, security, and performance information are among the options you can set for a particular server.

Server options can be set to persist over multiple connections to the data source, or set for the duration of a single connection. If you want to set it for the duration of a single connection you may use the SET SERVER OPTION command.

The SET SERVER OPTION command must be the first statement after establishing the connection, and the server option settings specified will remain

in effect for that particular application/connection until the application ends. This statement is not under transaction control.

You can get the server option settings for a given server by doing the following:

- ▶ Selecting from the system catalog tables: Server definitions are kept in the system catalog. You can query the SYSCAT.serveroptions table for a particular server definition as shown in Figure 4-41.

```
db2> describe table SYSCAT.serveroptions
```

Column	schema	name	Length	Scale	Nulls
WRAPNAME	SYSIBM	VARCHAR	128	0	Yes
SERVERNAME	SYSIBM	VARCHAR	128	0	Yes
SERVERTYPE	SYSIBM	VARCHAR	30	0	Yes
SERVERVERSION	SYSIBM	VARCHAR	18	0	Yes
CREATE_TIME	SYSIBM	TIMESTAMP	10	0	No
OPTION	SYSIBM	VARCHAR	128	0	No
SETTING	SYSIBM	VARCHAR	2048	0	No
SERVEROPTIONKEY	SYSIBM	VARCHAR	18	0	No
REMARKS	SYSIBM	VARCHAR	254	0	Yes


```
SELECT servername, option, setting
FROM SYSCAT.serveroptions
WHERE servername = 'IFXLIN'
```

SERVERN...	OPTION	SETTING
IFXLIN	COLLATING_SEQUENCE	Y
IFXLIN	DBNAME	stores
IFXLIN	IUD_APP_SVPT_ENFORCE	N
IFXLIN	NODE	ifx_linsvr_tcp
IFXLIN	PASSWORD	Y
IFXLIN	PUSHDOWN	Y

Figure 4-41 SYSCAT.serveroptions

- ▶ To extract the server definition via db2look, as described in “Capturing server options and nickname definitions” on page 121, you can use **db2look** to extract the server definitions in DDL format. Execute:

```
db2look -d <dbname> -e -o <output file>
```

The output file will contain the DDL for the server definitions, as shown in Example 4-9.

Example 4-9 DDL produced by db2look output

```
-- -----
-- DDL Statements for SERVER
```

```

-----
CREATE SERVER "ORAWIN"
  TYPE ORACLE
  VERSION 9
  WRAPPER ORACLE
  OPTIONS
    (NODE 'ora.winsvr'
     ,PASSWORD 'Y'
    );

```

- From the Command Center tool: When you expand the server list, click the **Alter** button on a particular server and select the **Settings** tab for a list of server options. See Figure 4-42 for details.

Note: Not all server options are shown via the Command Tool window. For a complete list of server options, query the system catalog or extract the server definition via the **db2look** utility.

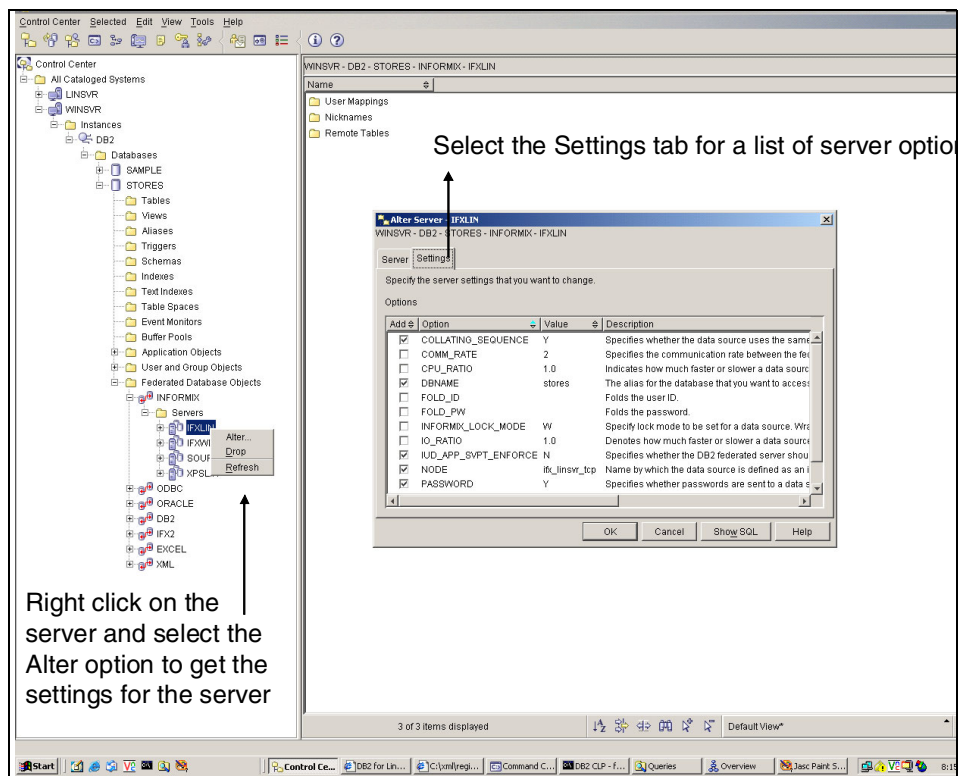


Figure 4-42 Displaying the server options list from the Control Center

4.4.8 Capacity planning for DB2 Information Integrator

In a federated environment you have distributed resources, such as CPU, memory, and disk space, for query processing at the different data sources. To achieve maximum performance in such an environment it is important to use as many of the available resources as possible concurrently. In addition to an efficient use of resources at the data source, it is also important that the network traffic between the federated server and the data source, generated through queries, is as minimal as possible. This is typically achieved when as many parts of a query are pushed down to the remote data sources as possible. However, there might be situations where it is better, or necessary, to process parts of the query at the federated server. In these situations you need to make sure that the machine where DB2 Information Integrator is running will not be the performance bottleneck in your federated environment. Consider the following for the federated server:

- ▶ Run DB2 Information Integrator on a dedicated machine or an existing machine with sufficient spare resources.
- ▶ Plan sufficient CPU and main memory resources for complex queries, such as joins, group by, and sorts, and for the number of users that will be submitting queries to the federated system.
- ▶ Plan sufficient permanent and temporary disk space for query processing, MQTs, and local instances.

These are basically the same considerations that would be applicable for any machine where DB2 is running. Therefore, you should also optimize your I/O subsystem and the DB2 instance where DB2 Information Integrator is running. It would be very difficult to suggest a standard configuration for a federated server, so our recommendation is that you should simply be aware that capacity planning for a federated server, as with other systems, is very important for overall good system performance.

4.5 Troubleshooting

With so many components involved in a federated environment, isolating the problem to a particular component may become a challenging task. In this section we provide you with some guidance to assist you in the problem determination process.

4.5.1 Connecting to data sources

If you encounter problems selecting from a data source, make sure you can access the database target from outside of DB2 Information Integrator. For

example, by connecting to it from a simple application using the native client, or through an ODBC connection using the same user ID and password as used with DB2 Information Integrator. If you cannot connect to your data source from outside of DB2, you should focus on the client/server configuration. You must be able to connect to your data source successfully before you can continue troubleshooting.

4.5.2 Error messages

Wrappers usually issue errors by filling in the SQLCA with the SQLCODE and message inserts. DB2 returns the SQLCA to the application. For example, if the SQL statements are issued through the command-line processor (CLP) then the CLP displays error messages SQL0901 and SQL0902, which indicate that an internal error in the wrapper of the DB2 engine has occurred.

Error codes from remote data sources are usually converted to the equivalent DB2 error code, if there is one. If there is no equivalent, then the wrapper will usually issue error SQL1822 along with some additional information that the wrapper might also have written to db2diag.log.

SQL1822: Unexpected error code "<error code>" received from data source "<data source name>".

Associated text and tokens are "<tokens>".

Look up the error code in the data source documentation. For example, you may want to use the **finderr** utility in Informix to get a description of the error code. The DBMS message file may also contain additional information that can help you to isolate the problem.

Possible causes of SQL statement errors include:

- ▶ The wrapper issued an SQL statement that is *syntactically* invalid. For Informix data sources, the offset of the error within the SQL statement is in sqlerrd[4] in the Informix SQLCA, which is shown in the diaglog.log.
- ▶ The wrapper issued an SQL statement that is *semantically* invalid. The most likely cause is a logic error in the wrapper. Please consult your DB2 Technical Support Representative.

4.5.3 Informix environment settings and DB2DJ.INI file

The db2dj.ini file is only picked up during **db2start**. If you make changes to the db2dj.ini file, make sure you recycle DB2 so that the new settings can take effect. You can inspect the variables in effect for a particular query execution. By tracing the execution you will be able to see the environment variables and the context. See iterations 87916 through 87918 in Example 4-14 on page 127.

4.5.4 Capturing server options and nickname definitions

Server definitions are kept in catalog tables. You can obtain information about wrappers, servers, nicknames, and server options by querying the system catalog tables. The main tables are:

- ▶ SYSCAT.wrappers
- ▶ SYSCAT.servers
- ▶ SYSCAT.serveroptions
- ▶ SYSCAT.useroptions

In addition, you can collect the DDL statements by using the **db2look** utility. See Example 4-10. **db2look** will extract DDL statements for:

- ▶ User mappings
- ▶ Nicknames
- ▶ Servers
- ▶ Wrappers
- ▶ Indexes with specifications only

Example 4-10 Collecting DDL statements with db2look -d STORES -e

```
-- -----  
-- DDL Statements for CREATE INDEX...SPECIFICATION ONLY  
-- -----  
  
CREATE INDEX "DB2DBAW"."ZIP_IX0"  
  ON "DB2DBAW"."IFXW_CUSTOMER" (ZIPCODE ASC)  
  SPECIFICATION ONLY;  
  
CREATE INDEX "DB2DBAW"."ZIP_IX1"  
  ON "DB2DBAW"."NICK" (ZIPCODE ASC)  
  SPECIFICATION ONLY;  
-- -----  
-- DDL Statements for WRAPPER  
-- -----  
  
CREATE WRAPPER "INFORMIX"  
  LIBRARY 'db2informix.dll'  
  OPTIONS (DB2_FENCED 'N'  
  );  
CREATE WRAPPER "ORACLE"  
  LIBRARY 'db2net8.dll'  
  OPTIONS (DB2_FENCED 'N'  
  );  
  
-- -----  
-- DDL Statements for SERVER  
-- -----
```

```

CREATE SERVER "ORAWIN"
  TYPE ORACLE
  VERSION 9
  WRAPPER ORACLE
  OPTIONS
    (NODE 'ora.winsvr'
     ,PASSWORD 'Y'
    );

CREATE SERVER "SOURCE"
  TYPE INFORMIX
  VERSION 9
  WRAPPER INFORMIX
  OPTIONS
    (DBNAME 'source'
     ,IUD_APP_SVPT_ENFORCE 'N'
     ,NODE 'ifx_linsvr_tcp'
     ,PASSWORD 'Y'
    );
-----
-- DDL Statements for USER MAPPING
-----
CREATE USER MAPPING FOR DB2DBAW
  SERVER "DB2LIN"
  OPTIONS
    (REMOTE_AUTHID 'db2dbal'
     ,REMOTE_PASSWORD ''
    );

CREATE USER MAPPING FOR DB2DBAW
  SERVER "IFXLIN"
  OPTIONS
    (REMOTE_AUTHID 'ifxdbal'
     ,REMOTE_PASSWORD ''
    );

-----
-- DDL Statements for NICKNAME
-----

CREATE NICKNAME "DB2DBAW"."DB2L_CALL_TYPE"
  FOR "DB2LIN"."DB2DBAL"."CALL_TYPE";

ALTER NICKNAME "DB2DBAW"."DB2L_CALL_TYPE"
  ALTER COLUMN "CALL_CODE" LOCAL TYPE CHARACTER (1);
ALTER NICKNAME "DB2DBAW"."DB2L_CALL_TYPE"
  ALTER COLUMN "CODE_DESCR" LOCAL TYPE CHARACTER (30);

```

4.5.5 Access plans and the query optimizer

DB2 comes with a very powerful query optimizer that is cost-based. Cost-based algorithms will attempt to determine the lowest cost way, in terms of performance and resource usage, to perform your federated query. When a query is issued against the database, several major events occur including the generation of the query graph model, and the query rewrite and optimization phases. By examining the strategy chosen by the optimizer based on your settings, statistics, and resources, you may gain a better understanding of how the data is being collected and returned to the application. And that is critical when investigating performance problems.

In order to view the access plan chosen by the optimizer first you have to create the EXPLAIN tables for the database, as shown below:

```
db2 connect to stores
db2 -tvf $HOME/sqllib/misc/EXPLAIN.DDL
```

This will create EXPLAIN tables, which will then be populated by the optimizer when a request to capture the access plan is made. Using the **db2exfmt** tool, you can extract this information and generate a copy of the access plan used by the optimizer. The commands are shown in Example 4-11.

Example 4-11 Viewing the access plan

```
db2 connect to stores
db2 set current explain mode explain
db2 "select * from IFXL_CUSTOMER"
db2 set current explain mode no
db2exfmt -d stores -g TIC -w -1 -s % -n % -o db2exfmt.out
```

Alternatively you can use the Visual Explain option from the Command Center, which draws the plan graph in a GUI interface. You can access the Visual Explain by selecting from the top menu, **Interactive -> Create Access Plan**.

For additional information on interpreting query plans, please consult the “*performance problem determination*” section in the Problem Determination series which is available online at:

http://www.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/tutorial_main.d2w/toc

Note: Effectively designing and tuning a system is key to improved access plans and faster query run times.

4.5.6 Diagnostic information to collect

When contacting technical support, it is a good idea to collect the db2diag.log along with the output of **db21eve1** and the error messages received by the application level. Also, by matching the timestamp of the error, try to identify any messages produced at the data source that may have been associated with the error reported by the application. The **db2support** utility is also very useful when collecting information about the environment, as shown in Example 4-12. It includes system information, database configuration, and environment variable settings.

Example 4-12 Collecting diagnostic information with db2support

```
Output file is "C:\tmp\db2look\db2support.zip"
Time and date of this collection: "07/09/2003 05:09:23 PM Pacific Daylight
Time"
```

```
Collecting "Basic operating system and hardware information"
Collecting "System files"
```

```
"db2cchis.prf"
"db2dbamr.prf"
"db2eventlog.000"
"db2misc.prf"
"db2nodes.cfg"
"db2profile.bat"
"db2system"
"db2tools.prf"
"jdbcerr.log"
"db2dasdiag.log"
"DB2SYSTEM"
"db2cli.ini"
"SQLDBBAK"
"SQLDBDIR"
"SQLDBINS"
"SQLNOBAK"
"SQLNODIR"
"report.log"
"CPUinfo.out"
"diskusage.out"
"DrWatson.out"
"hosts.out"
"ipconfig.out"
"kernelinfo.out"
"memusage.out"
"netshare.out"
"netstat.out"
"netstat_an.out"
"netstat_en.out"
```

```
"netstat_r.out"
"netuse.out"
"networks.out"
"numdisks.out"
"protocol.out"
"services.out"
"set.out"
"ver.out"

Collecting "System resource info (disk, CPU, memory)"
Collecting "Operating system and level"
Collecting "JDK Level"
Collecting "DB2 Release Info"
Collecting "DB2 install path info"
Collecting "Registry info"
Collecting "EEE node information"
Collecting "Get dbm cfg"
Collecting "Get admin cfg"
Collecting "List database directory"
Collecting "Sqllob directory listing"
Collecting "List node directory"
Collecting "List admin node directory"
Collecting "List DCS directory"
Collecting "Get cli cfg"
Collecting "List DCS applications extended"
Collecting "Query client"
Collecting "List active databases"
Creating final output archive
  "db2support.html"
  "db2_sqllob_directory.txt"
  "db2supp_system.zip"
  "admin.cfg"
  "cli.cfg"
  "dbm.cfg"
  "JDKlevel.cfg"
  "active_db.out"
  "admin_nodedir.out"
  "db2level.out"
  "db2set.out"
  "db_dir.out"
  "dcsext.out"
  "dcs_dir.out"
  "nodedir.out"
  "query_client.out"
  "db2diag.log"
```

db2support is now complete.

An archive file has been produced: "db2support.zip"

Note: **db2support** does not collect the db2dj.ini file. When submitting diagnostic information to DB2 Technical Support, please include the db2dj.ini file along with the **db2support** output file.

4.5.7 Tracing DB2 Information Integrator wrapper calls

If you experience a recurring and reproducible problem with DB2, tracing sometimes allows you to capture additional information about it. Under normal circumstances, you should only use a trace if asked to by DB2 Customer Support. The process of taking a trace entails setting up the trace facility, reproducing the error, and collecting the data. When taking a trace, use the smallest scenario possible to reproduce a problem.

All major wrapper functions trace their entry and exit. Most functions also trace their input and output parameters, and a few intermediate data values. A list of tracing masks is provided in Table 4-4.

Table 4-4 Tracing masks for each relational wrapper

Wrapper	Tracing mask
DRDA	**.44.**
Informix	**.71.**
MS SQL Server	**.68.**
OLE DB	**.40.**
Oracle (net8)	**.63.**
Oracle (sqlnet)	**.39.**
Sybase (ctlib)	**.66.**
Sybase (dblib)	**.65.**
Teradata	**.37.**

An example of a tracing session collected with **db2trc** is shown in Example 4-13 on page 127. In this example, we turn the tracing facility ON to trace Informix calls processed by the wrapper. The output is in both *flow* and *format* formats. For additional information please consult the DB2 Command Reference manual at:

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

Example 4-13 Collecting trace information with db2trc

```
C:\Program Files\IBM\SQLLIB\BIN>db2trc on -m *.*.71.*.*
Trace is turned on

C:\Program Files\IBM\SQLLIB\BIN>db2 insert into items_tar select * from
items_src

DB21034E The command was processed as an SQL statement because it was not a
valid Command Line Processor command. During SQL processing it returned:
SQL1822N Unexpected error code "-400" received from data source "SOURCE".
Associated text and tokens are "Fetch attempted on unopen cursor.".
SQLSTATE=560BD

C:\Program Files\IBM\SQLLIB\BIN>db2trc dmp db2trc.dmp
Trace has been dumped to file "db2trc.dmp"

C:\Program Files\IBM\SQLLIB\BIN>db2trc dmp db2trc.dmp

C:\Program Files\IBM\SQLLIB\BIN>db2trc off
Trace is turned off

C:\Program Files\IBM\SQLLIB\BIN>db2trc fmt db2trc.dmp db2trc.fmt
Trace truncated          : NO
Trace wrapped           : YES
Total number of trace records : 88065
Number of trace records formatted : 88065

C:\Program Files\IBM\SQLLIB\BIN>db2trc flw db2trc.dmp db2trc.flw
Total number of trace records : 88065
Trace truncated              : NO
Trace wrapped                : YES
Number of trace records formatted : 88065 (pid: 1064 tid 1680 node: 0)

C:\Program Files\IBM\SQLLIB\BIN>
```

If you examine the contents of the two files produced on Example 4-13 you will find details such as returning status of internal functions being called, arguments passed and returned, and SQLCA and SQLRA structures. Example 4-14 shows an excerpt of the trace output in formatted as *format*.

Example 4-14 DB2 trace output (format)

```
87909entry DB2 Informix wrapper informix_esql_fetch_cursor fnc (1.3.71.292.0)
  pid 1064 tid 1680 cpid -1 node 0 sec 1 nsec 835224703

87910data DB2 Informix wrapper informix_esql_fetch_cursor fnc (3.3.71.292.0.10)
  pid 1064 tid 1680 cpid -1 node 0 sec 1 nsec 835225291 probe 10
  bytes 9
```

```

73716C63 75723030 31                                sqlcur001

87911exit DB2 Informix wrapper informix_esql_fetch_cursor fnc (2.3.71.292.0)
  pid 1064 tid 1680 cpid -1 node 0 sec 1 nsec 835280365
  rc = 0xFFFFFE70 = -400

87912entry DB2 Informix wrapper Informix_Connection::report_error fnc
(1.3.71.112.0)
  pid 1064 tid 1680 cpid -1 node 0 sec 1 nsec 835282725

87913data DB2 Informix wrapper Informix_Connection::report_error fnc
(3.3.71.112.0.10)
  pid 1064 tid 1680 cpid -1 node 0 sec 1 nsec 835283769 probe 10
  bytes 1
  00

87914data DB2 Informix wrapper Informix_Connection::report_error fnc
(3.3.71.112.0.20)
  pid 1064 tid 1680 cpid -1 node 0 sec 1 nsec 835284913 probe 20
  bytes 116
  70FEFFFF 00000000 00000000 00000000 p.....
  00000000 00000000 00000000 00000000 .....
  00000000 00000000 00000000 00000000 .....
  00000000 00000000 00000000 00000000 .....
  00000000 00000000 00000000 00000000 .....
  00000000 01000000 00000000 01000000 .....
  01000000 00000000 86050000 20202020 .....
  20202020

87915data DB2 Informix wrapper Informix_Connection::report_error fnc
(3.3.71.112.0.30)
  pid 1064 tid 1680 cpid -1 node 0 sec 1 nsec 835449125 probe 30
  bytes 36
  494E464F 524D4958 44495263 3A5C7072  INFORMIXDIRc:\pr
  6F677261 6D206669 6C65735C 696E666F  ogram files\info
  726D6978                                     rmix

87916data DB2 Informix wrapper Informix_Connection::report_error fnc
(3.3.71.112.0.40)
  pid 1064 tid 1680 cpid -1 node 0 sec 1 nsec 835650578 probe 40
  bytes 28
  494E464F 524D4958 53455256 45526966  INFORMIXSERVERif
  785F7769 6E737672 5F746370                                     x_winsvr_tcp

87917data DB2 Informix wrapper Informix_Connection::report_error fnc
(3.3.71.112.0.50)
  pid 1064 tid 1680 cpid -1 node 0 sec 1 nsec 835670965 probe 50
  bytes 25
  434C4945 4E545F4C 4F43414C 45656E5F  CLIENT_LOCALEen_

```

```

75732E43 50313235 32                               us.CP1252

87918data DB2 Informix wrapper Informix_Connection::report_error fnc
(3.3.71.112.0.60)
  pid 1064 tid 1680 cpid -1 node 0 sec 1 nsec 835707448 probe 60
  bytes 9
  44425F4C 4F43414C 45                               DB_LOCALE

87919entry DB2 Informix wrapper Informix_Connection::report_db2_sqlcode fnc
(1.3.71.109.0)
  pid 1064 tid 1680 cpid -1 node 0 sec 1 nsec 835755633

87921entry DB2 Informix wrapper Informix_Connection::report_db2_sqlcode_1822
fnc (1.3.71.110.0)
  pid 1064 tid 1680 cpid -1 node 0 sec 1 nsec 835759176

87922data DB2 Informix wrapper Informix_Connection::report_db2_sqlcode_1822 fnc
(3.3.71.110.0.20)
  pid 1064 tid 1680 cpid -1 node 0 sec 1 nsec 836321941 probe 20
  bytes 34
  46657463 68206174 74656D70 74656420  Fetch attempted
  6F6E2075 6E6F7065 6E206375 72736F72  on unopen cursor
  2E0A                                             ..

87924error DB2 Informix wrapper Informix_Connection::report_db2_sqlcode_1822
fnc (4.3.71.110.0.40)
  pid 1064 tid 1680 cpid -1 node 0 sec 1 nsec 836374947 probe 40
  bytes 4
  60012680                                         `.&.

87925exit DB2 Informix wrapper Informix_Connection::report_db2_sqlcode_1822 fnc
(2.3.71.110.0)
  pid 1064 tid 1680 cpid -1 node 0 sec 1 nsec 836376017
  rc = 0x80260160 = -2144992928 = SQLQG_ERROR

```



Informix Enterprise Gateway Manager

IBM Informix Enterprise Gateway Manager (EGM) is a member of the Informix Enterprise Gateway family, a complete set of standards-based gateways that also includes IBM Informix Enterprise Gateway with DRDA. IBM Informix Enterprise Gateway Manager allows Informix database application developers and users to access information on DB2, Oracle, Sybase, and other non-Informix databases as easily and transparently as if they were accessing an Informix database server. In combination with the industry-leading parallel capabilities of Informix Dynamic Server, IBM Informix Enterprise Gateway products provide high-performance solutions for joining and updating data across heterogeneous data sources such as DB2, Oracle, Sybase, and others.

In this chapter we provide an overview of the Enterprise Gateway Manager discussing installation and configuration as well as considerations when using this powerful tool in a federated environment.

5.1 Product overview

IBM Informix Enterprise Gateway Manager (EGM) is designed for IBM Informix customers who also need access to non-Informix databases such as DB2, Oracle, and Sybase. At the core of the product is an SQL-based gateway that allows Informix tools, applications, and databases to interoperate transparently with non-Informix databases. Enterprise Gateway Manager can help customers migrate from, or coexist in, a heterogeneous database environment.

Universal access to enterprise data

Based on the Open Database Connectivity (ODBC) specification from Microsoft, Enterprise Gateway Manager provides access to non-Informix data sources through ODBC Drivers that can be loaded dynamically and enable gateway access to a number of legacy data sources, such as ISAM and VSAM.

ODBC

Microsoft has developed a call level interface (CLI) called Open Database Connectivity (ODBC) that implements a version of the standard CLI from the X/Open SQL Access Group (SAG). ODBC is an API that provides transparent data access to relational databases. Application developers can use ODBC as a standardized way of writing database-independent applications.

The ODBC architecture has four components:

- ▶ ODBC-compliant application
The ODBC-compliant application performs the ODBC function calls, submits SQL statements, and retrieves results.
- ▶ ODBC Driver Manager
The ODBC Driver manager is supplied by Microsoft as a dynamically linked library (DLL) for Windows, and as shared libraries for UNIX from other vendors. The driver manager loads the requested driver on behalf of the application and provides the interface to the application.
- ▶ ODBC Driver
The ODBC Driver processes the ODBC function calls, modifies the request to conform to the data source's syntax, submits SQL requests to a specific data source, and returns results to the application.
- ▶ Data source
The data source is the DBMS where the data resides and may include sources such as ISAM, VSAM, and other nonrelational sources, as well as relational data sources.

The ODBC Driver and the driver manager appear to the application as one unit that processes ODBC calls.

IBM Informix Enterprise Gateway Manager features

Enterprise Gateway Manager translates Informix SQL requests into ODBC-compliant function calls. Using emulation, it makes the underlying target DBMS appear to both client and server applications as an instance of Informix Dynamic Server. Users can access data on the target DBMS from Informix client applications written with products such as Informix ESQL/C, or using IDS OnLine 5.x to coordinate a distributed join.

Distributed access model

Distributed access means that the Informix client application connects to Informix Dynamic Server or another Informix DSA-based server, Version 5.x or higher. This server coordinates a distributed join between one or more Informix servers and/or other data sources, using EGM or Informix Enterprise Gateway with DRDA. This is depicted in Figure 5-1.

When the client connects to Enterprise Gateway Manager in distributed-access mode from a coordinating Informix server, Enterprise Gateway Manager identifies itself as a database with characteristics identical to those of another Informix participating database server.

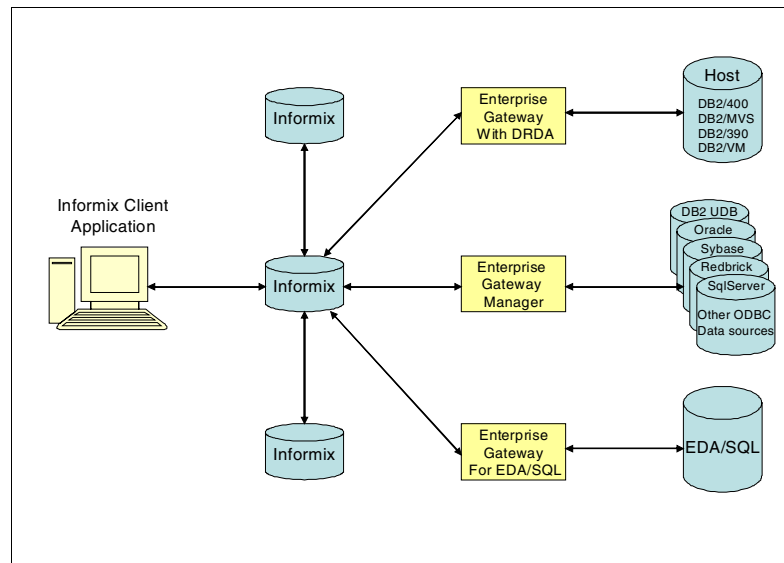


Figure 5-1 Distributed access model

Distributed join capability

Using Informix Dynamic Server, customers can perform distributed joins with, for example, DB2, Oracle, Red Brick, and Sybase, data in a single SQL statement. In fact, customers can transparently join data between Informix OnLine 5.x or Informix Dynamic Server, and any data source accessible through Enterprise Gateway Manager, Informix Enterprise Gateway with DRDA, and Informix Enterprise Gateway for EDA/SQL. With this capability, data from various locations within a company can be integrated with data in an Informix database.

User mapping and security

Enterprise Gateway Manager facilitates a security system that permits only authorized users to access remote data sources. The user of Enterprise Gateway Manager must have a valid user ID for the data source. Enterprise Gateway Manager provides mechanisms that enable mapping of user IDs and passwords between the different database servers. This mechanism is built into the gateway database administration facility, *egmdba*.

Unrestricted connection support

The ODBC interface allows Enterprise Gateway Manager to connect to a wide range of data sources that might have varying levels of support for transactions or different isolation levels. Combined with the appropriate ODBC Driver, unrestricted connection support can provide access to nonrelational data sources such as file systems that do not provide transaction- or isolation- level support.

SQL syntax support

Enterprise Gateway Manager provides SQL92 Entry Level support. Tools and applications can use SQL to read from and write to Oracle, Sybase, and other data sources. The SQL syntax supported includes the ANSI-standard SQL statements SELECT, INSERT, UPDATE, and DELETE.

Scrolling cursor support

EGM includes scrolling cursor support, which lets users move backward and forward through retrieved data sets for added convenience and enhanced user productivity.

System requirements

EGM requires:

- ▶ An installed target DBMS such as DB2, Red Brick, Oracle, and Sybase
- ▶ An installed supported ODBC Driver or customized driver for the target DBMS
- ▶ An installed EGM resident on the same computer as the ODBC Driver

- ▶ An installed client application that connects to Informix
- ▶ A valid connection between the Informix client product and the Gateway, such as local pipe, TCP, or IPX/SPX
- ▶ A copy of either IBM INFORMIX OnLine 5 or Informix Dynamic Server 7.x and later

Note: IBM Informix XPS can be a participant on a federated environment with I-STAR beginning on version 8.40 but it cannot act as a coordinator server.

5.1.1 Systems environment

EGM uses a single-threaded architecture, meaning each user of EGM is served by a different operating system process. Client applications connect to an Informix database server (*coordinator*) that will then send the remote portion of the query to EGM, which might or might not reside on the same computer. For this publication, we installed EGM on a separate Windows machine where there was no Informix database server installed. We called it the *gateway server computer*.

In Figure 5-2 on page 136 we show an Informix client connecting to IDS, which is a participant in an I-STAR environment that includes Informix XPS and Informix OnLine 5. EGM is installed in another system where all the ODBC Drivers are installed and configured to access the remote data sources through their respective clients.

When you install EGM, one gateway service is created and started. More than one EGM process can run at the same time on the same gateway server computer. EGM processes are spawned from the gateway service when the client connects to the EGM through a coordinating Informix database server.

Each gateway service is associated with a unique `gateway_server_name`. The `gateway_server_name` that identifies a gateway service is the Informix server name that is used in SQL statements to access the gateway service.

In Windows, the service name assigned to a gateway service is Informix EGM Gateway. Gateway service names are shown in the list of available Windows Services that can be accessed via the Windows Control Panel. You can also get a list of gateway services defined in your system by running `egmconfig -view`.

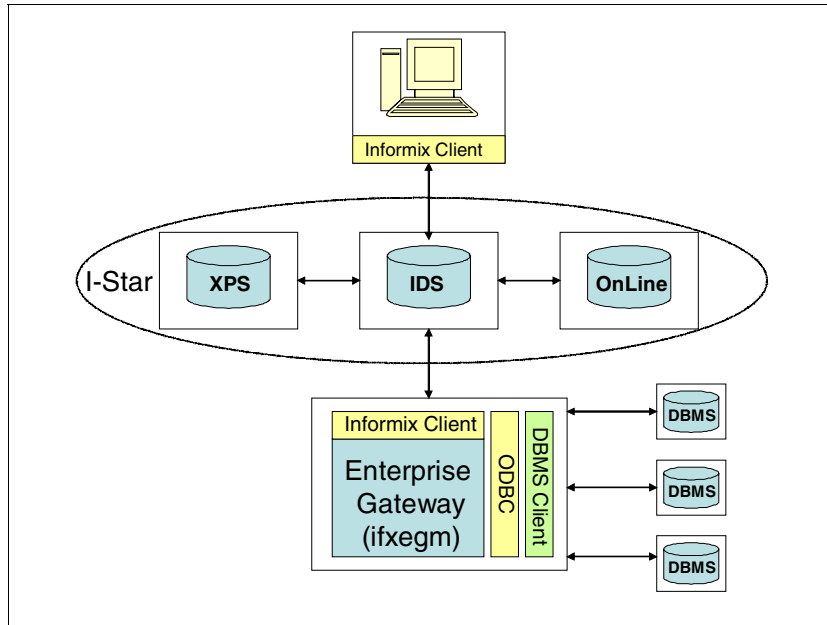


Figure 5-2 Architecture diagram for EGM

The EGM uses a gateway service to *listen* for processing requests. Although you usually need only one gateway service, you can have multiple gateway services running on your server. The gateway services share the same EMG installation directory. Please refer to “Multiple data sources within a single application” on page 151 for additional details.

An Informix database server relates to the EGM server as if the gateway were a subordinate database server. However, the data sources that the EGM accesses are different from Informix database servers. When you write the application code, you must use syntax that is acceptable to the Informix database server, the data source, and the ODBC Driver, since the statement will be executed in a three-step process. Here are the steps:

1. Processed by the Informix database server

The behavior of the Informix database server is not influenced when it is attached to EGM. Informix OnLine checks the SQL statements for correct Informix SQL syntax. If a statement does not comply with Informix SQL syntax, the statement execution fails and the statement is not sent to the EGM. Therefore, the following rules apply:

- SQL statements that do not conform to Informix SQL syntax cannot be sent to the data source.

- Because Informix database server does not permit DDL statements to be sent to the subordinate database servers, you cannot send DDL statements to the data source.
- Non-DDL SQL statements that conform to Informix SQL syntax can be sent to the data source.
- A fully qualified reference to an Informix database base object has the following format: *database@servername:owner.object*. The Informix coordinator uses the *servername* value to connect to the server (in this case, the EGM). The EGM then uses the *database* value to connect to the data source.

2. Processed by the EGM

Statements that the EGM receives for processing are subject to further processing by the EGM. The EGM performs the following SQL transformations before it sends an SQL statement to the data source:

- The EGM recognizes the limits of the target data source and enforces limits on authorization, and table and column name length.
- The Informix database server coordinator uses the server name value to connect to the server (in this case, the EGM). The EGM then uses the database value to connect to the data source.
- The EGM changes four-part table names to conform to the two-part table-name syntax of the data source. That is, four-part object names of the form *database@gateway_service:owner.table* are turned into two-part object names of the form *owner.tabname* truncated to the length that the data source supports.
- The EGM provides temporary storage and management for BLOB data and scrollable cursors.
- If necessary, the EGM changes the standard Informix catalog table names found in SQL statements to the actual Informix system catalog table names used on data sources.
- The EGM allows the use of wild card characters used by Informix tools translating MATCHES, *, ? into LIKE keywords. This mapping permits existing applications to work unmodified with the EGM.
- The EGM transforms delimited identifiers into a form acceptable to the ODBC Driver according to the ODBC Driver support.

3. Processed by the data source

All SQL statements are sent to the data source as dynamic SQL statements. The data source processes each statement and sends the results back to the EGM. The EGM appears as a client application to the data source.

Figure 5-3 on page 138 shows the steps taken during a remote query execution.

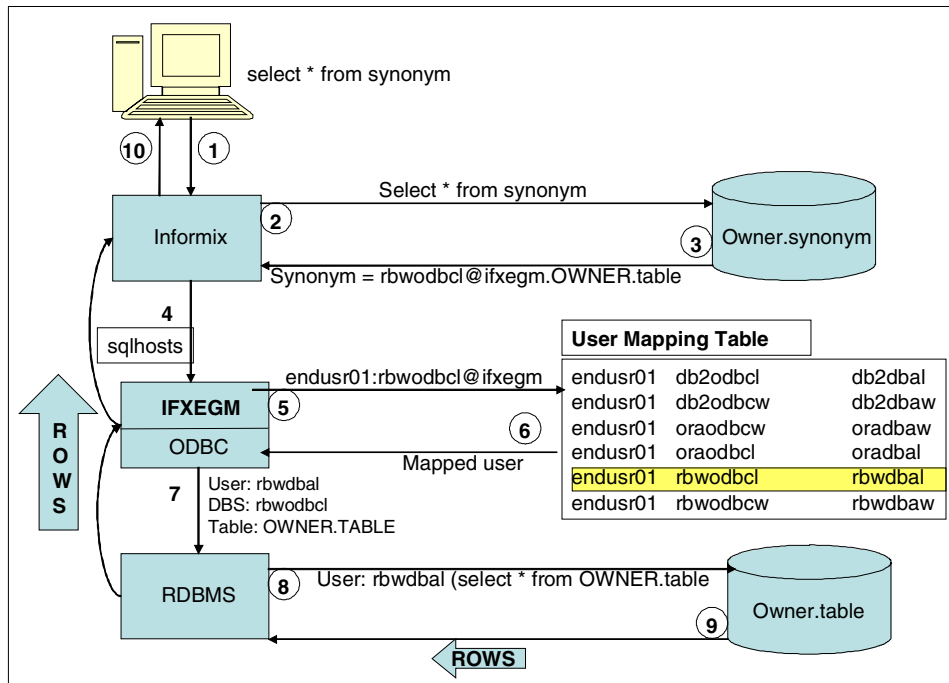


Figure 5-3 Remote query execution steps

Here is the description of each of the steps taken:

1. The client application is connected to an Informix database server, which selects from an object name defined as a synonym or view (rbwodbcl@ifxegm.OWNER.table) where rbwodbcl is a valid data source name (DSN) defined in the gateway machine and ifxegm is the gateway server name.
2. Informix parses the select and resolves the synonym or view object name into a four-part fully qualified name.
3. Now Informix knows it is a remote table at rbwodbcl@ifxegm.
4. The request is passed to the ifxegm server based on the SQLHOSTS file. The EGM is seen as another INFORMIXSERVER in the Informix environment.
5. The EGM identifies the DSN (rbwodbcl) and starts a new process to serve the Informix request.
6. The EGM validates the local user (endusr01) against the user mapping table and associates the local user to the remote user (rbwodbcl) that will be used to connect to the remote database on Red Brick (dsn=rbwodbcl).

7. A connection to the remote database server is established as user RBWDBAL.
8. The select is processed by the remote DBMS as user RBWDBAL.
9. User RBWDBAL must have the appropriate privileges granted to be able to select from table OWNER.TABLE. If the access is granted, the query is executed.
10. Results are passed back to the link of requester, then to the client application.

5.2 Installation and configuration

The installation and configuration of an EGM environment involves the setup of ODBC data sources, installation of DBMS clients, and the setup of your EGM server and services, as well as configuring the user mapping—among other things. The instructions found in this section are specific to the Windows operating system. For details on installing EGM on a Unix system, please refer to the Installation Guide for your particular Unix operating system.

5.2.1 Pre-installation requirements

Before getting into the actual installation, there are a number of tasks that need to be done first. Those pre-installation tasks are discussed below.

Configuring RDBMS clients

If you do not already have the DBMS client software installed in your machine, install and configure the client making sure a client application can access the target DBMS. The DBMS client library and tools should be installed on the computer where the gateway is installed.

You can invoke the appropriate client SQL tool and connect to the target DBMS in order to confirm your client/server setup is correct.

Configuring ODBC data sources

EGM accesses target data sources using ODBC. You must install and configure your ODBC data sources, mapping them to the databases residing in the target DBMS.

You can either use the ODBC Drivers provided with EGM or use the ODBC Driver of your choice. Most RDBMS client packages include an ODBC Driver.

You must define your data source as a *system data source*.

On the gateway system, please test your ODBC connection using an ODBC application to make sure you can access the target DBMS with ODBC and using a valid user id and password. Most ODBC Drivers provide a simple interface that can be used to test the access to the data source. A data source definition is also known as Data Source Name (DSN).

5.2.2 Installation instructions

Once the pre-installation tasks are complete, you are ready to begin installation. This section contains a description of the steps to complete installation.

Installation steps

To install the EGM, log in as a member of the Windows Administrators group. From the software distribution, execute **setup.exe** to start the installation process. In the Informix User Verification window, you will be asked for the password of the user called *informix*. If the user *informix* does not exist in your machine, the installation process will create one for you.

Tip: On Windows systems, if you change the user *informix* password at the operating system level, make sure you reflect the change in the service registration, as the gateway service is started as user *informix* and the password is kept in the startup settings for the service.

If you encounter an error during the installation process where the gateway service is unable to log on to the system, as shown in Figure 5-4, make sure the Informix Admin group has the appropriate user rights policy at the operating system level. The user rights policy can be set with the User Manager option found in the Control Panel. For additional information, please refer to the *EGM machine notes* found under \$INFORMIXDIR/release.

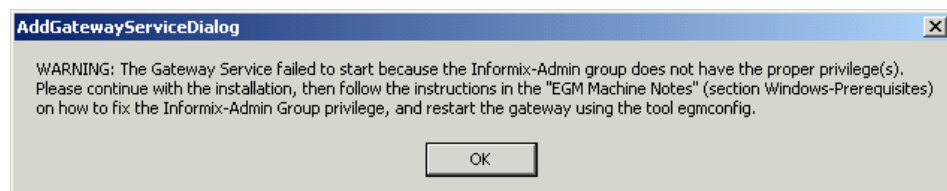


Figure 5-4 Error received trying to start the EGM service during installation

You will then be prompted for the destination folder and the setup type you want to have. The options are EGM Only, ODBC Only, and Typical. If you choose EGM only, no ODBC drivers will be installed with it. Informix distributes ODBC Drivers for some common RDBMS systems, but you are free to use the ODBC Driver of your choice as long as it complies with ODBC Version 2.0 or later. A summary of

your current settings will be displayed before the installation continues, as shown in Figure 5-5.

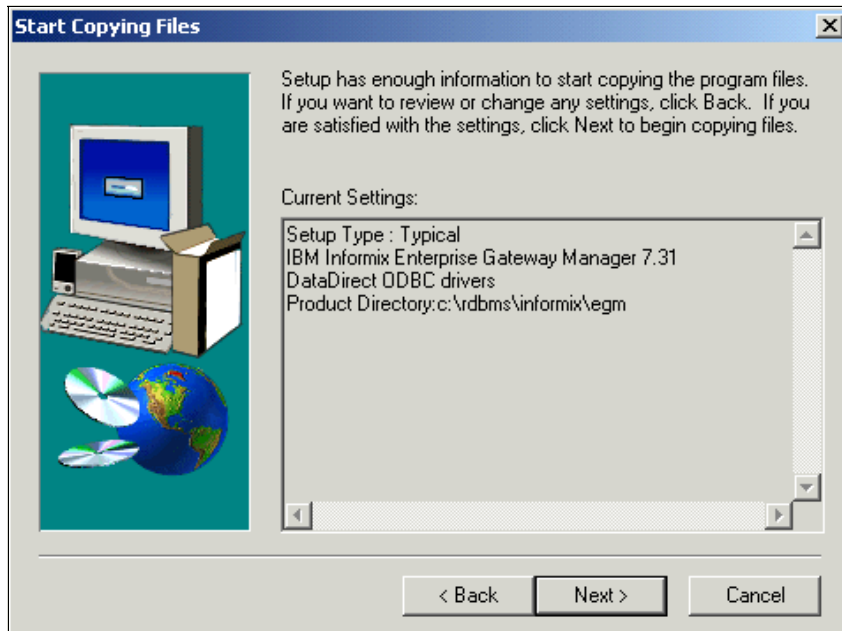


Figure 5-5 Current settings window

The next step in the installation process is to verify the machine name so the Enterprise Gateway service registration can be done appropriately. You will then be prompted for the EGM service name, as in Figure 5-6 on page 142. The EGM service is used to listen for processing requests and will be associated with a Windows service that will be started at the operating system level. It is going to be seen as the INFORMIXSERVER name in your Informix environment by which the EGM will be known.

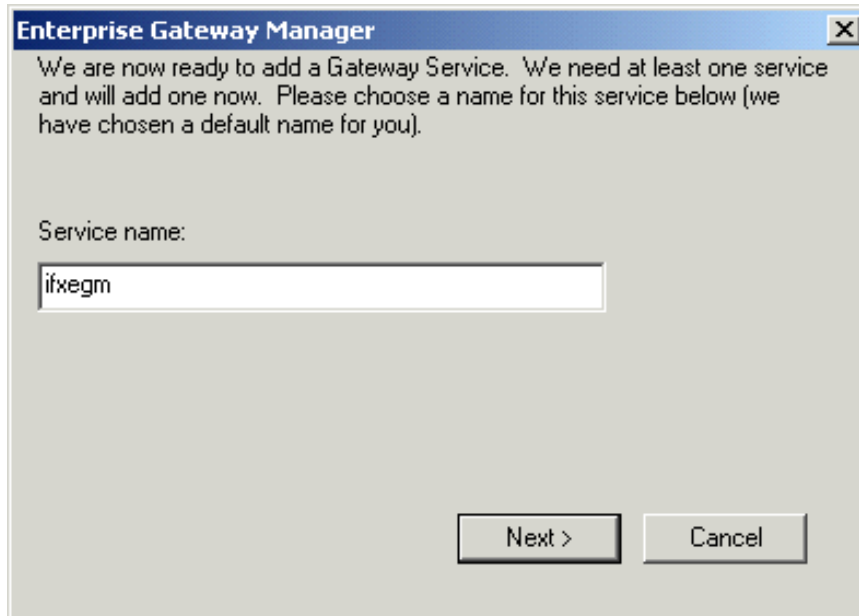


Figure 5-6 EGM service name

When the installation process is complete you must update the `egmenv.cmd` file under `$INFORMIXDIR`. This file is used to set the environment variables needed for the command-line utilities. Set the variables you want to use by supplying the appropriate values after consulting the documentation. You will at least need the variables shown in Table 5-1.

Table 5-1 Environment variables needed for EGM command line utilities

INFORMIXDIR	Full path to the installation directory
INFORMIXSERVER	Enterprise gateway service name
PATH	%INFORMIXDIR\$/bin

5.2.3 Configuration tasks

The EGM configuration steps include defining the user mapping table; setting up environment variables for the gateway service, ODBC, and Informix client; starting the gateway service; testing the ODBC access with EGM; and configuring Informix for use with EGM. Most of these activities are performed using the `egmconfig` and `egmdba` utilities. Below we describe these steps in detail.

1. Define the gateway service environment variable settings including EGM, ODBC, and Informix environment variables.

2. Start the gateway service.
3. Define user mapping.
4. Test the Gateway-to-ODBC data source connection.
5. Configure Informix for use with EGM.

Defining the gateway user ID mapping is where you associate a gateway user ID to a user ID in the target DBMS that will have permission to access the remote table. Use the **egmdba** utility with the **-ua** option to set up the user map table. See “User mapping and authentication” on page 143 for additional details.

Environment variable settings

The **egmconfig** utility with the **-setenv** option can be used to maintain environment variable settings for ODBC, Informix, and EGM. Here are some of the variable settings you will need to include:

```
egmconfig
  -setenv <Gateway Server Name> {<name> [<value>] | -f <filename>}
  -getenv <Gateway Server Name> [<name>] [-f <filename>]
  -unsetenv <Gateway Server Name> {<name> | -f <filename>}
```

When you set a new environment variable using the **egmconfig** utility, the setting will be associated with a particular gateway service and will take effect when the service is started. The **-f** option allows you to set your environment variables from a given file.

Starting the gateway service

You can start or stop the gateway service with the **egmconfig** utility or from the Windows Services dialog box. To start it from the **egmconfig** use the **-start** option as we show here:

```
egmconfig:
  -start <Gateway Server Name>
  -stop <Gateway Server Name>
```

For example, starting the gateway with the server name previously defined would be issued with the following command:

```
egmconfig -start ifxeg
```

User mapping and authentication

To access the target DBMS, you will need a user ID and password that the target DBMS recognizes. The user IDs that are valid on the gateway computer might not be valid target DBMS user IDs. The **egmdba** utility allows you to define a user mapping table so that the user connected to Informix can be mapped to a user ID in the target data source in order to gain access to the database.

The user ID that is sent to EGM from an Informix database server is known as the *effective* user ID of the Informix coordinator. The effective user ID is the one, for example, that *Informix OnLine* uses to connect to the EGM. Here are some important points to consider:

- ▶ The user that connects to the Informix server must have a login account in the gateway server machine. A trust relationship between the two machines must exist for this user account, thus users can connect to the gateway server machine without the need to provide a password. You can establish a trust relationship by editing the hosts.equiv file. This does not apply to a configuration where you have both the Informix server to which the application is connected and the gateway server on the same machine.
- ▶ Use **egmdba -ua** to add the user mapping to the table. To do this, edit a text file and include the following information: gateway_userId, data_source_name, data_source_userId, and data_source_userIDPassword. Then run **egmdba -ua** to process the rows in the file into the user map table. Example 5-1 shows file userlist.txt being processed by **egmdba -ua**. A log file is produced for this operation. **egmdba -uv** can then be used to verify the current mapping table contents. Notice that passwords are not displayed. Once the user mapping table is populated, you no longer need the text file. The same strategy can be used to delete users from the mapping table. The command to execute is **egmdba -ud**, providing a text file with the list of users to be removed.

Example 5-1 Adding users to the user map table

```
C:\rdbs\informix\egm\usermap>type userlist.txt
endusr01      db2odbc1      db2dba1      itsosj
endusr01      db2odbcw      db2dbaw      itsosj
endusr01      oraodbc1      oradba1      itsosj
endusr01      oraodbcw      oradbaw      itsosj
endusr01      rbwodbc1      rbwdba1      itsosj

C:\rdbs\informix\egm\usermap>egmdba -ua userlist.txt
Error/warning messages logged in c:\rdbs\informix\egm\infxtmp\egmdba_err.log

C:\rdbs\informix\egm\usermap>egmdba -uv
#Gateway_Userid Data_Source_Name Data_Source_Userid
endusr01      db2odbc1      db2dba1
      endusr01      db2odbcw      db2dbaw
      endusr01      oraodbc1      oradba1
      endusr01      oraodbcw      oradbaw
      endusr01      rbwodbc1      rbwdba1
```

Tip: The user map table is kept in RSAM files under \$INFORMIXDIR/egm/sysinfo. If you have trouble getting users authenticated, check the consistency of these files by running the **bcheckegm** utility. This utility will check the consistency of data and index files that hold user mapping information. Please consult the Informix Enterprise Gateway Manager User Manual for additional information. It can be found at:

<http://www-.ibm.com/software/data/informix/pubs/library>

Figure 5-7 on page 146 illustrates user *toddl* connecting to an Informix server and selecting from the table *customer* residing on a Red Brick database:

1. The client application connects to the Informix server as user *toddl* and issues the SQL statement `select * from rdbwodbcl@ifxegm:"RBWDBAL".customer.`
2. The Informix server identifies the query as a remote query and passes the request over to the gateway server *ifxegm*.
3. EGM searches for user *toddl* in DSN *rbwodbcl* and finds a corresponding user in the target database (user *willc* with password *willpass*).
4. EGM passes the new user information into the DSN ODBC driver for the connection to be established with the Red Brick server.
5. User *willc* must have been granted permissions by the table owner or the Red Brick DBA to **select** from the CUSTOMER table.

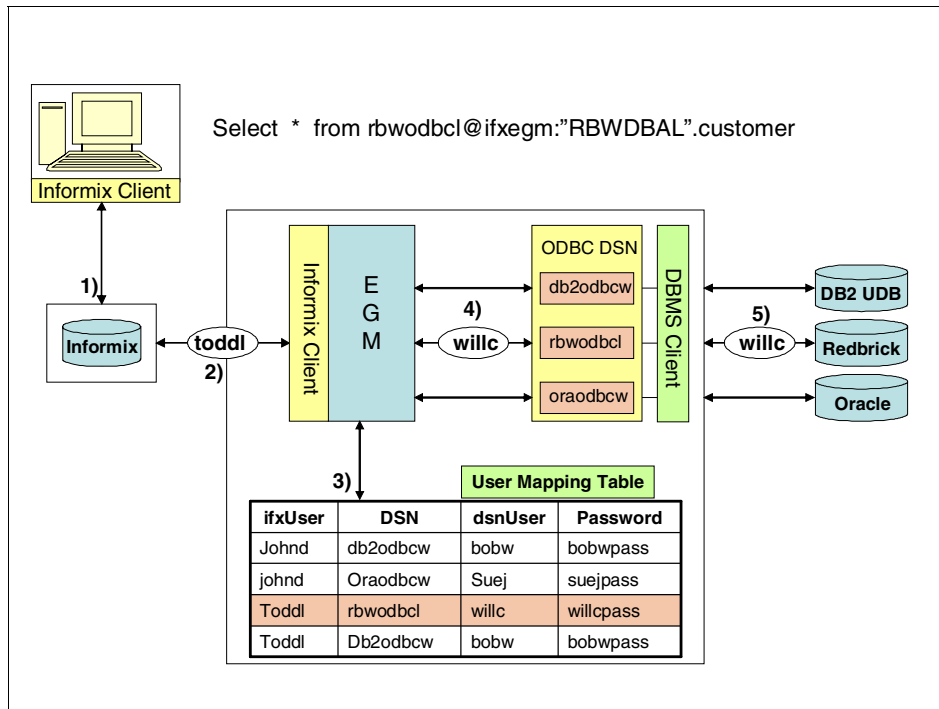


Figure 5-7 User mapping illustration

Testing the gateway to ODBC data source connection

The `egmdba` utility with the `-ts` option can be used to test an ODBC connection from the EGM server. This is the best way to test the connection and authentication between the ODBC data source and the target DBMS. When you test the data source, the user mapping will take place and a connection will be established to the DBMS, selecting from the table given at the command line. The usage is shown here:

egmdba:

```
-ts <data_source_name> <gateway_server_name> <owner> <table>
```

Suppose you are connected as user `enduser01` and you want to test the connection to a data source called `db2odbcw` selecting from table `CUSTOMER` owned by Db2 user `DB2DBAW`. This is depicted in Figure 5-8 on page 147

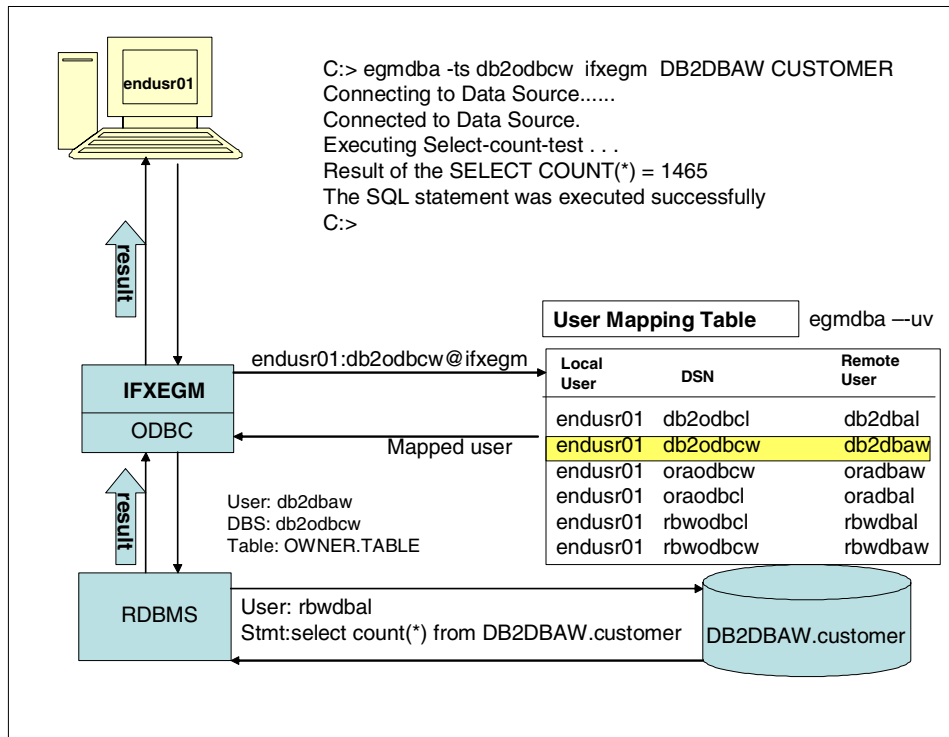


Figure 5-8 Testing the ODBC connection using egmdba

Tip: First make sure that *enduser01* is mapped to a valid user in DB2 that has **select** permission on the table *customer*. You can view the user mapping table by executing **egmdba -uv**. (See “User mapping and authentication” on page 143 for additional details.)

Configuring Informix for use with EGM

When you have an EGM server installed, it is seen as another Informix server in your environment. Other instances of Informix will refer to the EGM by its *server name*. Therefore you must configure the SQLHOSTS to include the EGM server name in the list of server definitions. In a Windows environment, you can use the **setnet32.exe** utility to add a new Informix server definition. On Unix platforms, you can manually edit the SQLHOSTS file. The parameters shown in Table 5-2 are need to add the EGM to the list of Informix database servers.

Table 5-2 SQLHOSTS parameters

dbservername	The gateway server name associated with the gateway service.
--------------	--

nettype/protocol	onsoctcp or ontlitcp; check the release notes or supported protocols.
hostname	The host name of the gateway computer.
servicename	The TCP/IP service name on the Informix server computer that maps to the TCP/IP port number that the gateway service on the gateway computer uses.

Tip: You can use the command `egmconfig -view` to get a list of EGM servers and port numbers. Here is the command, and the results you will see:

```
C:>egmconfig -view
```

```
Gateway Service Name: INFORMIX EGM ifxegm
Gateway Server Name: ifxegm
TCP/IP Port Number: 1550
Log File Name: C:\rdbms\informix\egm\egm\log\ifxegm.slog
Environment Variables:
    GWDEBUG 121.122.123
```

If you have Informix servers in another computer, you must include an entry in your HOSTS.EQUIV file on the gateway computer so that you may have a trust relationship with the security subsystem between the systems. Informix distributed database systems require that all subordinate servers trust the Informix server coordinator so that you do not need to provide the user password for authentication.

Once the gateway server definition is complete across the Informix servers you want to have participating in the distributed environment, you can test a remote query using the notation `datasource_name@gateway_server:"OWNER".table` in the FROM clause of the SELECT statement. We show an example of this in Figure 5-9 on page 149.

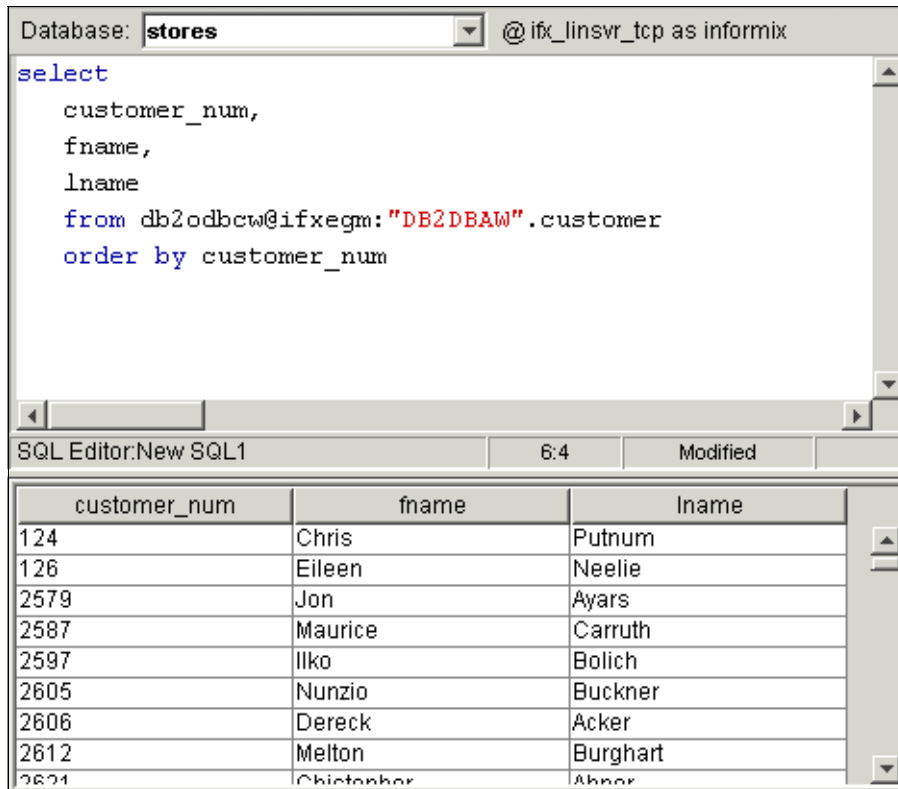


Figure 5-9 Testing the remote query

Tip: You can hide the fully qualified name by either creating a SYNONYM or a VIEW. See “Using views versus synonyms” on page 150.

5.3 Considerations for use

In this section we discuss some things to be considered when using specific functions, features, applications, and environments.

5.3.1 Using non-Informix SQL statements

If you have applications that use non-Informix SQL statements and want to use these applications with EGM, you must remove the non-Informix SQL statements from the application. If the application must control how SQL statements are issued, you may want to consider putting these statements into a stored

procedure on the data source and executing the data-source stored procedure using an Informix SQL dialect from the application. Alternatively, from your application you may want to consider invoking a data source client application.

Another approach is to consider the use of the EXECUTE IMMEDIATE or the PREPARE and EXECUTE statements, which allow you to send data source specific dialect, bypassing the Informix query compiler.

5.3.2 Object names in queries

The Informix database server coordinator uses the *server name* value to connect to the server (in this case, the EGM). The EGM then uses the *database value* to connect to the data source. Therefore, the data source will receive a two-part object name *OWNER.object*. The coordinator sends the values for OWNER.object and does not change the *owner* value specified by the user. The following rules are used to construct the owner value sent to the EGM:

- ▶ If owner is specified in quotes in the original statement, the owner is sent as specified.
- ▶ If owner is specified, but not specified in quotes, the following conditions apply:
 - If the coordinator is an ANSI database, the original owner value is shifted to uppercase and is sent as a quoted value.
 - If the coordinator is a non-ANSI database, the original owner value is shifted to lowercase and is sent as quoted value.

5.3.3 Using views versus synonyms

This topic is discussed in more detail in 8.6, “Using views versus synonyms with data federation” on page 220. In our testing environment we had initially decided to declare the remote objects as *synonyms*. For example:

```
CREATE SYNONYM db2w_customer FOR db2odbcw@stores:"DB2DBAW".customer
```

However, client tools, such as Brio Explorer, need the column definition to allow you to draw the queries based on a GUI interface, dragging and dropping columns to establish relationships and form the queries. That is not possible if you use synonyms since the column definition is not present in the database catalog where the synonym resides.

To get around this issue, we have decided to create views instead. When a view is created based on a data source, the column definition is imported into the local database catalog to describe the view based on the **select** list, thus making it possible for such tools to operate as designed. For example:


```
CREATE VIEW db2w_customer
AS SELECT * FROM db2odbcw@stores:"DB2DBAW".customer
```

5.3.4 Using Informix 4GL

Here are a few considerations when using Informix 4GL applications for accessing data sources through EGM: Depending on the 4GL statements used, 4GL applications cannot access tables with uppercase table names without creating lowercase wrapper synonyms or views based on the data source table. This is necessary, for example, with Oracle, which uses by default uppercase table name identifiers. You can create views on an Informix database based on the remote data-source table to make it easier for 4GL applications to manipulate the data referring to the table names.

5.3.5 Distributed transactions

In an *multi-site update* transaction, you can have as many Informix database server participants as you want, but only one EGM data source. On a single-site update scenario, updates can always be committed either if the participant is an Informix database server or an EGM data source. You must set the HETERO_COMMIT parameter in your ONCONFIG file for your Informix database server in order to turn on multi-site update capabilities.

5.3.6 Multiple data sources within a single application

Within a single application, each data source must be accessed through a separate gateway service using a unique gateway server name. If this rule is violated sqlcode -29052 is returned.

Within a single user connection (application) accessing EGM, a unique EGM server is required for each data source that the connection accesses. The following query would fail if only one gateway service was defined:

```
SELECT * from db2odbcw@ifxegm:"DB2DBAW".customer
UNION
SELECT * from oraodbcw@ifxegm:"ORADBWA".customer
```

In order to execute this query, you must add a second EGM server using **egmconfig -add** and then have each data source serviced by a unique gateway server.

5.3.7 Effect of accuracy of statistical information

Much of the query optimization that can be applied in distributed queries depends on the correctness of table statistics that are available from the system

catalog information on a server. Ensure that table-statistics information on both the target DBMS servers and the Informix database servers is kept up-to-date for the tables that the EGM gateway application uses.

5.3.8 Effect of Informix style system catalog

Before it executes a distributed query, the Informix database server requests catalog information about objects in the query from the EGM. EGM in turn collects this information from the target DBMS. The Informix optimizer uses the collected information to select the most efficient way to process the query.

By default, EGM uses the ODBC *catalog-information* function calls to obtain this information for the Informix coordinator, which includes such things as column, procedure, and statistics about index descriptions and number of rows.

Querying the ODBC catalog is more expensive than collecting the information needed from an Informix style system catalog. This is because the ODBC function calls collect much more information than what Informix needs, and often performs complex joins of the target DBMS system-catalog tables to obtain the information. When an Informix-like system catalog is created in the target DBMS, only the information required by Informix will be collected and no expensive operations are needed. As a result, you should experience better performance.

However, if you choose to create an Informix style system catalog in the foreign database system, you must consider the fact that these tables are static and must be refreshed periodically, thus requiring some additional maintenance.

Note: If the Informix style catalog is not installed, it is recommended that you set GWCATALOG to control the source and amount of catalog information that EGM collects about objects in distributed queries. Refer to the EGM User Manual for additional information on how to use GWCATALOG.

5.3.9 Oracle date and datetime conversion

The DATE data type in an Oracle database has both date and time parts. In the native SQL used by the Oracle database, the time portion of a date column is not returned unless a query specifically requests it. However, because the Oracle ODBC Driver maps the DATE data type to the SQL_TIMESTAMP data type, the driver always returns both the date and time portions of a date column.

All date-oriented data coming from an Oracle table is passed in a datetime format and received by Informix in the form of 4YMMDD + any fraction. A feature request (157350 FEA) has been created to change the value returned in the MDY4 format. In addition, an explicit casting must take place on an UNION

between an Oracle table and an Informix table involving the DATE data type. For example, if you must UNION table ORDERS from Oracle and Informix you would write your query as shown in Example 5-2. If you do not apply the DATE function, error -308 Corresponding column types must be compatible for each UNION statement is returned.

Example 5-2 Union of Oracle and Informix date columns

```
SELECT order_num, DATE(order_date), customer_num, ship_instruct, backlog,  
po_num, DATE(ship_date), ship_weight, ship_charge, DATE(paid_date) from  
oraodbcw@ifxegmoraw  
UNION  
SELECT * from ifxl_orders
```

5.3.10 Using cursors

Scroll cursors are allowed with EGM. Cursors declared with the WITH HOLD keywords are supported only if the ODBC Driver specified that cursors remain open after a commit operation. Specifically, if the ODBC function SQLGetInfo invocation for the SQL_CURSOR_COMMIT_BEHAVIOUR option returns SQL_CB_PRESERVE, then cursors can be declared WITH HOLD. Update cursors are also allowed only if the ODBC Driver supports this feature. Error -29030 is returned otherwise.

5.3.11 Temporary files for cursor handling

Statements that include scroll cursors function the same way using EGM as they do using an Informix database server. EGM creates temporary files in the directory indicated by the DBTEMP variable in the client environment, even if DBTEMP is set in the gateway service environment. Here are the rules:

- ▶ If DBTEMP is set in the gateway service environment but is not set in the client environment, the gateway service value is used.
- ▶ If DBTEMP is not set in either environment, the gateway service creates temporary files in the %INFORMIXDIR\ifxtmp directory.

5.3.12 Limiting the number of rows returned in a query

You can set the GWMAXROWS environment variable to limit the number of rows a user can select from remote tables. When the limit specified in GWMAXROWS is reached, the subsequent FETCH will return error -2909, and the cursor is implicitly closed.

5.4 Troubleshooting

With so many components involved in a federated environment, diagnosing a problem when trying to access a remote database may become a challenge. Here are some things to consider in the troubleshooting process:

- ▶ Make sure you can access the remote database target from the outside of EGM by connecting from a simple application via ODBC. If you cannot connect to your data source, focus on the client/server. You must be able to connect to your data source successfully.
- ▶ If you can connect to your DSN from outside of EGM, try now to connect using **egmdba -ts**, as described in “Testing the gateway to ODBC data source connection” on page 146.
- ▶ Make sure the gateway service is started. You can configure a gateway service to write to a log file the errors that it encounters and a record of its operations. See Example 5-3. By default, the log is located at `%INFORMIXDIR%\EGM\LOG\gateway_servername.slog`. You can verify the location of the log file on your installation by using the `-view` option of the **egmconfig** utility.

Example 5-3 Enterprise Gateway service log

```
2003-07-01 09:02:44.000000 Starting Daemon INFORMIX-SQL Version 7.31.T D1
Command Line = sqlxecd ifxegmoraw -s egm -l
C:\rdbms\informix\egm\egm\log\ifxeg
moraw.slog
INFORMIXDIR = C:\rdbms\informix\egm
2003-07-01 09:02:52.000000 daemon err = -25555: oserr = 0: errstr = ifxegmoraw:
Server ifxegmoraw is not listed as a dbserver name in
sqlhosts.
```

- ▶ The gateway system log, as shown in Example 5-4, is also helpful while debugging connection and post-connection problems. It shows the errors reported by EGM and ODBC. The log can be found under `%INFORMIXDIR%\EGM\log\egm.log`.

Example 5-4 Enterprise Gateway system log

```
2003-07-01 18:28:57.000000 PID= 1896 Client User informix (Gateway Process User
informix) disconnected from Data Source db2odbcw
2003-07-01 18:28:57.000000 PID= 1896
Driver Name (first 5 characters): DB2CL
  Driver Version: 08.01.0000
  Target DBMS Name (first 5 characters): DB2/N
  Target DBMS Version: 08.01.0002
SQLError reported no error!
```

- ▶ Inspect the DBMS message/error log files for any errors reported at the remote server.
- ▶ You can also turn the *tracing facility* on for a particular EGM server. When tracing is turned on, it can report SQL statements, error message, input and output data, and ODBC function calls, on a per-user application basis based on a mask that you set for the GWDEBUG variable. To turn the tracing facility on, use the `-setenv` option of the **egmconfig** utility with the appropriate value for the desired traces before you start the gateway service. See Example 5-5. There are three levels of tracing you can set in GWDEBUG. They can be used in combination for more detailed information according to the level of detail you want to trace. The levels are:
 - 121: Enable the EGM to trace SQL statements received by the gateway, SQL statements after modification by the gateway, SQL statements after modification by the ODBC Driver (if the driver supports the `SQLNativeSql` function), `SQLCA` information returned for SQL statements, and all error and warning messages received from the ODBC API.
 - 122: Enable the EGM to trace input `SQLDA` (input host-variable data types and values) and output `SQLDA` (output field-data types and values) information.
 - 123: Enable the EGM to trace ODBC function calls and the input and output parameters of each ODBC function.

Example 5-5 Turning on the EGM tracing facility

```

C:\rdbms\informix\egm>egmconfig -setenv ifxegmdb2w GWDEBUG 121,122,123
C:\rdbms\informix\egm>egmconfig -stop ifxegmdb2w
C:\rdbms\informix\egm>egmconfig -start ifxegmdb2w
C:\rdbms\informix\egm>egmconfig -view ifxegmdb2w
Gateway Service Name: INFORMIX EGM ifxegmdb2w
Gateway Server Name: ifxegmdb2w
TCP/IP Port Number: 1556
Log File Name: C:\rdbms\informix\egm\egm\log\ifxegmdb2w.slog
Environment Variables:
GWDEBUG 121,122,123

```

- ▶ The tracing file is created in the directory specified by `DBTEMP`. If `DBTEMP` is not set, EGM will create the trace file under `%INFORMIXDIR%\infxtmp`. The file name is formed by `userName` and `ProcessId`. We show a sample portion of trace output in Example 5-6.

Example 5-6 A portion of the trace output

```

===== Time: 2003-07-02 18:44:52.000000 =====
** Gateway Product ID, Trace Levels, and Environment Variable Settings **
Gateway Product ID: IBM Informix Enterprise Gateway Manager Version 7.31.TD1
Gateway Build Timestamp: Apr 15 11:44:51 2003

```

```

Trace level(s) selected: 121,122,123
INFORMIXSERVER: "ifxegmdb21"
  INFORMIXSQLHOSTS: "\\GTWSVR"
===== Time: 2003-07-02 18:44:52.000000 =====
Gateway PID: "1540", PPID: "0"
Gateway Login Directory: ""
Gateway Current Working Directory: "C:\rdbms\informix\egm\etc"
Gateway UID: endusr01(7), EUID: endusr01(6)
Gateway GID: Group Not Found(5373952), EGID: Group Not Found(0)
===== Time: 2003-07-02 18:44:53.000000 =====
** After SQLAllocConnect **
Return Code: "SQL_SUCCESS"
  Raw Return Code: 0
  Address of allocated hdbc: 0x006F1348
===== Time: 2003-07-02 18:44:53.000000 =====
** Before SQLSetConnectOption **
Connect Option To Set: "SQL_ODBC_CURSORS"
Option Value To Set: "SQL_CUR_USE_DRIVER"
Address of hdbc: 0x006F1348
fOption: 110
vParam: 2
===== Time: 2003-07-02 18:44:53.000000 =====
** After SQLSetConnectOption **
Return Code: "SQL_SUCCESS"
Raw Return Code: 0
===== Time: 2003-07-02 18:44:53.000000 =====
** Before SQLConnect **
Address of hdbc: 0x006F1348
szDSN: "db2odbcw"
cbDSN: 8
szUID: "db2dbaw"
cbUID: 7
===== Time: 2003-07-02 18:44:53.000000 =====
** After SQLConnect **
Return Code: "SQL_SUCCESS"
Raw Return Code: 0
===== Time: 2003-07-02 18:44:53.000000 =====
** Before SQLSetConnectOption **
Address of hdbc: 0x006F1348
fOption: 1041
===== Time: 2003-07-02 18:44:53.000000 =====
** After SQLSetConnectOption **
Return Code: "SQL_ERROR"
Raw Return Code: -1
===== Time: 2003-07-02 18:44:53.000000 =====
** Before SQLError **
Address of henv: 0x006F12A0
Address of hdbc: 0x006F1348
Address of hstmt: 0x00000000

```

Address of szSqlState: 0x0012E814
Address of pfNativeError: 0x0012E804
Address of szErrorMsg: 0x0012E5F0
cbErrorMsgMax: 511
Address of pcbErrorMsg: 0x0012E808



Data federation in action

In this chapter we show you the results obtained when we ran a set of sample queries against the federated databases using both products, DB2 Information Integrator and Informix Enterprise Gateway Manager. We describe the queries and the query plans produced at the federated database servers and how the data was retrieved by the application, demonstrating some of the features and capabilities of the federation products, such as *pushdown* and *query rewrite*.

6.1 Test application examples

To fully illustrate the power of data federation technology, we constructed a set of sample queries. These queries were designed to be realistic from a business sense and also to somewhat challenge the federated databases. In some cases, our federated query accessed 28 different tables, across seven different data sources.

We executed essentially the same queries using both data federation products, DB2 Information Integrator and Informix EGM, in order to get a good comparison of the features and capabilities of each.

We also used Brio Explorer to generate the same queries graphically, using the Brio GUI, to prove that we could equally well use a graphical Business Intelligence (BI) tool in addition to native SQL.

6.1.1 Query 1 - Summarizing regional data

Our first sample query is known as Query 1- Summarizing regional data, and is shown in Example 6-1. The objective of this query is to find a set of orders that meets a given set of business criteria across all seven regions in our federated case study database. The query determines those orders after a given date, which contain at least five stock items from a given manufacturer. The information retrieved is the customer number and company name, the order number and revenue attributable from that order to that manufacturer.

To summarize data from these seven regions, seven subqueries, all with essentially the same SQL, are used and then combined with a series of **union** statements. This is a somewhat verbose query to develop, but we do it for the sake of example. In the subsequent query examples we will demonstrate how to improve upon this query.

Example 6-1 SQL for sample Query 1

```
select c.customer_num, c.company, o.order_num, sum (i.total_price) as
total_manu_rev
from db21_customer c, db21_orders o, db21_items i, db21_manufact m
where o.order_date > '12/25/2001' and m.manu_name = 'Stephani Inc'
and m.manu_code = i.manu_code and o.order_num = i.order_num
and o.customer_num = c.customer_num
group by c.customer_num, c.company, o.order_num
having count (*) >= 5

union all
```

```

select c.customer_num, c.company, o.order_num, sum (i.total_price) as
total_manu_rev
from db2w_customer c, db2w_orders o, db2w_items i, db2w_manufact m
where o.order_date > '12/25/2001' and m.manu_name = 'Stephani Inc'
and m.manu_code = i.manu_code and o.order_num = i.order_num
and o.customer_num = c.customer_num
group by c.customer_num, c.company, o.order_num
having count (*) >= 5

```

union all

```

select c.customer_num, c.company, o.order_num, sum (i.total_price) as
total_manu_rev
from oraw_customer c, oraw_orders o, oraw_items i, oraw_manufact m
where o.order_date > '12/25/2001' and m.manu_name = 'Stephani Inc'
and m.manu_code = i.manu_code and o.order_num = i.order_num
and o.customer_num = c.customer_num
group by c.customer_num, c.company, o.order_num
having count (*) >= 5

```

union all

```

select c.customer_num, c.company, o.order_num, sum (i.total_price) as
total_manu_rev
from ifxl_customer c, ifxl_orders o, ifxl_items i, ifxl_manufact m
where o.order_date > '12/25/2001' and m.manu_name = 'Stephani Inc'
and m.manu_code = i.manu_code and o.order_num = i.order_num
and o.customer_num = c.customer_num
group by c.customer_num, c.company, o.order_num
having count (*) >= 5

```

union all

```

select c.customer_num, c.company, o.order_num, sum (i.total_price) as
total_manu_rev
from ifxw_customer c, ifxw_orders o, ifxw_items i, ifxw_manufact m
where o.order_date > '12/25/2001'
and m.manu_name = 'Stephani Inc' and m.manu_code = i.manu_code
and o.order_num = i.order_num and o.customer_num = c.customer_num
group by c.customer_num, c.company, o.order_num
having count (*) >= 5

```

union all

```

select c.customer_num, c.company, o.order_num, sum (i.total_price) as
total_manu_rev
from rbwl_customer c, rbwl_orders o, rbwl_items i, rbwl_manufact m
where o.order_date > '12/25/2001'
and m.manu_name = 'Stephani Inc' and m.manu_code = i.manu_code

```

```
and o.order_num = i.order_num and o.customer_num = c.customer_num
group by c.customer_num, c.company, o.order_num
having count (*) >= 5

union all

select c.customer_num, c.company, o.order_num, sum (i.total_price) as
total_manu_rev
from xpsl_customer c, xpsl_orders o, xpsl_items i, xpsl_manufact m
where o.order_date > '12/25/2001' and m.manu_name = 'Stephani Inc'
and m.manu_code = i.manu_code and o.order_num = i.order_num
and o.customer_num = c.customer_num
group by c.customer_num, c.company, o.order_num
having count (*) >= 5

order by 1;
```

The results from query 1 are shown in Figure 6-1 on page 163.

Command Center

Command Center Query Results Edit Tools Help

Command type
SQL statements and DB2 CLP commands

System

Interactive Script Results Access Plan

Perform required changes and then click the commit update button.

CUSTOMER_NUM	COMPANY	ORDER_NUM	TOTAL_MANU_REV
102	Sports Spot	1,845	19,655.35
106	Watson & Son	1,486	25,183.04
2585	Components Division	1,995	33,452.34
2599	Scott Integrations I	1,861	44,269.87
2609	Soho Development Gro	1,667	21,986.68
2638	Easy Street, Inc	1,622	23,131.30
2655	Intermallamerica.com	1,247	17,953.53
2665	Frim Where Minnesota	1,595	29,723.87
2668	America's Web Statio	1,724	31,262.67
2670	GERBER SCIENTIFIC IN	1,271	16,945.44
2672	Local Connect	1,331	21,537.00
2681	Infoserve Training C	1,516	34,510.41
2687	Ssds, Inc	1,424	21,252.48
2713	Internet 2	1,490	25,057.29
2719	Internet 2	1,381	26,498.75
2722	Symb, Inc	1,839	28,357.79
2725	Qsi	1,608	28,695.60
2728	Computer Technical S	1,676	27,772.17
2738	Devcomm Information	1,184	11,712.67
2738	Devcomm Information	1,752	30,085.58
2741	C M G and Associates	1,620	22,621.49
2745	Intermallamerica.com	1,838	15,045.29
2760	Creative Concepts Co	1,796	26,754.75

Next Rows in memory 100 [1 - 100] Automatically commit

Figure 6-1 Results for Query 1

6.1.2 Query 2 - Federated views

Our second sample query, Query 2, is based on the premise that we desire to isolate the end user as much as possible from the exact topology of the federated database. For example, in Query 1, the end user needs to know that there are seven regions that constitute the federated database. Furthermore he or she needs to code the entire query in seven subqueries with essentially the same SQL in each case. This responsibility on the end user is, at best, somewhat tedious. In the worst case, the need for all the repetition could be a source of errors. The end user may introduce errors as part of the process of replicating the same basic SQL construct to each of the seven sub-queries or if one of the seven data sources was omitted, the results set could be incomplete.

From the end user's perspective, a much simpler approach would be to create a series of federated views for each of the tables customer, orders, and items representing the total set of customers, orders, or items in each case. Each view would contain a UNION of all the data from each remote data source for a particular table. The required views are shown in Figure 6-2 on page 165. Now we can simplify Query 1 considerably. The simplified query version of Query 1 is known as Query 2, and is shown in Figure 6-3 on page 166.

```

CREATE VIEW DB2W_JOIN_CUSTOMER AS
SELECT * FROM DB2DBAW.DB2L_CUSTOMER
UNION ALL
SELECT * FROM DB2DBAW.DB2W_CUSTOMER
UNION ALL
SELECT * FROM DB2DBAW.ORAW_CUSTOMER
UNION ALL
SELECT * FROM DB2DBAW.IFXL_CUSTOMER
UNION ALL
SELECT * FROM DB2DBAW.IFXW_CUSTOMER
UNION ALL
SELECT * FROM DB2DBAW.RBWL_CUSTOMER
UNION ALL
SELECT * FROM DB2DBAW.XPSL_CUSTOMER;

CREATE VIEW DB2W_JOIN_ORDERS AS
SELECT * FROM DB2DBAW.DB2L_ORDERS
UNION ALL
SELECT * FROM DB2DBAW.DB2W_ORDERS
UNION ALL
SELECT * FROM DB2DBAW.ORAW_ORDERS
UNION ALL
SELECT * FROM DB2DBAW.IFXL_ORDERS
UNION ALL
SELECT * FROM DB2DBAW.IFXW_ORDERS
UNION ALL
SELECT * FROM DB2DBAW.RBWL_ORDERS
UNION ALL
SELECT * FROM DB2DBAW.XPSL_ORDERS;

CREATE VIEW DB2W_JOIN_ITEMS AS
SELECT * FROM DB2DBAW.DB2L_ITEMS
UNION ALL
SELECT * FROM DB2DBAW.DB2W_ITEMS
UNION ALL
SELECT * FROM DB2DBAW.ORAW_ITEMS
UNION ALL
SELECT * FROM DB2DBAW.IFXL_ITEMS
UNION ALL
SELECT * FROM DB2DBAW.IFXW_ITEMS
UNION ALL
SELECT * FROM DB2DBAW.RBWL_ITEMS
UNION ALL
SELECT * FROM DB2DBAW.XPSL_ITEMS;

```

Figure 6-2 Federated views for the customer, orders, and items tables

```
select c.customer_num, c.company, o.order_num, sum (i.total_price) as
total_manu_rev
from db2w_join_customer c, db2w_join_orders o, db2w_join_items i,
db2w_manufact m
where o.order_date > '12/25/2001'
and m.manu_name = 'Stephani Inc'
and m.manu_code = i.manu_code
and o.order_num = i.order_num
and o.customer_num = c.customer_num
group by c.customer_num, c.company, o.order_num
having count (*) >= 5
order by 1
```

Figure 6-3 SQL for sample Query 2

The use of the federated views has now hidden all the complexity of the data federation from the end user. From the end user's perspective there is only one logical table (view) for each of the customer, orders, and items tables.

Figure 6-4 on page 167 shows the results from running Query 2. The results are the same as those for Query 1.

Note: In fact Query 2 is not entirely functionally identical to Query 1, but as a result of the data distribution in our case study, it *will* produce the same results. If it were possible that a given set of values in the grouping columns (customer_num, company, order_num) existed in more than one remote source, Query 2 could produce more rows. This is because in Query 2 the HAVING clause is evaluated across all data sources rather than individually for each data source as it is in Query 1. In our case study, a given customer record is present in *one* data source only and so the two queries will produce the same result.

CUSTOMER_NUM	COMPANY	ORDER_NUM	TOTAL_MANU_REV
102	Sports Spot	1,845	19,655.35
106	Watson & Son	1,486	25,183.04
2585	Components Division	1,995	33,452.34
2599	Scott Integrations I	1,861	44,269.87
2609	Soho Development Gro	1,667	21,986.68
2638	Easy Street, Inc	1,622	23,131.30
2655	Internallamerica.com	1,247	17,953.53
2665	Frim Where Minnesota	1,595	29,723.87
2668	America's Web Statio	1,724	31,262.67
2670	GERBER SCIENTIFIC IN	1,271	16,945.44
2672	Local Connect	1,331	21,537.00
2681	Infoserve Training C	1,516	34,510.41
2687	Ssds, Inc	1,424	21,252.48
2713	Internet 2	1,490	25,057.29
2719	Internet 2	1,381	26,498.75
2722	Symbx, Inc	1,839	28,357.79
2725	Qsi	1,608	28,695.60
2728	Computer Technical S	1,676	27,772.17
2738	Devcomm Information	1,184	11,712.67
2738	Devcomm Information	1,752	30,085.58
2741	C M G and Associates	1,620	22,621.49
2745	Internallamerica.com	1,838	15,045.29
2760	Creative Concepts Co	1,796	26,754.75
2766	Rational Consulting	1,296	21,037.39

Figure 6-4 Results for Query 2

Later on in “Scenario 2 with DB2 II” on page 179 we consider the performance implications of Query 2.

6.1.3 Query 3 - Consolidating the data

Our third and final sample query, Query 3, is intended to illustrate the power of the data federation technology to be able to update data in remote sources. Although currently there are limitations on the on the number of updates to remote sources, the power to update is still there in both of the data federation tools. For example, even if you decided to use the data consolidation, rather than the data federation, approach for your end-user queries or applications, you could still use data federation technology to move the data between the data

sources. For Query 3, we decided to create a new table in the Stores Demo schema. This table is to be used to consolidate data for customers from all regions into one physical table in one data source. In our example, we wish to implement the consolidated table in the data source with database identifier ifxl (Informix Dynamic Server on our Linux server linsvr).

The definition of the new table is shown in Figure 6-5.

```
CREATE TABLE corp_customer
( customer_num serial not null primary key,
  fname   char(15),      lname   char(15),
  company char(20),      address1 char(20),
  address2 char(20),    city    char(15),
  state   char(02),     zipcode char(05),
  phone   char(18)
);
```

Figure 6-5 SQL for the new corp_customer table

Sample Query 3 uses a federated insert to populate this new table from the federated view of all customers shown in Figure 6-2 on page 165. The SQL for Query 3 is shown in Figure 6-6.

```
insert into corp_customer select * from db2w_join_customer order by 1;
```

Figure 6-6 SQL for sample Query 3

Important: The **order by 1** statement is not typically required in this type of federated insert statement. It was only necessary for us as a workaround to the Informix insert problem discussed in 8.1, “Federated inserts with Informix” on page 218.

The results for Query 3 are shown in Figure 6-7 on page 169.

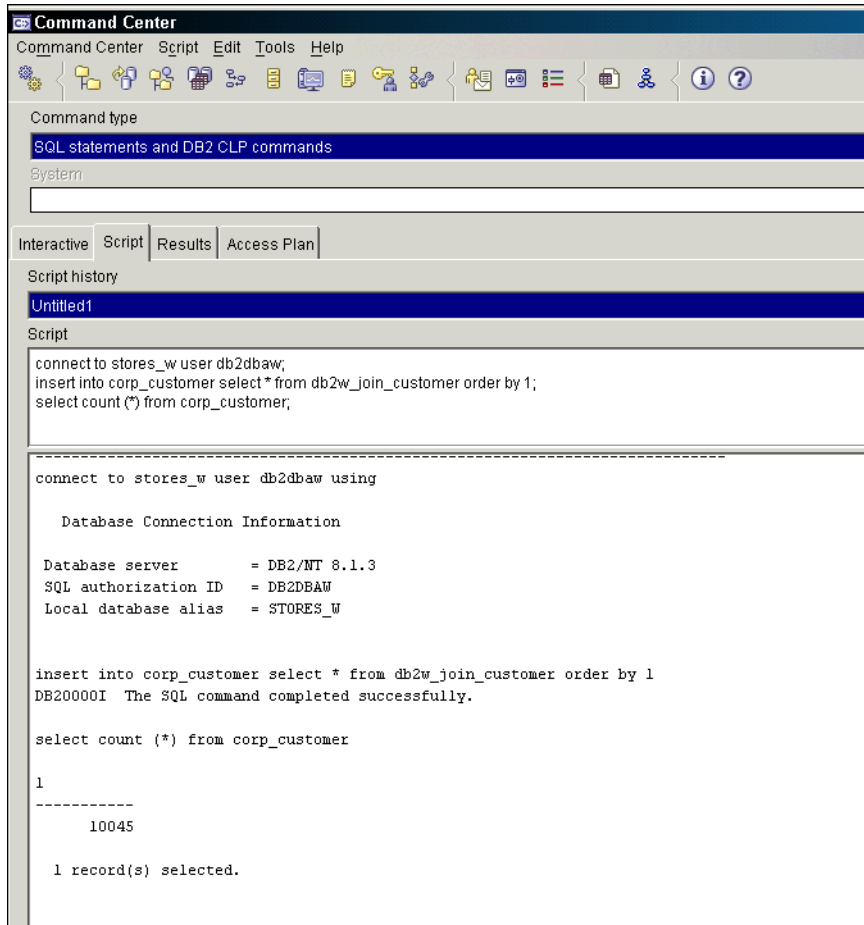


Figure 6-7 Results from Query 3

6.2 Testing with DB2 Information Integrator

In this section we illustrate the execution of the sample queries with DB2 Information Integrator. Each of the sample queries was executed using the DB2 Control Center. The Brio GUI was also used to create and execute equivalent SQL for Queries 1 and 2. We then used the DB2 Command Center interactive explain tool to analyze the access strategy chosen for the queries.

In some of the tests we found issues affecting performance of the sample queries, so in Chapter 7, “Optimization in a federated environment” on page 193,

we examine a number of techniques that we used to improve the performance of our sample queries.

6.2.1 Join and insert testing with DB2 II

In this section we discuss join and insert testing with DB2 II.

Scenario 1 with DB2 II

To understand this section, it is important to first acquaint yourself with the contents of sample Query 1 (see Example 6-1 on page 160).

We used the DB2 Command Center to execute Query 1. Using the Command Center menu option **Interactive -> Create Access Plan**, we produced the interactive explain graph for this query. This graph of the query plan for Query 1 appears at first to be quite complex (see Figure 6-8). In fact it is difficult to capture the entire query plan in a readable fashion on a single screen. To support large query plans the command center also creates an overview window with a blue *zoom box*, which you can position over your area of interest. When you do so, the contents of the zoom box are displayed in the main window of the control center.

At the overview level, we can discern that there are seven individual legs to the query plan, which correspond to the seven subqueries that constitute the *union* in Query 1. We used the zoom box to observe the individual parts of the query plan in a legible fashion.

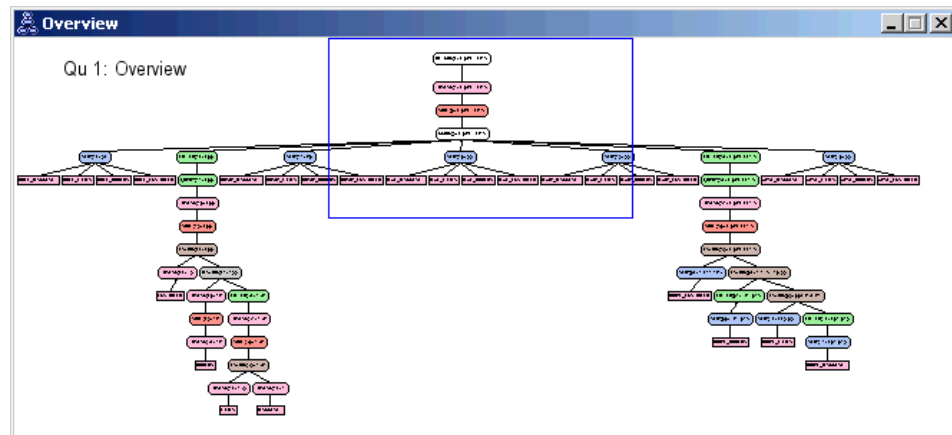


Figure 6-8 DB2 interactive explain graph overview for Query 1

Using the zoom box, we can also see that the sequence of the subqueries in our main union has been preserved in this query plan and this also helps us to work out what each part of the finely detailed overview graph represents.

If we position the zoom box over the top leg of the query plan graph, we can see the detail of the union. This is illustrated in Figure 6-9. We have shown only the detail of the union in Figure 6-9. The zoom box actually displays a larger area than this and we have excluded the extra information for the purpose of clarity.

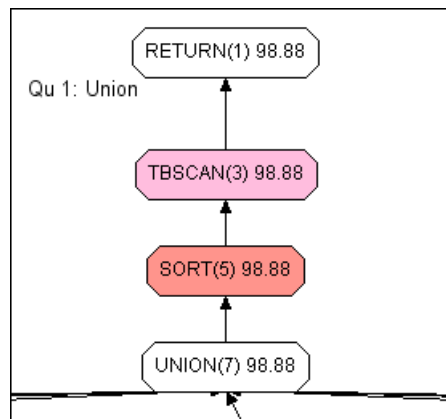


Figure 6-9 Union leg of the explain graph for Query 1

The UNION (operator 7) in Figure 6-9 collates the seven subquery legs. The subquery legs are shown in more detail in the figures below. Also in Figure 6-9 we have a SORT (operator 5) to facilitate the *order by* clause in Query 1. There is nothing really surprising here. Actually, we would have obtained the same result if we had examined the interactive explain graph of a *union* and *order by* of *local* DB2 tables.

Figure 6-10 on page 172 is much more interesting and shows how the query optimizer decided to process the first remote subquery, shown on the far left side of Figure 6-8 on page 170. That subquery retrieves data from the DB2 tables on our Linux server (called *linsvr*).

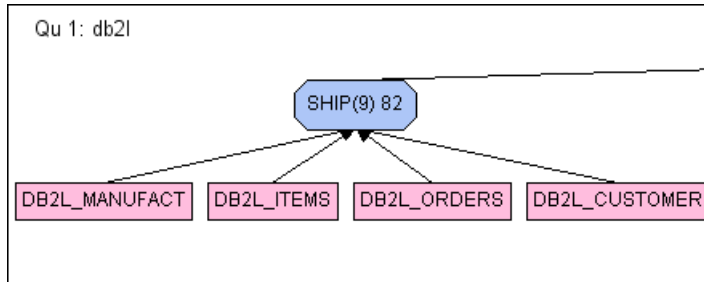


Figure 6-10 Pushed down operation from Query 1 for database identifier db2l

Note first the appearance of a new SHIP operator. This operator is responsible for transferring an interim result table from the remote DB2 data source to the local federated database. Secondly observe that there are no operations between the SHIP operator in Figure 6-10 and the UNION operator shown in Figure 6-9 on page 171. This can be determined from the level of detail shown in the overview graph in Figure 6-8 on page 170. Also observe that there are four tables connected directly to the SHIP operator in Figure 6-10. This is the representation of a pushdown join operation. The details of the filtering for the `order_date > '12/25/2001'` and `manu_name = 'Stephani Inc'` predicates are not shown in this query plan. Neither the GROUP BY operation, for the `sum(total price)`, nor the filtering operation, for the `having count (*) >= 5`, are visible in the query plan. All of this detail is missing from the query plan shown at the federated database. Why? The answer is that the query optimizer has chosen to push down these operations to the remote data source.

At first you may feel uncomfortable in interpreting this query plan by the *absence* of the detail concerning the predicates, the group by, and having operations; however, we can verify that we are interpreting the missing (pushed down) detail by right clicking the **SHIP** operator in Figure 6-10 and selecting the **Show Details** option in the interactive explain graph. This is illustrated in Figure 6-11 and Figure 6-12 on page 173.

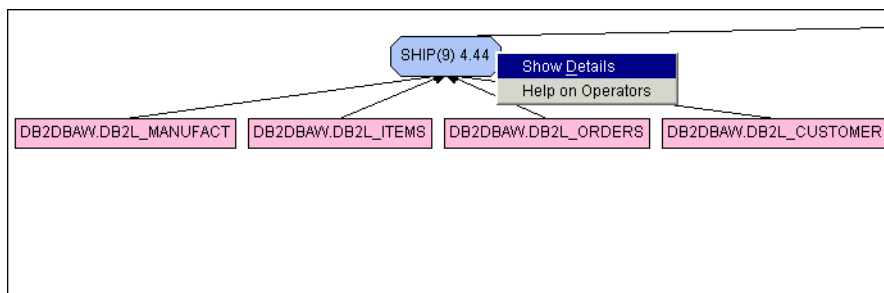


Figure 6-11 Right click the SHIP operator and select Show Details

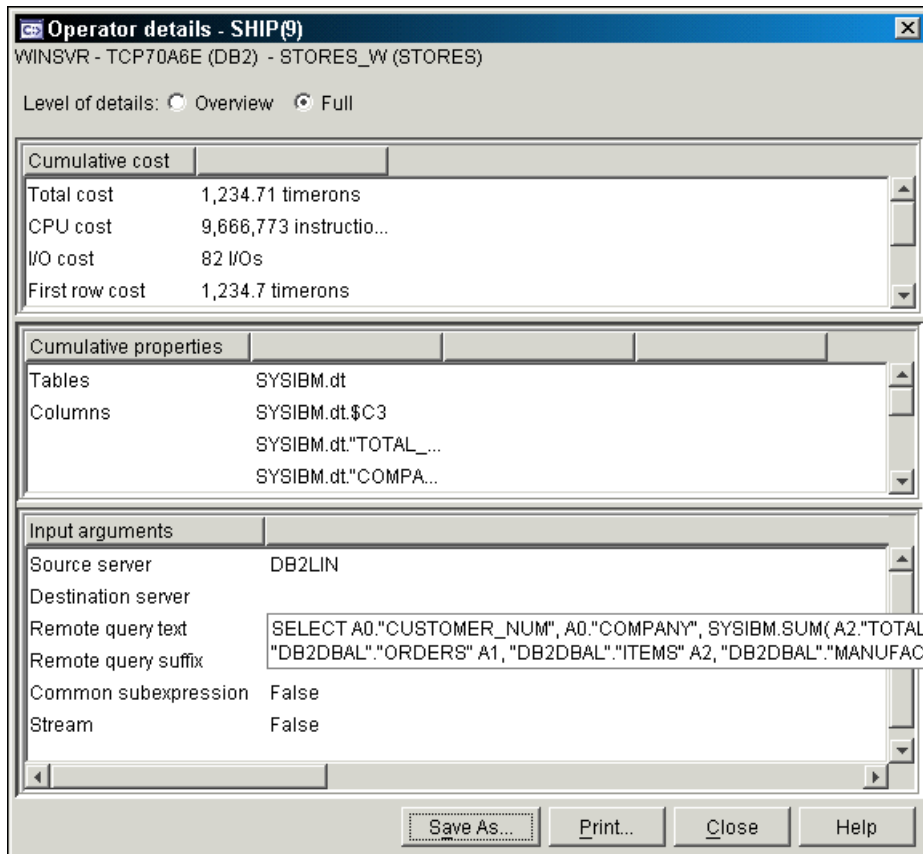


Figure 6-12 The remote query text for the SHIP operator is shown

The remote query text extends over the width of the screen shown in Figure 6-12 and can be truncated if quite long, so you can use the **Save As** button to capture the entire content of this window. From the saved file you can extract the remote query text as shown in Figure 6-13.

```
SELECT A0."CUSTOMER_NUM", A0."COMPANY", SYSIBM.SUM( A2."TOTAL_PRICE"),
A1."ORDER_NUM" FROM "DB2DBAL"."CUSTOMER" A0, "DB2DBAL"."ORDERS" A1,
"DB2DBAL"."ITEMS" A2, "DB2DBAL"."MANUFACT" A3 WHERE ('2001-12-25' <
A1."ORDER_DATE") AND (A3."MANU_NAME" = 'Stephani Inc ') AND
(A3."MANU_CODE" = A2."MANU_CODE") AND (A1."ORDER_NUM" = A2."ORDER_NUM") AND
(A1."CUSTOMER_NUM" = A0."CUSTOMER_NUM") GROUP BY A1."ORDER_NUM",
A0."COMPANY", A0."CUSTOMER_NUM" HAVING (5 <= COUNT(*)) FOR READ ONLY
```

Figure 6-13 Remote query text for the SHIP operator

Now we have official confirmation from the DB2 optimizer that the join, the filtering for the predicates, and the group by are being pushed down (or *shipped*) to the remote data source.

In general, the performance of a federated query will be best if the query fragments can be pushed down and executed at the remote data sources. As a general rule, the more a subquery is pushed down, the less data will be retrieved from the remote data source and sent to the federated server. One exception is when a cartesian style join is requested when the number of rows retrieved from a pushed down join could be close to the product of the cardinality of the remote tables involved. In our case in Figure 6-13 on page 173, only the qualifying rows from the pushed down query fragment are returned (shipped) to the federation server to take part in the main UNION statement in Query 1. This is the most efficient query plan we could hope for in this case.

Of course pushdown only occurs where the definition of the wrapper and the corresponding server options for the remote data source can support the required functionality. So within the scope of the operations that are candidates for pushdown, the DB2 optimizer makes its assessment of which operations within the federated query to push down based on the lowest anticipated cost. Later on in 7.1, “Performance options and considerations” on page 194, we will see that it is possible to override this cost-based decision.

Note: Now is probably a good time to reflect on how a naming convention for the nicknames in the federated database has facilitated easy analysis of the query plans for federated queries, once again validating the need for good and meaningful naming conventions.

Considering the rest of the query plan, it is fortunate that we can now simplify the analysis of the other legs in the interactive explain graph shown in Figure 6-8 on page 170. The other legs in this explain graph for the database identifiers oraw (third leg), ifxl (fourth leg), ifxw (fifth leg), and xpsl (seventh leg) are all equivalent to that for db2l with the joins, predicates, and filtering operations, and are all pushed down to the remote data sources. These parts of the explain graph are illustrated in Figure 6-14 on page 175, Figure 6-15 on page 175, Figure 6-16 on page 175, and Figure 6-17 on page 175, respectively.

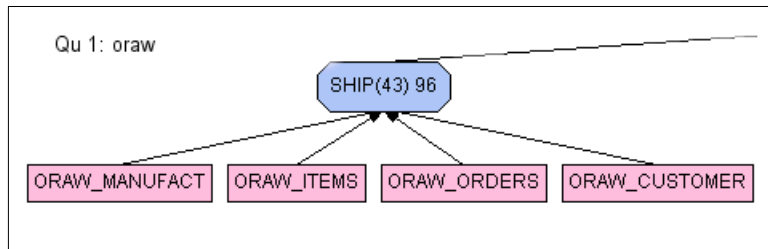


Figure 6-14 Pushed down operation from Query 1 for database identifier oraw

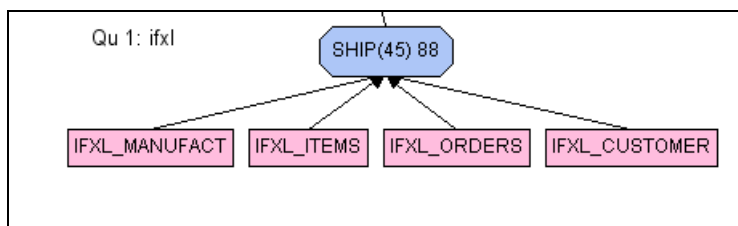


Figure 6-15 Pushed down operation from Query 1 for database identifier ifxl

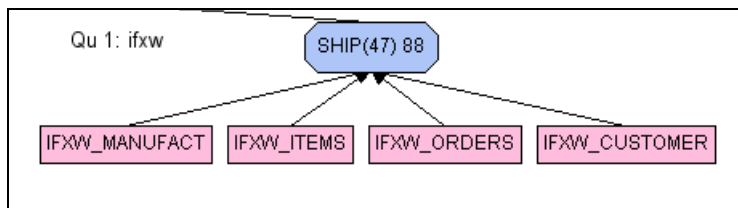


Figure 6-16 Pushed down operation from Query 1 for database identifier ifxw

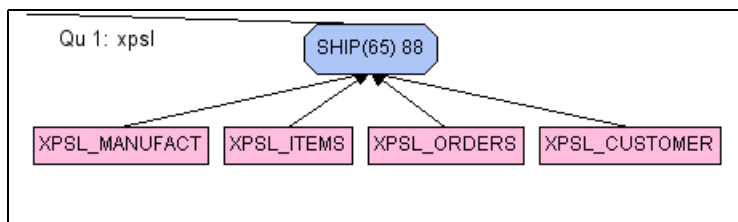


Figure 6-17 Pushed down operation from Query 1 for database identifier xpsl

We have now covered all but two remaining parts of the query plan graph for Query 1 shown in Figure 6-8 on page 170. The query plan graph for the local tables (database identifier db2w) is shown in Figure 6-18 on page 176.

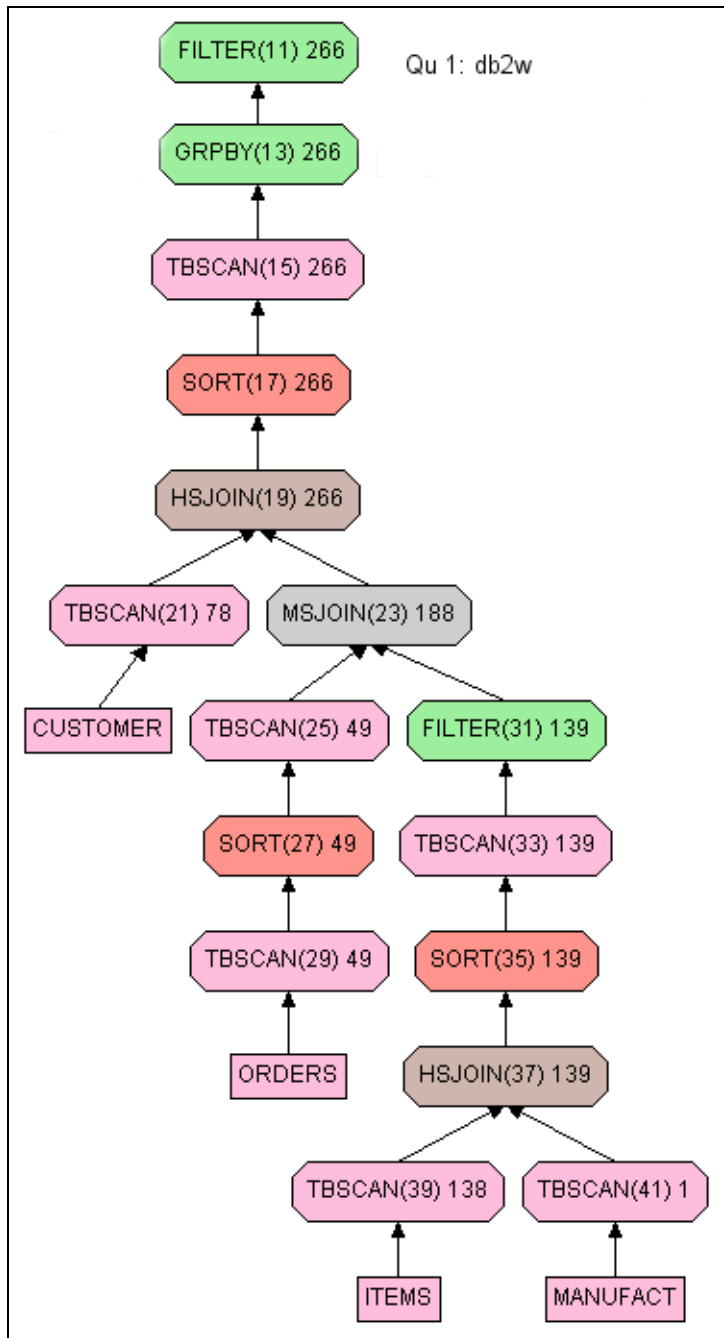


Figure 6-18 Query plan graph for Query 1 for database identifier db2w

This leg of the query plan looks different in two ways. First, there is much more detail shown. In fact, this detail only appears because this is the plan for the local data source. It would be possible to have this level of detail for each of the legs of the query plan using the applicable analysis tools at each of the remote data sources.

Secondly, note that the table names that appear in the explain graph are the real table names as per the schema shown in Figure 2-5 on page 40. We did define synonyms (such as `db2w_items` for the `items` table) to be able to use the same queries against the Informix tables, but these names have been translated to the real names by the optimizer.

We will not discuss further the access plan for the local tables, as this is not particularly relevant to our discussion of data federation. We only need to observe that the presence of the local tables in the explain graph for the entire federated query confirms the fact that we can integrate local and remote data sources easily using DB2 II.

The only remaining part of the interactive explain graph for Query 1, shown in Figure 6-8 on page 170, is the sixth leg in the overall query plan, which is the plan for the remote data source `rbwl`. This final part is shown in Figure 6-19 on page 178. Compare this with the query plan shown for Informix Dynamic Server in Figure 6-15 on page 175. Observe how different these two query plans are. For the `rbwl` data source, most of the work is being done by DB2 II after the remote data has been shipped. The join, the filtering for predicates, the group by, and the filtering for the *having* clause are all shown above the SHIP operators in Figure 6-19 on page 178.

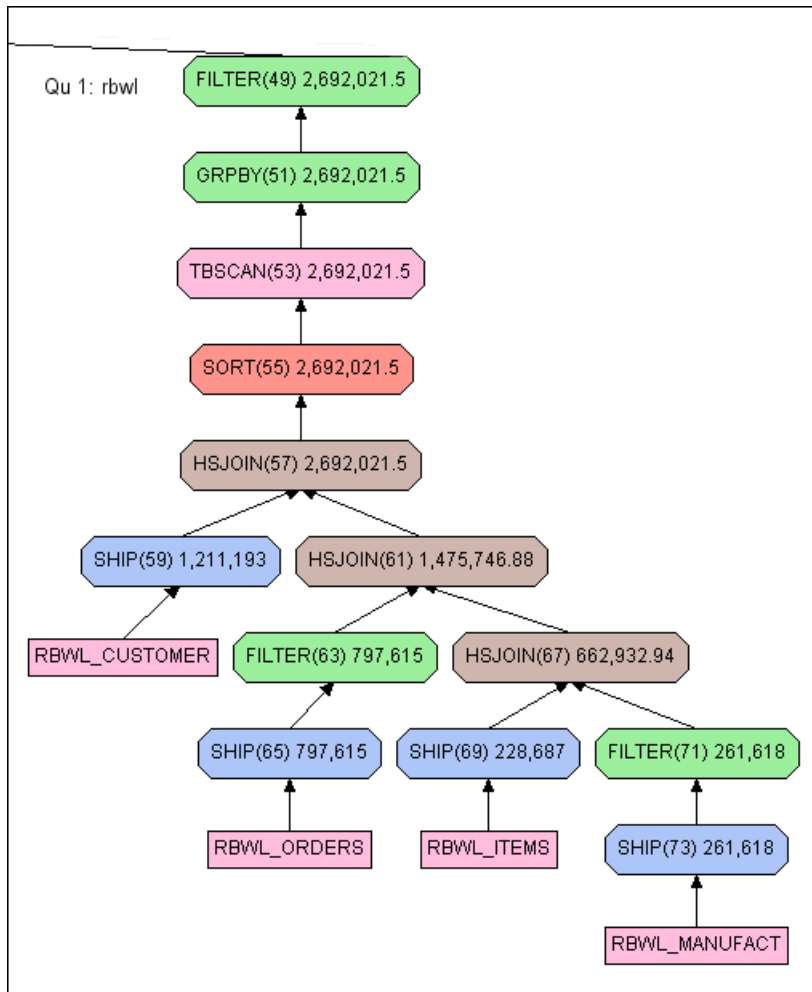


Figure 6-19 Query plan graph for Query 1 for database identifier rbwl

If we examine the remote query text for the SHIP operator, shown as operation number 65 in Figure 6-19, we will see that not even the predicate `order_date > '12/25/2001'` is being pushed down to the remote data source. This is shown in Figure 6-20.

```
SELECT A0."ORDER_DATE", A0."ORDER_NUM", A0."CUSTOMER_NUM" FROM "ORDERS" A0
```

Figure 6-20 Remote query text for ORDERS table from rbwl

The same is also true for the predicate `manu_name = 'Stephani Inc'`. The other tables in the join have no other filters to be applied. So for each table in this subquery, the required columns from every row in the *entire table* will be shipped to the federation server for the subsequent evaluation of filters, joins, grouping, and further post grouping, filters. If the populations of these tables are large, the performance consequences could be *significantly detrimental* to the remote data source, the network, or the federation database server itself.

The differences between the access plan for this data source and the other data sources occur because the connection from DB2 II to Red Brick uses a generic ODBC wrapper. ODBC is the Open Database Connectivity specification from Microsoft. This wrapper is multi-purpose and is used for a large variety of data sources with widely varying capabilities. Consequently in its default implementation, DB2 II assumes that the remote data source *cannot* perform pushdown operations. Later on in Chapter 7, “Optimization in a federated environment” on page 193, we examine ways to optimize the subquery for this data source and significantly improve performance.

Scenario 2 with DB2 II

As previously with Scenario 1, to understand this scenario it is necessary to familiarize yourself with the contents of sample Query 2 (refer to Figure 6-2 on page 165 and Figure 6-3 on page 166).

We executed Query 2 using the DB2 Command Center. After about one hour, we abandoned the query. So we used the interactive explain graph for this query to understand what happened. The graph of the query plan for Query 2 is very complicated (see Figure 6-21).

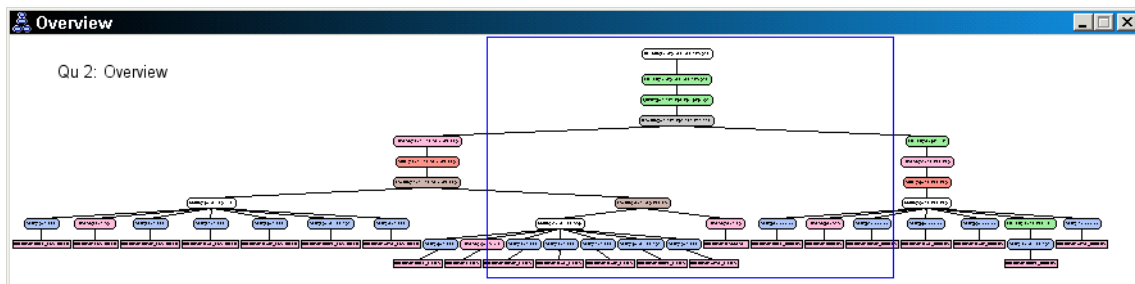


Figure 6-21 DB2 interactive explain graph overview for Query 2

From the overview graph, we can see the overall shape of the query plan. The DB2 optimizer has determined that to service this query, it is necessary to materialize the three unions in the federated views (shown in Figure 6-2 on page 165) at the federated server. One example of this materialization is shown for the union of orders in Figure 6-22 on page 180.

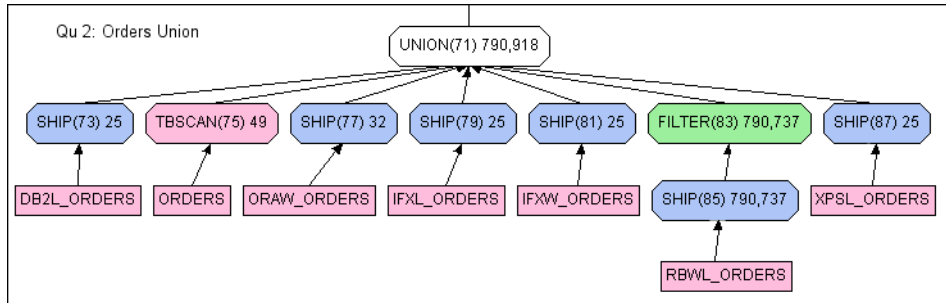


Figure 6-22 Materialization of the federated view on orders

Note that as in Query 1, the predicate on order date is not pushed down to Red Brick and appears as operation 83 in Figure 6-22. The materialization of the unions on the customer and items tables are similar, although there are no filters to be applied, and have not been illustrated. Each of these materialized unions is then joined with each other and with the manufact table. The important feature of this query, however, is the order the DB2 optimizer has chosen for the joins. This is shown schematically in Figure 6-23.

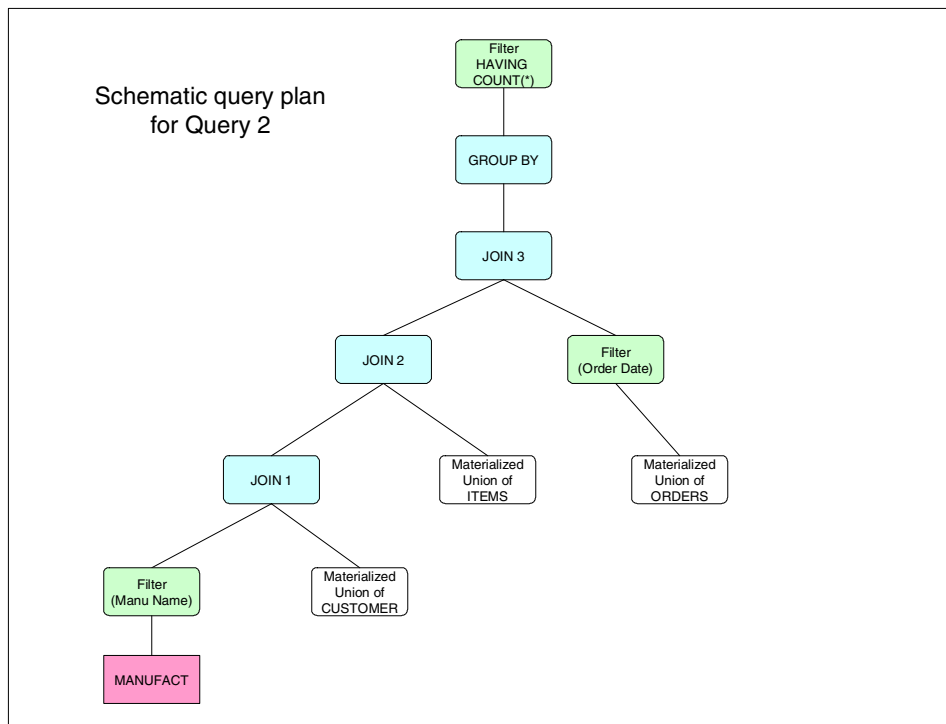


Figure 6-23 Schematic query plan for Query 2

The first join is between the qualifying rows of the `manufact` table and the materialized union of customers, across all data sources. This is a cartesian join, but as the cardinality of the `manufact` table is relatively small (around 30) and there is also an equality predicate on the column `manu_name`, the cartesian join product is no larger than the cardinality of the union of the customer table (about 10 k rows across all regions).

This interim result table is then joined to the materialized union of items across all data sources (Join 2 in Figure 6-23 on page 180). The performance problem caused a combination of a high cardinality of the items tables in our case study (total 100 K rows across all regions) and the low cardinality of the `manufact` table (around 30) as the join condition for Join 2 is only `manu_code`. The interim result table for join 2 will be on average around $10k * 100k / 30$ (about 33 million rows).

So what has gone wrong? If we look at the DB2 catalog statistics for the nicknames on the Red Brick tables we see that there are some strange looking numbers present for the cardinalities of the remote tables. All of the cardinalities are between 80–160 million rows, which far exceeds the numbers in our case study. An issue then to consider is that Red Brick does not maintain statistics in its catalog for the cardinality of tables or the distribution of field contents. The normal process of creating a nickname copies the statistics from the remote catalog to the DB2 catalog. In the case of Red Brick, there is nothing meaningful to copy. Exactly how the strange numbers turn up (instead of the normal default for DB2 of -1 when statistics are not available) is unclear. In 7.5, “Remote data source catalog statistics” on page 209, we discuss how to optimize this query through the use of the `get_stats` utility.

Scenario 3 with DB2 II

In this scenario we executed the sample Query 3 (see 6.1.3, “Query 3 - Consolidating the data” on page 167). This query generates a series of insert statements at the remote server for each row in the interim results set from the select clause in the query.

The query was executed without incident.

6.3 Testing with Informix Enterprise Gateway Manager

The tests performed with Informix Enterprise Gateway Manager involved the same data distribution and queries used for the DB2 Information Integrator. The native ODBC Drivers from each DBMS vendor were used in these tests. We chose to install EGM on a separate machine to emulate a gateway server, although it could have been installed on `winsvr` or any other server platform where the software is supported.

Note: As opposed to DB2 Information Integrator, you do not need to have an Informix database server running on the same machine where the Enterprise Gateway Manager is installed.

From the client machine, we connected to an Informix server (ifx_linsvr_tcp) running on the Linux server (linsvr), which acted as the coordinator server. We chose to execute the queries from the ServerStudioJE client and we used xtree along with EXPLAIN ON to visualize the execution plan chosen by the optimizer for the execution of each query. See 2.1, “Environment and server configuration” on page 33, for details on the implementation. You can also get the exact query fragment passed to the remote sources by setting GWDEBUG at the EGM server and observing where the query rewrite takes place.

6.3.1 Join and insert testing with EGM

In this section we discuss join and insert testing with EGM.

Scenario 1 with EGM

In our first example, we executed a simple query that joins tables *customer*, *orders*, and *items* from a DB2 data source and the *manufact* table residing at the Informix database server, which is the coordinator. The query is described in Example 6-2.

Example 6-2 The query

```
select fname, lname,o.order_num,i.item_num, m.manu_name
  from db2w_customer c, db2w_orders o, db2w_items i, manufact m
  where o.customer_num = c.customer_num
        and c.fname      = 'Julio'
        and o.order_num  < 10000
        and i.order_num  = o.order_num
        and i.manu_code  = m.manu_code
```

The query plan produced shows both the JOIN and predicates being fully pushed down to DB2 for *customer*, *orders*, and *items*. The EXPLAIN output is shown in Example 6-3 followed by the xtree diagram in Figure 6-24 on page 184. By tracing the EGM service, you will find only the query fragment being sent to the data source. This demonstrates the pushdown in action.

Example 6-3 SQL explain output

```
QUERY:
-----
select fname, lname,o.order_num,i.item_num, m.manu_name
  from db2w_customer c, db2w_orders o, db2w_items i, manufact m
```



```
where o.customer_num = c.customer_num
and c.fname = 'Julio'
and o.order_num < 10000
and i.order_num = o.order_num
and i.manu_code = m.manu_code
```

Estimated Cost: 1891

Estimated # of Rows Returned: 137

1) informix.c, informix.o, informix.i: REMOTE PATH

Remote SQL Request:

```
select x0.fname ,x0.lname ,x0.customer_num ,x1.order_num ,x1.cus
tomer_num ,x2.item_num ,x2.order_num ,x2.manu_code from db2odbcw
:"DB2DBAW".customer x0 ,db2odbcw:"DB2DBAW".orders x1 ,db2odbcw:"
DB2DBAW".items x2 where (x0.fname = 'Julio' ) AND ((x1.customer_
num = x0.customer_num ) AND (x1.order_num < 10000. ) ) AND ((x2.
order_num = x1.order_num ) AND (x2.order_num < 10000. ) )
```

2) informix.m: INDEX PATH

(1) Index Keys: manu_code (Serial, fragments: ALL)

Lower Index Filter: informix.i.manu_code = informix.m.manu_code

NESTED LOOP JOIN

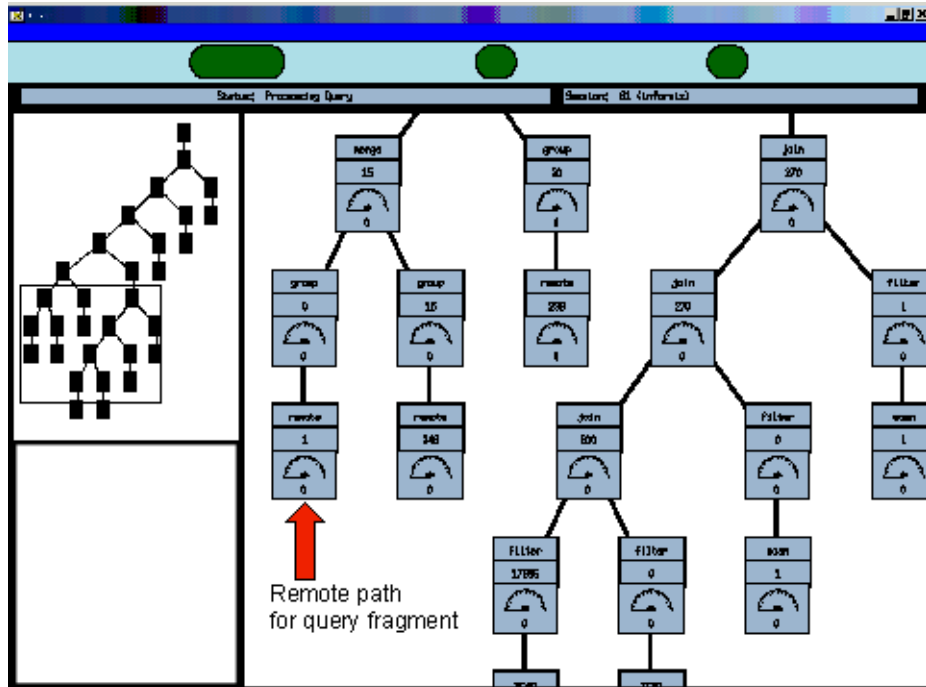


Figure 6-24 Query tree diagram showing full pushdown of join on three tables

Note: When the query is issued against a synonym, only the columns selected in the query are included in the data set returned by the remote data source. However, when issued against views based on the remote tables, the data will contain all the columns that are part of the VIEW construct. Therefore the data set may contain more columns than the ones specified in the query. See 8.6, “Using views versus synonyms with data federation” on page 220, for additional details.

Scenario 2 with EGM

In this scenario Query 1 is used as an example. In Example 6-5 on page 185 you see the query plan produced for the query. In this output, notice that there is a *remote path* for each integrator reaching a remote data source. Let us take a closer look at the query fragment shown for the SELECT over the Oracle data source based on the query fragment received by the ODBC Driver. It is depicted in Example 6-4.

Example 6-4 Query fragment for select

Native SQL Statement That the ODBC Driver Will Send to the Data Source **

Length of Native SQL (bytes): 476

Native SQL:

```
"SELECT x12.total_price,x12.manu_code,x12.order_num,x13.manu_name,x13.manu_code,x14.order_num,x14.order_date,x14.customer_num,x15.customer_num,x15.company FROM "ORADBAW".items x12,"ORADBAW".manufact x13,"ORADBAW".orders x14,"ORADBAW".customer x15 WHERE (((x13.manu_code=x12.manu_code) AND (x13.manu_name='Stephani Inc')) AND ((x14.order_num=x12.order_num) AND (x14.order_date> TO_DATE('2001-12-25 00:00:00','YYYY-MM-DD HH24:MI:SS')) AND (x14.customer_num=x15.customer_num))"
```

As you can observe, the *join* and the *where* clauses (predicates) are both being pushed down to Oracle and resolved at the remote data source while the *group by* has been chosen by the optimizer to be performed by the coordinator database. The query plan is shown in Example 6-5 followed by the xtree snapshot shown in Figure 6-25 on page 189.

Tip: See 8.6, “Using views versus synonyms with data federation” on page 220, for considerations about using views versus synonyms on remote queries.

Example 6-5 QUERY 1 execution plan (sqexplain.out)

Estimated Cost: 27563

Estimated # of Rows Returned: 296

Temporary Files Required For: Order By Group By

1) informix.i, informix.m, informix.o, informix.c: REMOTE PATH

Remote SQL Request:

```
select x20.total_price ,x20.manu_code ,x20.order_num ,x21.manu_name ,x21.manu_code ,x22.order_num ,x22.order_date ,x22.customer_num ,x23.customer_num ,x23.company from db2odbc1:"DB2DBAL".items x20 ,db2odbc1:"DB2DBAL".manufact x21 ,db2odbc1:"DB2DBAL".orders x22 ,db2odbc1:"DB2DBAL".customer x23 where ((x21.manu_code = x20.manu_code ) AND (x21.manu_name = 'Stephani Inc' ) ) AND ((x22.order_num = x20.order_num ) AND (x22.order_date > DATE ('12/25/2001' ) ) ) AND (x22.customer_num = x23.customer_num )
```

Union Query:

Temporary Files Required For: Group By

1) informix.i, informix.m, informix.o, informix.c: REMOTE PATH

Remote SQL Request:

```

select x16.total_price ,x16.manu_code ,x16.order_num ,x17.manu_n
ame ,x17.manu_code ,x18.order_num ,x18.order_date ,x18.customer_
num ,x19.customer_num ,x19.company from db2odbcw:"DB2DBAW".items
x16 ,db2odbcw:"DB2DBAW".manufact x17 ,db2odbcw:"DB2DBAW".orders
x18 ,db2odbcw:"DB2DBAW".customer x19 where ((x17.manu_code = x1
6.manu_code ) AND (x17.manu_name = 'Stephani Inc' ) ) AND ((x18.
order_num = x16.order_num ) AND (x18.order_date > DATE ('12/25/2
001' ) ) ) AND (x18.customer_num = x19.customer_num )

```

Union Query:

Temporary Files Required For: Group By

1) informix.i, informix.m, informix.o, informix.c: REMOTE PATH

Remote SQL Request:

```

select x12.total_price ,x12.manu_code ,x12.order_num ,x13.manu_n
ame ,x13.manu_code ,x14.order_num ,x14.order_date ,x14.customer_
num ,x15.customer_num ,x15.company from oraodbcw:"ORADBAW".items
x12 ,oraodbcw:"ORADBAW".manufact x13 ,oraodbcw:"ORADBAW".orders
x14 ,oraodbcw:"ORADBAW".customer x15 where ((x13.manu_code = x1
2.manu_code ) AND (x13.manu_name = 'Stephani Inc' ) ) AND ((x14.
order_num = x12.order_num ) AND (x14.order_date > datetime(2001-
12-25 00:00:00.00000) year to fraction(5) ) ) AND (x14.customer_
num = x15.customer_num )

```

Union Query:

Temporary Files Required For: Group By

1) informix.i: SEQUENTIAL SCAN

2) informix.m: INDEX PATH

Filters: informix.m.manu_name = 'Stephani Inc'

(1) Index Keys: manu_code (Serial, fragments: ALL)

Lower Index Filter: informix.m.manu_code = informix.i.manu_code

NESTED LOOP JOIN

3) informix.o: INDEX PATH

Filters: informix.o.order_date > 12/25/2001

(1) Index Keys: order_num (Serial, fragments: ALL)

Lower Index Filter: informix.o.order_num = informix.i.order_num

NESTED LOOP JOIN

4) informix.c: INDEX PATH

(1) Index Keys: customer_num (Serial, fragments: ALL)
Lower Index Filter: informix.o.customer_num = informix.c.customer_num
NESTED LOOP JOIN

Union Query:

Temporary Files Required For: Group By

1) informix.o, informix.c, informix.m, informix.i: REMOTE PATH

Remote SQL Request:

```
select x8.order_num ,x8.order_date ,x8.customer_num ,x9.customer
_num ,x9.company ,x10.manu_name ,x10.manu_code ,x11.total_price
,x11.manu_code ,x11.order_num from stores:"ifxdbaw".orders x8 ,s
tores:"ifxdbaw".customer x9 ,stores:"ifxdbaw".manufact x10 ,stor
es:"ifxdbaw".items x11 where (x8.order_date > DATE ('12/25/2001'
) ) AND (x8.customer_num = x9.customer_num ) AND (x10.manu_name
= 'Stephani Inc' ) AND ((x8.order_num = x11.order_num ) AND (x1
0.manu_code = x11.manu_code ) )
```

Union Query:

Temporary Files Required For: Group By

1) informix.i, informix.m, informix.o, informix.c: REMOTE PATH

Remote SQL Request:

```
select x4.total_price ,x4.manu_code ,x4.order_num ,x5.manu_name
,x5.manu_code ,x6.order_num ,x6.order_date ,x6.customer_num ,x7.
customer_num ,x7.company from rbwodbcl:"RBWDBAL".items x4 ,rbwod
bcl:"RBWDBAL".manufact x5 ,rbwodbcl:"RBWDBAL".orders x6 ,rbwodbc
l:"RBWDBAL".customer x7 where ((x5.manu_code = x4.manu_code ) AN
D (x5.manu_name = 'Stephani Inc' ) ) AND ((x6.order_num = x4.ord
er_num ) AND (x6.order_date > DATE ('12/25/2001' ) ) ) AND (x6.c
ustomer_num = x7.customer_num )
```

Union Query:

Temporary Files Required For: Group By

1) informix.i, informix.m, informix.o, informix.c: REMOTE PATH

Remote SQL Request:

```
select x0.total_price ,x0.manu_code ,x0.order_num ,x1.manu_name
,x1.manu_code ,x2.order_num ,x2.order_date ,x2.customer_num ,x3.
customer_num ,x3.company from stores:"xpsdbal".items x0 ,stores:
"xpsdbal".manufact x1 ,stores:"xpsdbal".orders x2 ,stores:"xpsdb
```

```

a1".customer x3 where ((x1.manu_name = 'Stephani Inc' ) AND (x1.
manu_code = x0.manu_code ) ) AND ((x2.order_num = x0.order_num )
AND (x2.order_date > DATE ( '12/25/2001' ) ) ) AND (x2.customer_
num = x3.customer_num )

```

By having GWDEBUG set to 121 for all gateway servers involved in the query, it is possible to examine the trace file and see what transformation in the query text was needed (*if any*). For Query 1, we can see the SQL statement portion sent to DB2 on Windows being rewritten, in Example 6-6, to reflect the correct remote source syntax.

Example 6-6 Query rewrite as seen in the trace output

SQL statement received by Gateway to be prepared [Len: 536 bytes]:

```

"select x12.total_price ,x12.manu_code ,x12.order_num ,x13.manu_name ,x1
3.manu_code ,x14.order_num ,x14.order_date ,x14.customer_num ,x15.custome
r_num ,x15.company from oraodbcw:"ORADBAW".items x12 ,oraodbcw:"ORADBAW
".manufact x13 ,oraodbcw:"ORADBAW".orders x14 ,oraodbcw:"ORADBAW".custome
r x15 where ((x13.manu_code = x12.manu_code ) AND (x13.manu_name = 'Ste
phani Inc' ) ) AND ((x14.order_num = x12.order_num ) AND (x14.order_date
> datetime(2001-12-25 00:00:00.00000) year to fraction(5) ) ) AND (x14.
customer_num = x15.customer_num )"

```

The Gateway recognizes this SQL as valid IBM Informix syntax.

SQL statement as modified by Gateway [Len: 446 bytes]:

```

"SELECT x12.total_price,x12.manu_code,x12.order_num,x13.manu_name,x13.ma
nu_code,x14.order_num,x14.order_date,x14.customer_num,x15.customer_num,x
15.company FROM "ORADBAW".items x12,"ORADBAW".manufact x13,"ORADBAW".ord
ers x14,"ORADBAW".customer x15 WHERE (((x13.manu_code=x12.manu_code) AN
D (x13.manu_name='Stephani Inc')) AND ((x14.order_num=x12.order_num) AND
(x14.order_date>{ts '2001-12-25 00:00:00'}))) AND (x14.customer_num=x15
.customer_num)"

```

```

===== Time: 2003-07-12 15:49:15.000000 =====

```

```

===== Time: 2003-07-12 15:49:15.000000 =====

```

```

** Native SQL Statement That the ODBC Driver Will Send to the Data Source **

```

Length of Native SQL (bytes): 476

Native SQL:

```

"SELECT x12.total_price,x12.manu_code,x12.order_num,x13.manu_name,x13.ma
nu_code,x14.order_num,x14.order_date,x14.customer_num,x15.customer_num,x
15.company FROM "ORADBAW".items x12,"ORADBAW".manufact x13,"ORADBAW".ord
ers x14,"ORADBAW".customer x15 WHERE (((x13.manu_code=x12.manu_code) AN
D (x13.manu_name='Stephani Inc')) AND ((x14.order_num=x12.order_num) AND

```

```
(x14.order_date> TO_DATE('2001-12-25 00:00:00','YYYY-MM-DD HH24:MI:SS')
))) AND (x14.customer_num=x15.customer_num))"
```

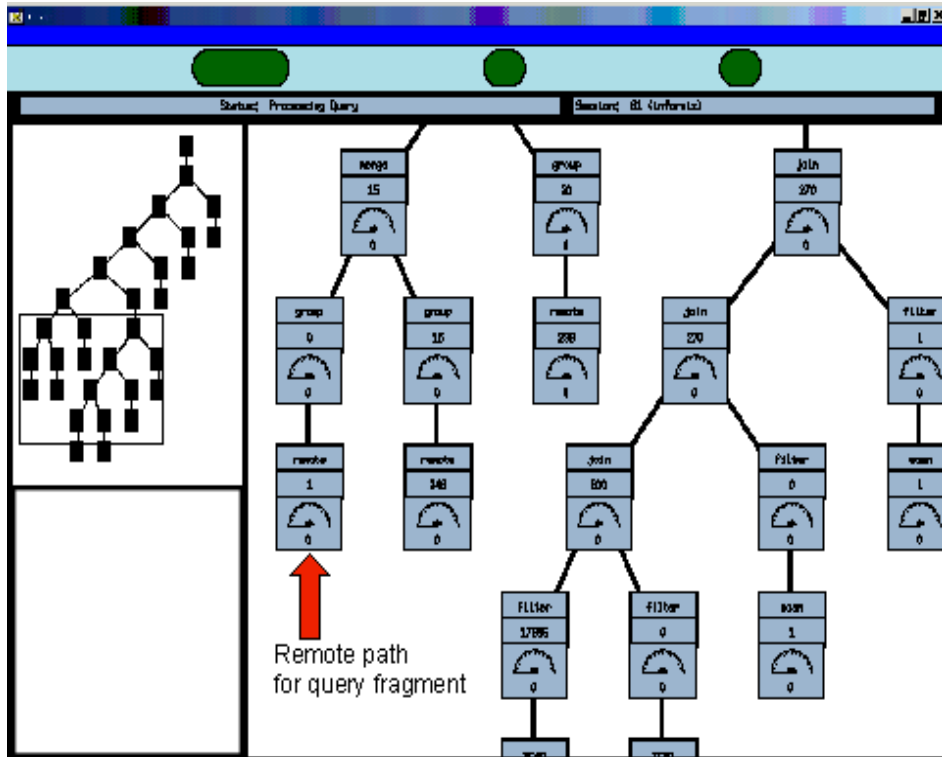


Figure 6-25 Query tree diagram for Query 1 execution

Note: When we executed the Red Brick portion of Query 1 through EGM, it returned the correct number of rows. However, when the same query fragment was added to the UNION with the other data sources, this particular result set returned an invalid number of rows.

We were unable to include predicates over the date column in views that included Oracle data sources on a UNION with other databases. Such an operation fails with error 29044. The commands below illustrate the scenario:

```
CREATE VIEW tab as
SELECT col1 from ifx_table UNION SELECT date(col1) from ora_table;
SELECT * from TAB where col1 > '01/01/2001');
```

These situations have been reported to development and are being investigated.

Scenario 3 with EGM

In this last scenario we executed Query 3 (see 6.1.3, “Query 3 - Consolidating the data” on page 167). This query populates a physical table `corp_customer` from a union of all customers from all data sources. It populates the table by selecting all rows from the view `ifxl_join_customer`. The sequence of commands is shown in Example 6-7.

Example 6-7 Populating Informix tables from other data sources

```
CREATE VIEW ifxl_join_customer as
select * from db2l_customer union all
select * from db2w_customer union all
select * from oraw_customer union all
select * from ifxl_customer union all
select * from ifxw_customer union all
select * from rbwl_customer union all
select * from xpsl_customer;

CREATE TABLE corp_customer
(
    customer_num serial not null primary key,
    fname char(15),      lname char(15),
    company char(20),    address1 char(20),
    address2 char(20),   city char(15),
    state char(02),      zipcode char(05),
    phone char(18)
);
INSERT INTO corp_customer SELECT * FROM ifxl_join_customer;
```

6.3.2 Updating multiple data sources

As discussed in “Distributed transactions” on page 151 you can have multiple Informix participants in your transaction but only one non-Informix data source. In Figure 6-26 on page 191 we show an attempt to give a 10 percent discount on the shipping charge for orders in Regions 1, 2, 3, 4, and 5. The first update on Region 1 targets an Oracle database and the UPDATE is performed successfully. However, the update on Region 2 fails because it tries to update a DB2 data source in the same transaction. Thus, `sqlerror -440` is returned stating 440: Cannot update more than one non_Informix DBMS within a transaction.

If the update on Region 2 is removed from the transaction, the updates in a two-phase commit fashion are completed in one single transaction, as shown in Example 6-8 on page 191.

Example 6-8 Distributed update

```
BEGIN WORK;  
  UPDATE ORAW_ORDERS set ship_charge = (ship_charge - ship_charge * 0.10);  
  UPDATE IFXL_ORDERS set ship_charge = (ship_charge - ship_charge * 0.10);  
  UPDATE XPSL_ORDERS set ship_charge = (ship_charge - ship_charge * 0.10);  
  UPDATE IFXW_ORDERS set ship_charge = (ship_charge - ship_charge * 0.10);  
COMMIT;
```

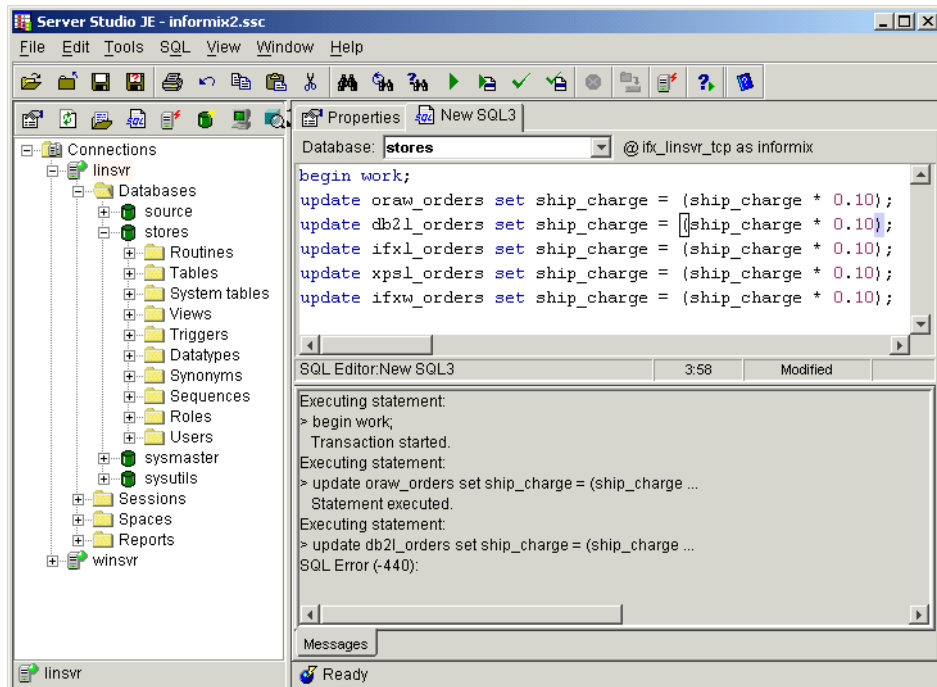


Figure 6-26 SQLerror -440 trying to update multiple non-Informix sources



Optimization in a federated environment

In this chapter we discuss a number of techniques that can be used to improve the performance of federated queries. We also extend our case study samples from Chapter 6, “Data federation in action” on page 159, to give practical examples of the use of these optimization techniques. We show examples of how to use these techniques in practice and also the detailed results of their application.

7.1 Performance options and considerations

There are many ways in which we can work with the federated environment to influence its performance:

- ▶ Remote data source tuning
- ▶ Changing server options (specific to DB2 II)
- ▶ Use of Materialized Query Tables (MQTs) (specific to DB2 II)
- ▶ Remote data source catalog statistics
- ▶ Optimizer hints in query text

We discuss these in the following sections.

7.2 Remote data source tuning

The federated database server appears to the remote clients as just another client application for that data source. So while we are considering performance in a federated environment, it is important not to forget that in addition to all the techniques we have available to us as a consequence of the federation technology, ultimately the performance of the federated server is based on the performance of the remote data sources.

With the exception of MQTs, the federated database server retrieves the requested data for the client application in real time, and is therefore ultimately dependent on the ability of the remote data sources to synchronously deliver the remote data in a timely fashion.

We do not intend to discuss here the many options and techniques for tuning the wide variety of data sources that can be federated using either DB2 II or Informix EGM. But a part of any exercise to optimize a federated database should include an analysis of performance at the remote data sources.

7.3 Server options in DB2 II

Each wrapper supplies defaults for server options, which will be applied every time a new server is created using that particular wrapper. You can alter these options for a particular server using either the DB2 Control Center or using the ALTER SERVER statement in SQL. When you alter the server options in this way, you change the options that will persist for all future connections to that server. In addition, you can override the server options for the *current connection only* using the SET SERVER OPTION SQL statement.

There are many server options that affect performance, but we will address the following options as the ones we consider to be the most important:

- ▶ PUSHDOWN
- ▶ DB2_MAXIMAL_PUSHDOWN
- ▶ COLLATING_SEQUENCE

We will consider the use of each of these options and, where relevant, illustrate them in our case study.

7.3.1 Pushdown

In the manual *IBM DB2 Information Integrator Federated Systems Guide*, SC18-7364, the definition for the PUSHDOWN parameter appears as follows:

- ▶ 'N' - DB2 will send the data source SQL statements that include only SELECT with column names. Predicates (such as WHERE=) column and scalar functions (such as MAX and MIN), sorts (such as ORDER BY or GROUP BY), and joins will not be included in any SQL sent to the data source.
- ▶ 'Y' - DB2 will consider letting the data source evaluate operations.

Tip: Pushdown is only considered for relational data sources.

From our understanding of the previous discussion of pushdown, we know the importance of this parameter. If pushdown of remote operations is not performed, then the required columns from *every* row in a remote data source will be sent to the federation server. If the cardinality of the table(s) in the remote data source is large, this can have a *significantly* adverse performance implication for the remote data source, the network, or the federation database server itself.

Generally speaking, the default value adopted for the PUSHDOWN parameter is 'Y' for relational wrappers (DB2, Informix, Oracle, Sybase). However, the default value of the PUSHDOWN parameter is 'N' in the *ODBC wrapper*. This is the wrapper used by DB2 II for accessing the Red Brick data source.

In “Scenario 1 with DB2 II” on page 170, we considered the performance of sample Query 1. We determined that the access path for this federated query was quite different for the Red Brick data source when compared to the other relational data sources used in our case study.

To optimize this subquery within sample Query 1, we decided to override the default value adopted for the PUSHDOWN parameter using the DB2 Control Center. This is shown in Figure 7-1 on page 196 and Figure 7-2 on page 196.

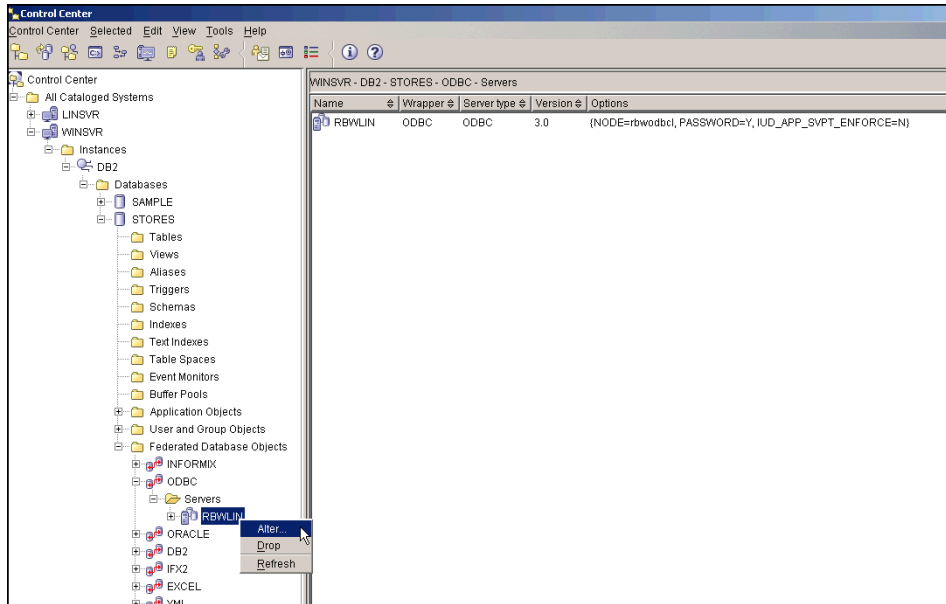


Figure 7-1 Accessing the server options panel using DB2 Control Center

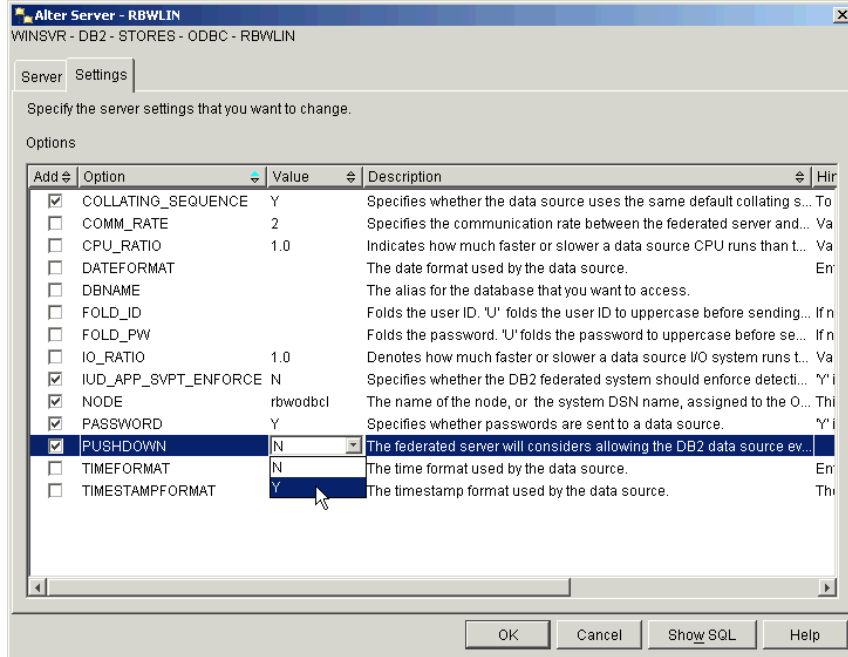


Figure 7-2 Setting the PUSHDOWN server option for an ODBC server

Of course we can also change the server options using native SQL which we can run using either the DB2 Command Center or the DB2 command line processor. We used the Show SQL function to capture the SQL being generated by the control center GUI. This is the native SQL command to achieve the same effect and is shown in Figure 7-3

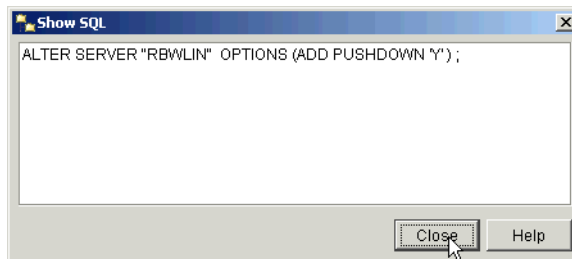


Figure 7-3 SQL to add the PUSHDOWN server option

Attention: The Release Notes for DB2 II V8.1 carry the following information concerning the use of the PUSHDOWN parameter with the ODBC wrapper:

“If setting PUSHDOWN server option value to 'Y' causes some queries to fail during remote statement generation, remove the server option or set the PUSHDOWN server option to 'N' to prevent this problem.”

What difference did the setting the PUSHDOWN server option make for the Red Brick data source in Query 1? Let us revisit Query 1 again and compare the interactive explain graph shown in Figure 7-4 on page 198 with the original shown in Figure 6-19 on page 178.

Not a lot appears to have been changed. Two of the hash joins have changed to nested loop joins, but the overall join strategy is similar, with the tables being joined in the same sequence. The only significant difference we can observe is that the FILTER (operation 63) in Figure 6-19 on page 178 has disappeared from the explain graph. If we use **Show Details** against the SHIP (operator 69) in Figure 7-4 on page 198, we will see that this predicate has been pushed down to the Red Brick data source (see Figure 7-5 on page 198).

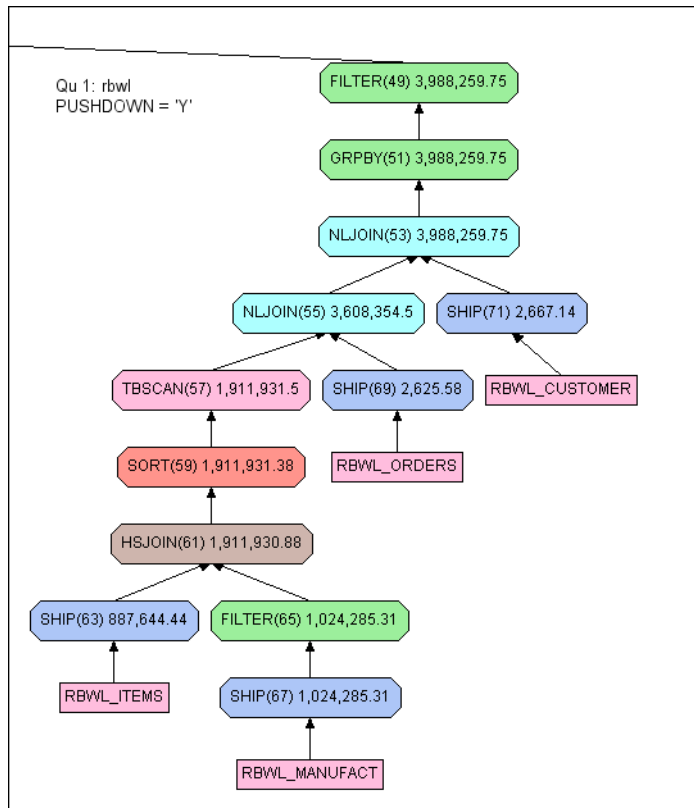


Figure 7-4 Explain for rbwl data source with PUSHDOWN option

```
SELECT A0."ORDER_NUM", A0."CUSTOMER_NUM" FROM "ORDERS" A0 WHERE
(A0."ORDER_NUM" = :HO ) AND ( {d' 2001-12-25' } < A0."ORDER_DATE" )
```

Figure 7-5 Pushed down predicate on data source rbwl

This pushdown will definitely improve the performance of this query, as less rows will be returned from the remote data source to the federation server. Note that the predicate on the manufact table was not pushed down. This is probably because the cardinality of this table in our case study is low and it was not considered that the cost of the pushed down predicate would be advantageous.

We have made some improvement to Query 1, but can we do more? Let us look at the next tuning parameter, DB2_MAXIMAL_PUSHDOWN, to see how we can improve the performance further still.

7.3.2 Maximal pushdown

In the manual *IBM DB2 Information Integrator Federated Systems Guide*, SC18-7364, there is the following discussion of the DB2_MAXIMAL_PUSHDOWN parameter:

With DB2_MAXIMAL_PUSHDOWN set to 'Y', reducing network traffic becomes the overriding criteria for the query optimizer. The query optimizer uses the access plan that performs the fewest number of “sends” to the data sources. Setting this server option to 'Y' forces the federated server to use an access plan that might not be the lowest cost plan.

In our case study we set this server option using native SQL, as this option cannot be set using the DB2 Control Center GUI in the current release of DB2 II. The SQL we used appears in Figure 7-6.

```
ALTER SERVER “RBWLIN” OPTIONS(ADD DB2_MAXIMAL_PUSHDOWN ‘Y’)
```

Figure 7-6 SQL to add the DB2_MAXIMAL_PUSHDOWN server option

Tip: Although the value of the server option DB2_MAXIMAL_PUSHDOWN cannot be set using the DB2 Control Center GUI, its current value can still be seen using the control center GUI.

Of course we were interested to use the interactive explain graph to see the impact that setting this server option would have. The new graph is shown in Figure 7-7 on page 200. Now compare this diagram with the original in Figure 6-19 on page 178. The three table joins for the customer, orders, and items tables and the predicate on order date have now been completely pushed down to the remote data source. This diagram now begins to show some similarities with explain graphs for the other data sources in Query 1 (such as the explain graph for Informix Dynamic Server shown in Figure 6-15 on page 175).

However, there are still some differences. DB2 is still responsible for the predicate on the manufact table, and there is a GRPBY (operator 51) in Figure 7-7 on page 200. The GRPBY operator is needed, as the ODBC wrapper does not support the required GROUP BY and HAVING clauses and DB2 is compensating for this at the federated server. The performance for this data source has been optimized through the use of the server option settings.

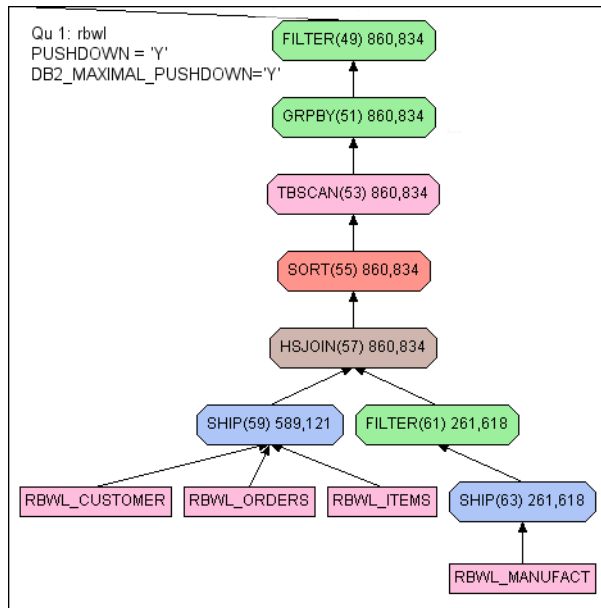


Figure 7-7 Pushdown operations for data source rbwl

7.3.3 Collating sequence

The `COLLATING_SEQUENCE` server option informs DB2 II about the collating sequence at the remote data source. When the collating sequence at the remote data source is the same as the federated database server, DB2 II can consider pushdown operations involving sorts and inequality operations. For example, had our sample Query 1 used the predicate `manu_name <= 'Stephani Inc'`, this predicate would not have been eligible for pushdown to a remote data source where the collating sequence was different.

When inequality predicates are used and the collating sequence is different at the remote data source, the explain graph for the query will show a `FILTER` operator above the `SHIP` operator to evaluate the inequality predicate according to DB2's collating sequence at the federation server.

The `COLLATING_SEQUENCE` server option has three possible values as follows:

- ▶ Y - The collating sequence at the remote data source is the same as the federation server.
- ▶ N - The collating sequence at the remote data source is different from the federation server but still obeys the rules for case sensitivity.

- ▶ I - The collating sequence at the remote data source is different from the federation server and is case insensitive. ('Stephani Inc' is considered equal to 'STEPHANI INC'.)

Note: The predicates '=' and 'NOT =' are still eligible for pushdown to a remote data source where the server option COLLATING_SEQUENCE is 'N', but not when it is 'I'.

To optimize your performance, ensure that you have the server option COLLATING_SEQUENCE correctly set for your remote data source. The default for this parameter is actually 'N', so if you see inequality predicates or sorts not being pushed down where you think they should be, check the settings for this parameter.

Even if the remote data source uses a collating sequence that is different from the federated database server, it may still be possible to have inequality predicates and ORDER BY pushed down to the remote data source. Suppose there is a field at the remote data source that is specified as CHAR or VARCHAR but actually contains only numeric characters. In this instance you can use the NUMERIC_STRING option of the ALTER NICKNAME command to instruct the DB2 optimizer to not exclude this column from pushdown operations because of collating sequence differences (see Example 7-1).

Example 7-1 Example of NUMERIC_STRING nickname option

```
ALTER NICKNAME nick_name ALTER COLUMN col_name  
OPTIONS (ADD NUMERIC_STRING 'Y');
```

Let us see this parameter in practice in our case study. As previously stated, the *default* value for the COLLATING_SEQUENCE server option is 'N'. So what is the actual behavior of the DB2 optimizer caused by this default value? We altered the predicate on manu_name in our sample Query 1 from an equality to an inequality predicate, as shown in Figure 7-8 on page 202.

```

...
union all

select c.customer_num, c.company, o.order_num, sum (i.total_price) as
total_manu_rev
from ifxl_customer c, ifxl_orders o, ifxl_items i, ifxl_manufact m
where o.order_date > '12/25/2001' and m.manu_name >= 'Stephani Inc'
and m.manu_code = i.manu_code and o.order_num = i.order_num
and o.customer_num = c.customer_num
group by c.customer_num, c.company, o.order_num
having count (*) >= 5

union all
...

```

Figure 7-8 Extract from modified version of Query 1 using inequality predicate

The leg of the interactive explain graph for the modified Query 1 for the ifxl data source is shown in Figure 7-9 on page 203. Note how much of an effect the default value for this parameter is having. The filter (operation 15) cannot be pushed down to the data source, so the interim results set shipped in operation 17 contains data for all values of the manu_code rather than the ones requested in the query. The *sort* (operation 13), the *group by* (operation 9), and the *having* filter all need to be performed at the federation database server.

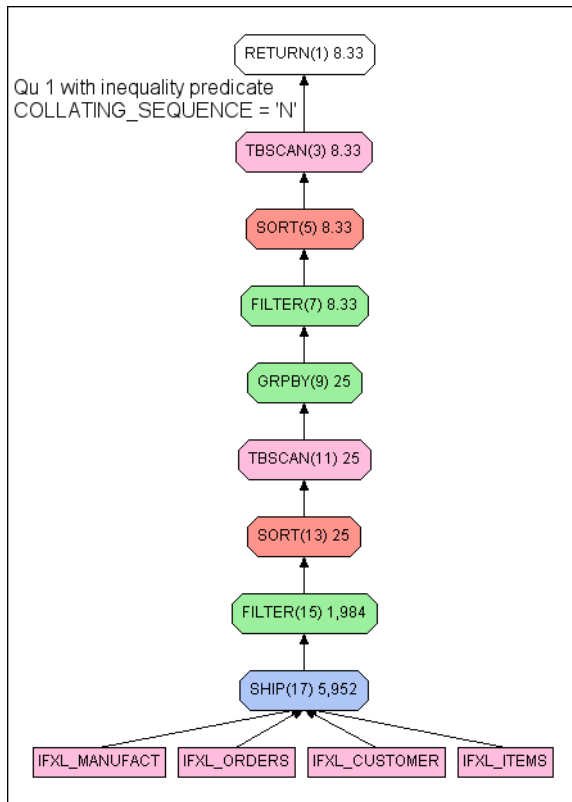


Figure 7-9 Explain with COLLATING_SEQUENCE = 'N'

Now we know that the collating sequence at the remote data source (in this case Informix Dynamic Server on Linux) is the same as DB2's collating sequence, so we used the SQL shown in Figure 7-10 to inform DB2 that the collating sequence at the remote data source can be trusted.

```
ALTER SERVER "IFXLIN" OPTIONS (SET COLLATING_SEQUENCE 'Y');
```

Figure 7-10 SQL to alter COLLATING_SEQUENCE at the server level

Now let us see the impact on the interactive explain graph. Figure 7-11 on page 204 shows that once we set the correct value for the COLLATING_SEQUENCE parameter, the access plan is much improved. The filtering and grouping are pushed down to the remote data source. DB2 has still determined that it is still more efficient to perform the final sort on the federated server, but we can be assured that the pushdown of the filtering and grouping predicates to the remote data source has improved performance significantly.

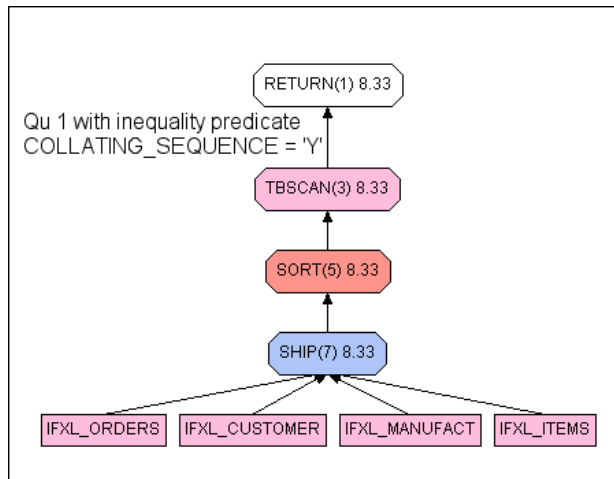


Figure 7-11 Explain with COLLATING_SEQUENCE = 'Y'

7.4 Use of Materialized Query Tables (MQTs)

One of the performance features of DB2 V8 is the ability of the query optimizer to rewrite a query to use a Materialized Query Table (MQT) instead of the underlying table(s) to satisfy a query request. If the query can be rewritten successfully to use the MQT instead of the underlying table, the query is said to have been *subsumed*.

When an MQT is created, the CREATE TABLE statement specifies whether the MQT is REFRESH IMMEDIATE or REFRESH DEFERRED. An MQT that is specified with REFRESH DEFERRED may be subject to some degree of latency. Before an MQT created with REFRESH DEFERRED can be considered eligible to subsume a query, certain conditions concerning the requestor's acceptable degree of latency in the data have to be met.

With DB2 II this ability to use an MQT is extended to federated queries using remote data sources. Not all the features associated with summary tables are currently available when using MQTs based on nicknames. For example, the automatic realtime refresh (REFRESH IMMEDIATE option) is not supported. So if we decide to create an MQT to support a federated query, then we must also be prepared to accept some degree of data latency in our query results.

We must also inform the DB2 optimizer that we are prepared to accept this data latency, in order to have the optimizer consider the MQT as eligible for subsuming the query request. If the application can suffer the latency in the data,

then using an MQT for federated data can have a *significant* effect on performance.

For some applications this data latency may not be an issue at all. Consider an accounting system that only updates the financial results for the company on a monthly basis. In this case, even though the data in an MQT is based on a refresh that may have taken place some considerable time beforehand, the data is actually still current, as the underlying tables have not changed. Obviously a thorough knowledge of the application is required to make good judgements on the maximum data latency, which is possible or acceptable depending on the circumstances.

In fact, the use of an MQT for federated data is really a compromise between the data federation and data consolidation strategies. (See 1.1, “The challenge of information integration” on page 14, for a fuller discussion of these approaches.)

Another interesting use of an MQT in federated system is to use it as a *replica*. As an example, consider a star schema with a fact table that is a union-all view of a local DB2 UDB table and a remote Informix table (that is accessed via nickname). Assume that the dimension tables reside on the remote Informix database, and that nicknames are created for them in DB2 UDB. A join of the fact table with the dimension tables will then always involve a join of the local table with the remote dimension tables, which could impact performance.

A good solution for such a performance impact would be to create MQTs over the dimension nicknames, if they are relatively small in size and do not change frequently. It would essentially just require a `select * from nickname` command to create the MQT on the local server. However, you should ensure that the replica MQTs have the same indexes as the local table. Then DB2 UDB would be able to use the MQTs for the join with the local table. Also, it should push the join of the remote fact table with dimension tables to the remote server. This should result in better performance.

In the previous example, one of the variables was data volatility. That is, how often does the data change? This becomes a vital question because online refresh of an MQT could be relatively slow, and potentially tie up connections. An alternative to this would be to export the query results from the remote system and then load them into the MQT, rather than performing an online refresh.

Let us look at the use of an MQT in our case study on Query 1 to see if we can improve the performance of the Red Brick data source further still.

In our case study let us assume that we are more interested in having this query run on a monthly basis and thus having the absolutely latest data is not critical to the results for Query 1. We decide to create an MQT to support this query as the performance for the Red Brick data source can be much faster with a complete

pushdown of the subquery to the data source. The definition of the MQT we created is shown in Figure 7-12.

Note the content of the subquery clause of this CREATE TABLE statement. We have included the columns manu_name and order_date. Although these columns are not in the result table of Query 1, they are included as predicates. It is *not* mandatory for an MQT to contain all of the columns of the query to be subsumed. The optimizer may still consider an access plan that uses an MQT without all the required columns by joining the MQT to the source tables, but this increases the cost of this plan option. In addition, there are other restrictions with the use of MQTs where refresh is deferred.

```
drop table rbwl_sum_orders;

create table rbwl_sum_orders as
(select c.customer_num, c.company, o.order_num, sum(i.total_price) as
total_order_price, count(*) as item_count, order_date, manu_name
from rbwl_customer c, rbwl_orders o, rbwl_items i, rbwl_manufact m
where m.manu_code = i.manu_code and o.order_num = i.order_num and
o.customer_num = c.customer_num
group by c.customer_num, c.company, o.order_num, order_date, manu_name
)

data initially deferred refresh deferred;

refresh table rbwl_sum_orders;
```

Figure 7-12 SQL for the MQT rbwl_sum_orders

The refresh options “data initially deferred refresh deferred” used in Figure 7-12 are *mandated* as this MQT is based on our federated nicknames.

Tip: If we had attempted to specify data initially deferred refresh immediate, the statement would have failed with SQL code -20058 and SQL state 428EC.

The **refresh table** command actually issues the select statement contained within the QMT definition. However, we are still not quite ready to use our MQT in Query 1. As our MQT was created with the (mandated) option refresh deferred, we must inform the DB2 optimizer that we are willing to accept any data latency associated with our MQT. To do this we must issue the statement shown in Figure 7-13 on page 207 to set the value of the special register current refresh age.


```
set current refresh age = any
```

Figure 7-13 Setting the special register current refresh age to use a deferred MQT

This special register must be set for the current database connection before we issue the SQL for Query 1. Now let us look at the interactive explain graph for Query 1. We now need to issue two commands (the **set current refresh** and the SQL for the query itself). Using the command center to do the interactive explain is now difficult. If you use the Script tab, which allows use of multiple SQL statements, and request the interactive graph, you will get the message shown in Figure 7-14.

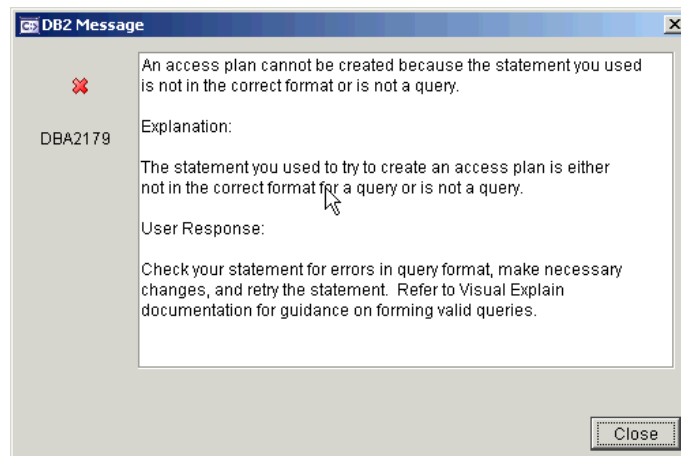


Figure 7-14 Error with explain when set current refresh

The best way to see the access plan is to use the db2exfmt tool (see Example 4-11 on page 123).

For consistency with the other diagrams in this publication, we used an internal DB2 facility to enable current refresh age with a wider context. This internal facility allowed us to use the command center to produce the interactive explain graph *as if we had issued the command* **set current refresh age = any**. The overview for Query 1 is shown in Figure 7-15 on page 208.

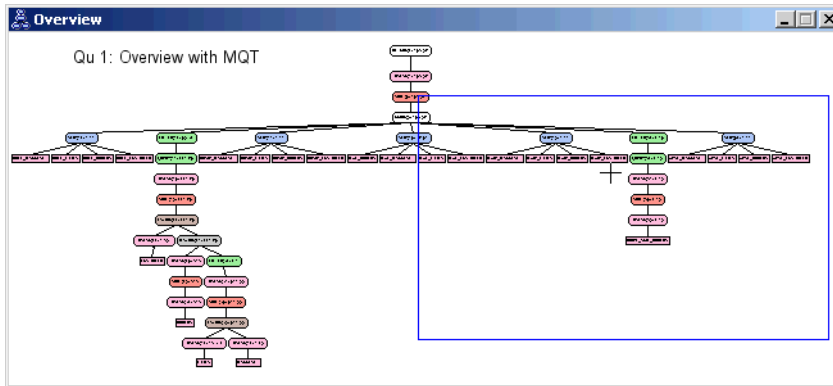


Figure 7-15 DB2 interactive explain graph for Query 1 with MQT

We can see from the shape of the overview that the sixth leg of the query plan for the Red Brick data source has changed considerably (compare this with the original in Figure 6-8 on page 170).

The detail for this leg of the interactive explain graph is shown in Figure 7-16. It shows the *rbwl* data source when using MQT.

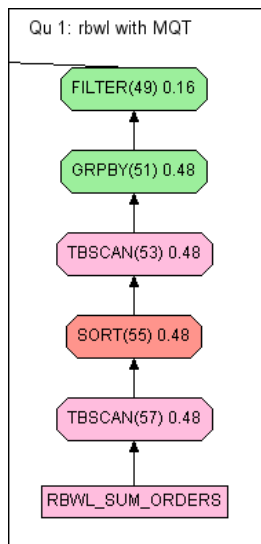


Figure 7-16 Using MQT

Now the join is eliminated completely. The DB2 optimizer has rewritten the original subquery from sample Query 1 to use the MQT instead. Because the MQT definition is summarized at a lower level (group by customer_num,

company, order_num, order_date, manu_name), the optimizer has been able to generate a new higher level grouping, summation, and count at the level specified in Query 1 (group by customer_num, company, order_num).

In fact, for Query 1, DB2 no longer even accesses the Red Brick data source. If the Red Brick data source was unavailable at query runtime, it would make no difference. This capability to support queries written against tables that are operationally unavailable at query runtime is one of the other common uses of MQT's.

As you will not be able to use the interactive explain feature of the DB2 Command Center to investigate the access plan for a deferred refresh MQT, we have also shown the explain output captured using the **db2exfmt** tool in Figure 7-17. It shows the *rbwl* data source when using MQT.

```
+-----+
|
| 0.16
| FILTER
| ( 74)
| 25.0622
| 1
| |
| | 0.48
| | GRPBY
| | ( 75)
| | 25.0619
| | 1
| | |
| | | 0.48
| | | TBSCAN
| | | ( 76)
| | | 25.0617
| | | 1
| | | |
| | | | 0.48
| | | | SORT
| | | | ( 77)
| | | | 25.061
| | | | 1
| | | | |
| | | | | 0.48
| | | | | TBSCAN
| | | | | ( 78)
| | | | | 25.0598
| | | | | 1
| | | | | |
| | | | | | 36
| | | | | | TABLE: DB2DBAW
| | | | | | RBWL_SUM_ORDERS
|
|
|
```

Figure 7-17 db2exfmt explain

7.5 Remote data source catalog statistics

The optimizer for a relational database management system is the program responsible for choosing the most efficient access strategy to service a particular

query. In general, to make its determination of the best access strategy, an optimizer program will utilize a variety of information, such as:

- ▶ The SQL request itself
- ▶ The structure of the table to be accessed and any available indexes
- ▶ Statistical information regarding the population (cardinality) of the tables and the number of unique values within the columns, the level of organization or disorganization of the tables and/or indexes, and the distribution of column values within their key ranges

The optimizers for DB2 UDB and Informix Dynamic Server are both typical in this respect in that they use a range of statistical information, which each DBMS stores in its respective metadata catalog.

In certain circumstances, the accuracy of the catalog statistics can be *critical* to the selection of a successful access strategy by the optimizer. The optimizer is designed to select an access path that is appropriate either based on the statistical information present in the catalog, or based on default values when actual values have not been captured. However, if the actual data volumes are significantly higher, then the query may in fact never complete. One reason for this, for example, could be due to insufficient temporary workspace.

Both of the federation technologies, DB2 Information Integrator and Informix Enterprise Gateway Manager, offer support for statistical information concerning the remote data source.

Refer to “Statistics and index specifications” on page 108 for the details on two ways to gather statistical information for remote data sources with DB2 II. Refer also to “Effect of Informix style system catalog” on page 152 for details on statistical information for remote data sources with Informix Enterprise Gateway Manager.

In “Scenario 2 with DB2 II” on page 179, our sample Query 2 could not be executed to completion because the join strategy was going to generate an interim result set containing 33 million rows.

We used the **get_stats** utility to update the catalog statistics for the Red Brick data source `rbwl`. Then, using the DB2 Command Center interactive explain facility, we examined the access strategy now chosen by the DB2 optimizer. Again the overview diagram is quite complex, so a schematic diagram for the access path chosen is easier to read. This is shown in Figure 7-18 on page 211.

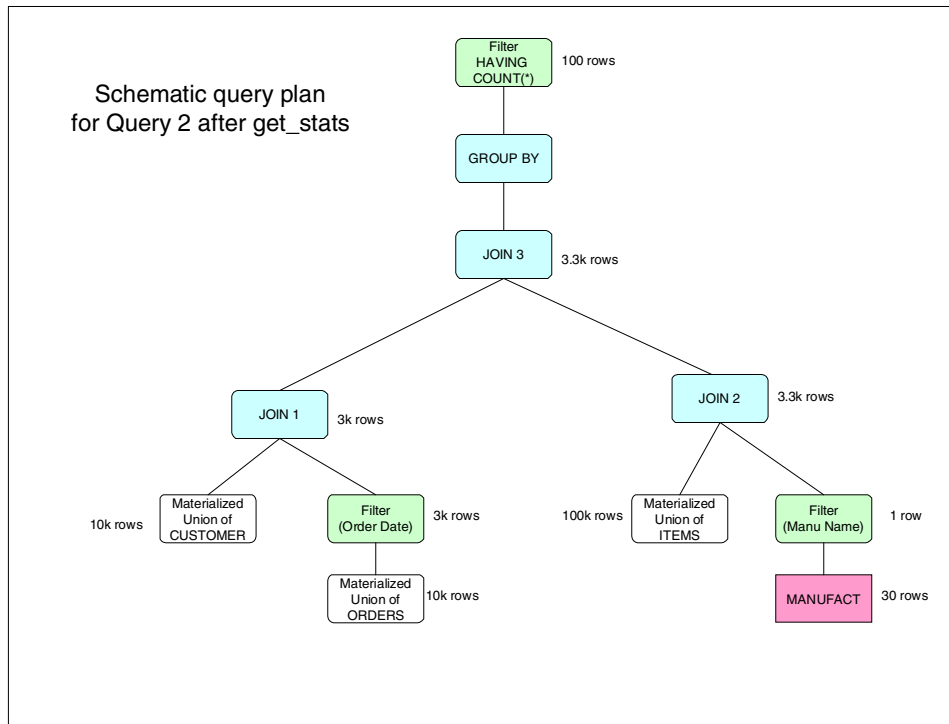


Figure 7-18 Schematic query plan for Query 2 after use of get_stats utility

The first join is between the materialized union of customer table (10 k) across all data sources and the qualifying rows of the materialized union of the orders table (3 k). Each qualifying row in the orders table will join with only one row in the customer table, giving an interim result table of 3 k rows. The second join is between the materialized union of orders table (100 k) and the qualifying rows of the manufact table. The predicate on manu_name = 'Stephani Inc' provides a 1-in-30 filtering effect, so the interim results table for join 2 has 3.3 k (100k/30) rows. Join 3 is performed between the interim result tables for joins 1 and 2. For each row in interim result table 2 there will be one qualifying row in interim result table 1, giving a result of 3.3 k rows for join 3.

The better statistics for the Red Brick data source have given us an access path that we can see will not lead to performance problems as all the interim result tables are of a manageable size. We verified this in practice as well. After using the **get_stats** utility, Query 2 ran in about 13 seconds in our environment. Prior to the use of the **get_stats** utility, this query had still not completed in one hour.

7.6 Optimizer hints in query text

As stated earlier in 1.1, “The challenge of information integration” on page 14, one of the objectives of data federation is to isolate the end user or application from the complexities of the underlying database schemas. Both of the data federation tools used in this publication undoubtedly score well against that objective.

However, there will still be occasions where a knowledge of the underlying environment, and the subsequent incorporation of that knowledge into the query text, can further assist with query performance. This type of information is sometimes called an *optimizer hint*.

Let us see how this can be applied to our case study. Consider our sample Query 2, which joins the federated views on customer, orders, and items (refer to 6.1.2, “Query 2 - Federated views” on page 163). Now refer to the information about data distribution contained within , “Populating the databases” on page 42.

If we study this information carefully, we can tell that because of our data distribution, the join in Query 2 will match customers in Region 1 *only with* orders from Region 1 *and* items in Region 1. Likewise with data from the other regions. So any attempt by the optimizer to join a customer row in the data source for Region 1 with an order row in a data source for any other region is pointless, as we know that no matching row can ever exist. The same rules apply to joins between the orders and items tables.

If we can translate this information into our query as an optimizer hint, we can save the optimizer a significant amount of work. Of course if we are *correct* in our analysis of the data structures, there should be *no difference* in the results set produced. We are simply optimizing the same query request by passing this extra information to the optimizer.

To illustrate this, we altered the federated views on customer, orders, and items, and the text of Query 2 to pass this hint about the data distribution to the DB2 optimizer. To do this we introduced the optimizer to the missing information concerning the region code. Note that the region code is *not* being added to the data sources themselves. It is simply being manufactured at query execution time through the view definitions.

Example 7-2 shows the altered view of the federated customer tables.

Example 7-2 Modified federated CUSTOMER view with REGION code

```
CREATE VIEW DB2W_JN7X_CUSTOMER AS
SELECT 'REGION7' AS REGION, CUSTOMER_NUM, FNAME, LNAME, COMPANY,
      ADDRESS1, ADDRESS2, CITY, STATE, ZIPCODE, PHONE
FROM DB2DBAW.DB2L_CUSTOMER
```

```

UNION ALL

SELECT 'REGION5' AS REGION, CUSTOMER_NUM, FNAME, LNAME, COMPANY,
      ADDRESS1, ADDRESS2, CITY, STATE, ZIPCODE, PHONE
FROM DB2DBAW.DB2W_CUSTOMER

UNION ALL

SELECT 'REGION3' AS REGION, CUSTOMER_NUM, FNAME, LNAME, COMPANY,
      ADDRESS1, ADDRESS2, CITY, STATE, ZIPCODE, PHONE
FROM DB2DBAW.ORAW_CUSTOMER

UNION ALL

SELECT 'REGION1' AS REGION, CUSTOMER_NUM, FNAME, LNAME, COMPANY,
      ADDRESS1, ADDRESS2, CITY, STATE, ZIPCODE, PHONE
FROM DB2DBAW.IFXL_CUSTOMER

UNION ALL

SELECT 'REGION2' AS REGION, CUSTOMER_NUM, FNAME, LNAME, COMPANY,
      ADDRESS1, ADDRESS2, CITY, STATE, ZIPCODE, PHONE
FROM DB2DBAW.IFXW_CUSTOMER

UNION ALL

SELECT 'REGION6' AS REGION, CUSTOMER_NUM, FNAME, LNAME, COMPANY,
      ADDRESS1, ADDRESS2, CITY, STATE, ZIPCODE, PHONE
FROM DB2DBAW.RBWL_CUSTOMER

UNION ALL

SELECT 'REGION4' AS REGION, CUSTOMER_NUM, FNAME, LNAME, COMPANY,
      ADDRESS1, ADDRESS2, CITY, STATE, ZIPCODE, PHONE
FROM DB2DBAW.XPSL_CUSTOMER;

```

The federated orders and items views were also altered in similar ways. Samples of each are shown in Example 7-3 and Example 7-4 on page 214. For brevity, only extracts of the view definitions are shown.

Example 7-3 Extract from modified federated ORDERS view with REGION code

```

CREATE VIEW DB2W_JN7X_ORDERS AS

SELECT 'REGION7' AS REGION, ORDER_NUM, ORDER_DATE, CUSTOMER_NUM, SHIP_INSTRUCT,
      BACKLOG, PO_NUM, SHIP_DATE, SHIP_WEIGHT, SHIP_CHARGE, PAID_DATE
FROM DB2DBAW.DB2L_ORDERS

```

```

UNION ALL

SELECT 'REGION5' AS REGION, ORDER_NUM, ORDER_DATE, CUSTOMER_NUM, SHIP_INSTRUCT,
      BACKLOG, PO_NUM, SHIP_DATE, SHIP_WEIGHT, SHIP_CHARGE, PAID_DATE
FROM DB2DBAW.DB2W_ORDERS

UNION ALL

SELECT 'REGION3' AS REGION, ORDER_NUM, ORDER_DATE, CUSTOMER_NUM, SHIP_INSTRUCT,
      BACKLOG, PO_NUM, SHIP_DATE, SHIP_WEIGHT, SHIP_CHARGE, PAID_DATE
FROM DB2DBAW.ORAW_ORDERS

UNION ALL
...

```

Example 7-4 Extract from modified federated ITEMS view with REGION code

```

CREATE VIEW DB2W_JN7X_ITEMS AS

SELECT 'REGION7' AS REGION, ITEM_NUM, ORDER_NUM, STOCK_NUM,
      MANU_CODE, QUANTITY, TOTAL_PRICE
FROM DB2DBAW.DB2L_ITEMS

UNION ALL

SELECT 'REGION5' AS REGION, ITEM_NUM, ORDER_NUM, STOCK_NUM,
      MANU_CODE, QUANTITY, TOTAL_PRICE
FROM DB2DBAW.DB2W_ITEMS

UNION ALL

SELECT 'REGION3' AS REGION, ITEM_NUM, ORDER_NUM, STOCK_NUM,
      MANU_CODE, QUANTITY, TOTAL_PRICE
FROM DB2DBAW.ORAW_ITEMS

UNION ALL
...

```

Now we can alter our query to tell the optimizer the missing information about our data distribution. We only wish the optimizer to attempt to join rows in the customer, orders, and items tables from matching regions. See the modified text for sample Query 2 in Example 7-5.

Example 7-5 Modified SQL for Query 2 including optimizer hint for REGION code

```

select c.customer_num, c.company, o.order_num,
      sum(i.total_price) as total_manu_rev
from db2w_jn7x_customer c, db2w_jn7x_orders o, db2w_jn7x_items i,

```



```
db2w_manufact m
where o.order_date > '12/25/2001'
and m.manu_name = 'Stephani Inc'
and m.manu_code = i.manu_code
and o.order_num = i.order_num
and o.region = i.region
and o.customer_num = c.customer_num
and o.region = c.region
group by c.customer_num, c.company, o.order_num
having count (*) >= 5
order by 1;
```

This modified Query 2 with optimizer hints produces the same result as the original Query 2, but with a *50 percent saving* in elapsed time and a *60 percent saving* in CPU time at the federated server.

These examples clearly demonstrate the impact of using optimizer hints. When there is additional information available regarding the elements of a query, both performance and resource utilization can be significantly improved.



Hints and tips

This chapter provides several hints and tips that may help as you begin implementing your federated database environment. These hints and tips document situations we encountered during this project. We have documented them to help you in the event that you encounter the same issues while working with your data federation implementation. We provide workaround solutions where possible to make your implementation easier and faster.

8.1 Federated inserts with Informix

We discovered a problem while attempting to perform federated inserts with Informix databases. This issue was encountered when we attempted to INSERT data into Informix target table A using a SELECT data from Informix source table B. If the two tables, target and source, are on different databases the operation may fail with error -400. See Example 8-1, which depicts the SQL statement used and the error code received.

Example 8-1 Error -400 reported during insert/select

```
insert into target_table select * from source_table;
```

```
SQL1822N Unexpected error code "-400" received from data source "SOURCE".  
Associated text and tokens are "Fetch attempted on unopen cursor."  
SQLSTATE=560BD
```

After performing a few tests, we discovered that if the source table and the target table are in the same Informix database everything works fine. This is because the statement is pushed down to that Informix database and executed there. However, if the two tables are in two different Informix databases, the SQL statement cannot be pushed down. So, to perform the operation DB2 Information Integrator (II) uses two different database connections.

A SELECT cursor is created in CONNECTION ONE to fetch the rows from the source table. An INSERT cursor is created in CONNECTION TWO to insert data into the target table. After the first fetch (of 32 K) with the SELECT cursor, DB2 II switches to CONNECTION TWO (that has the INSERT cursor) and performs the insert of the data into the target table. So far so good. Then DB2 II switches to the SELECT cursor to fetch the next buffer of data, but while still using CONNECTION TWO. The operation fails with error -400, Fetch attempted on unopen cursor.

The SELECT cursor is not open because DB2 II is using the wrong connection. It did not switch to CONNECTION ONE, and is still working with CONNECTION TWO. This issue was reported to DB2 II development and a fix was quickly developed. We have tested it, and it works. That fix is currently planned to be available in FixPak 4 for DB2 V8.1.

A work around to the problem, in the interim, is to change the SQL statement by adding an ORDER BY clause, as depicted in Example 8-2 on page 219. The ORDER BY clause will cause DB2 II to first fetch all the rows from the source table and do the appropriate sorting. Then, once the fetch is completed, the data is all inserted into the target table. Therefore, by following this strategy, there is

no need for switching between multiple cursors and the problem is not manifested. Until the formal FixPak is available, our workaround can be used.

Example 8-2 Workaround for error -400 during insert/select

```
insert into target_table  
select * from source_table order by 1;
```

8.2 Using DATE columns for a Union operation

Using Informix Enterprise Gateway Manager, we encountered problems while creating a VIEW on the result of a Union operation between tables in Oracle and Informix when applying the DATE function in the Oracle column to make it compatible to the Informix data type. When we tried to filter by the date column in the VIEW, we received error 29044. The sequence is shown below:

```
(IFX): CREATE TABLE TABA (col1 date);  
(ORA): CREATE TABLE TABB (col1 date);  
(IFX): CREATE VIEW tab as  
                SELECT col1 from taba  
                UNION  
                SELECT date(col1) from tabb;  
  
(IFX): SELECT * from TAB where col1 > '01/01/2001')
```

The select above returns error 29044.

However, if TABB is created in Informix as a datetime column, you are able to create a view over the UNION and SELECT using the predicate WHERE COL1 > '01/01/2001'.

8.3 Use of current schema with DB2 Interactive Explain

For each data source, we defined separate user IDs for end user access to tables. The topic of defining user IDs is presented in detail in “User IDs” on page 45, in publication SG24-7032, “IBM Informix: Integration through Data Federation”.

There is a feature of DB2 called the **set current schema** statement. It can be used to set the current schema for the session so that you need not specify the fully qualified name for database objects.

The Command Center allows you to execute **set current schema** for the query execution; therefore you do not need to provide the schema name for database

objects. However, you are unable to *explain* the SQL statement through the graphical interface if the tables involved in the query are from a different schema, even though you have the schema explicitly set via the **set current schema** command. Therefore, you must provide the table name in the form *schema_name.table_name* so that the query plan can be produced.

8.4 DB2 problem determination series tutorial

DB2 Technical Support has put together a great online tutorial series to help users supporting and developing with DB2. The series introduces many techniques for supporting the software including installation, performance, connectivity, and troubleshooting tools that you use while supporting DB2, or interacting with your technical support representative. It is an invaluable resource that has many benefits. The DB2 Problem Determination Tutorial Series can be found on the Web at:

http://www-3.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/tutorial_main.d2w/toc

8.5 Losing connection to a data source

You can open a connection to DB2 and select, for example, from three data sources. Let's call them A, B, and C. Then three connections from DB2 to the remote data sources will be established by DB2 Information Integrator. In the event that one of the connections is lost, let us say to B, the DB2 sessions to A and C are still accessible. But, to gain access again to remote data source B, the DB2 application needs to reconnect. If you simply try again to access data from data source B, the following error is returned:

```
SQL1822N Unexpected error code "XXXX" received from data source "XXXX".  
Associated text and tokens are "Network connection is broken."
```

There is no attempt to reconnect to a remote data source within an existent DB2 application session in the event of a data source failure. The application receiving the error must TERMINATE and issue a new CONNECT.

8.6 Using views versus synonyms with data federation

When using front-end tools that need column definitions to allow you to draw queries based on a GUI interface (for example, to drag and drop columns to establish relationships and form the queries), you cannot use synonyms if the base table is remote to the database where the front-end tool is connected. This

is because the column definitions are not present in the database catalog where the synonym resides. You may want to consider using a VIEW rather than a synonym. When a view is created, based on a remote data source, the column definitions are imported into the local database catalog making. This makes it possible for such front-end tools to operate as designed.

Another consideration is the fact that when the query is issued against a synonym, only the columns selected in the query are included in the data set returned by the remote source. However, when issued against views based on the remote tables, if there is a predicate in a column that belongs to a view, the data may contain all the columns that are part of the VIEW construct. Therefore the data set may contain more columns than the ones specified in the query, thus increasing the volume of data being sent across the network. Consider the query below:

```
select fname, lname,o.order_num,i.item_num, m.manu_name
  from db2w_customer c, db2w_orders o, db2w_items i, manufact m
  where o.customer_num = c.customer_num
        and c.fname      = 'Julio'
        and o.order_num  < 10000
        and i.order_num  = o.order_num
        and i.manu_code  = m.manu_code
```

If tables db2w_customer and db2w_orders and db2w_items are declared as synonyms, only the columns requested in the original query are returned by the fetch, thus resulting in a data set of *73 bytes*. In Example 8-3 you see the query fragment being pushed down to the remote source and a data row sample fetched back to the application.

Example 8-3 Data row fetched when query is issued against synonyms

```
"SELECT x0.fname,x0.lname,x0.customer_num,x1.order_num,x1.customer_num,x
2.item_num,x2.order_num,x2.manu_code FROM "DB2DBAW".customer x0,"DB2DBAW
".orders x1,"DB2DBAW".items x2 WHERE (((x0.fname='Julio') AND ((x1.custo
mer_num=x0.customer_num) AND (x1.order_num<10000))) AND ((x2.order_num=x
1.order_num) AND (x2.order_num<10000)))"
```

** ODBC Output SQLDA with row data **

```
sqlvar count: 8
#   col_name          sqltype  ctype      length scale cnull inull
datalength col_data
1   FNAME              CHAR     CHAR        15     0     1     0
15 "Julio             "
2   LNAME              CHAR     CHAR        15     0     1     0
15 "Bishop            "
```

3	CUSTOMER_NUM	INTEGER	LONG	10	0	0	0
4	10373						
4	ORDER_NUM	DECIMAL	CHAR	5	0	0	0
4	"1075"						
5	CUSTOMER_NUM	INTEGER	LONG	10	0	0	0
4	10373						
6	ITEM_NUM	SMALLINT	SHORT	5	0	0	0
2	1671						
7	ORDER_NUM	INTEGER	LONG	10	0	0	0
4	1075						
8	MANU_CODE	CHAR	CHAR	3	0	0	0
3	"ZZZ"						

Now look at Example 8-4 below. If the very same query is issued against views over the same tables in the remote data source, then all columns from all views involved in the query are fetched in the data set although the select statement is selecting only fname, lname, o.order_num, i.item_num. Now it results in a data set of 286 bytes (as opposed to 73 bytes when issued against synonyms).

Note: Views on remote tables were created based on a select of all columns on the remote table (SELECT * FROM remote_table).

Example 8-4 Data row fetched when query is issued against views

```
"SELECT x0.customer_num,x0.fname,x0.lname,x0.company,x0.address1,x0.address2,x0.city,x0.state,x0.zipcode,x0.phone,x1.order_num,x1.order_date,x1.customer_num,x1.ship_instruct,x1.backlog,x1.po_num,x1.ship_date,x1.ship_weight,x1.ship_charge,x1.paid_date,x2.item_num,x2.order_num,x2.stock_num,x2.manu_code,x2.quantity,x2.total_price FROM "DB2DBAW".customer x0,"DB2DBAW".orders x1,"DB2DBAW".items x2 WHERE (((x0.fname='Julio') AND ((x1.customer_num=x0.customer_num) AND (x1.order_num<10000))) AND ((x2.order_num=x1.order_num) AND (x2.order_num<10000)))"
```

** ODBC Output SQLDA with row data **

```
sqlvar count: 26
# col_name sqltype ctype length scale cnull inull
datalength col_data
1 CUSTOMER_NUM INTEGER LONG 10 0 0 0
4 10373
2 FNAME CHAR CHAR 15 0 1 0
15 "Julio"
3 LNAME CHAR CHAR 15 0 1 0
15 "Bishop"
4 COMPANY CHAR CHAR 20 0 1 0
20 "Inforte, Corp"
```



```

5 ADDRESS1 CHAR CHAR 20 0 1 0
20 "2803 Pinckard Ave "
6 ADDRESS2 CHAR CHAR 20 0 1 0
20 "Level 210 "
7 CITY CHAR CHAR 15 0 1 0
15 "Dallas "
8 STATE CHAR CHAR 2 0 1 0
2 "TX"
9 ZIPCODE CHAR CHAR 5 0 1 0
5 "75219"
10 PHONE CHAR CHAR 18 0 1 0
18 "(205) 410-8486 "
11 ORDER_NUM DECIMAL CHAR 5 0 0 0
4 "1075"
12 ORDER_DATE DATE CHAR 10 0 1 0
10 "2002-08-02"
13 CUSTOMER_NUM INTEGER LONG 10 0 0 0
4 10373
14 SHIP_INSTRUCT CHAR CHAR 40 0 1 0
40 "Leave around side "
15 BACKLOG CHAR CHAR 1 0 1 0
1 "n"
16 PO_NUM CHAR CHAR 10 0 1 0
10 "S044AN1F "
17 SHIP_DATE DATE CHAR 10 0 1 0
10 "2001-03-08"
18 SHIP_WEIGHT DECIMAL CHAR 8 2 1 0
7 "6380.18"
19 SHIP_CHARGE DECIMAL CHAR 6 2 1 0
6 "456.42"
20 PAID_DATE DATE CHAR 10 0 1 0
10 "2002-03-25"
21 ITEM_NUM SMALLINT SHORT 5 0 0 0
2 1671
22 ORDER_NUM INTEGER LONG 10 0 0 0
4 1075
23 STOCK_NUM SMALLINT SHORT 5 0 0 0
2 9503
24 MANU_CODE CHAR CHAR 3 0 0 0
3 "ZZZ"
25 QUANTITY SMALLINT SHORT 5 0 1 0
2 18
26 TOTAL_PRICE DECIMAL CHAR 8 2 1 0
7 "4914.45"

```

===== Time: 2003-07-12 14:21:12.000000 =====



A

DB2 II and Informix EGM function summary

This appendix contains a summary of information regarding DB2 Information Integrator and Informix Enterprise Gateway Manager. The intent is not to compare them for purposes of determining which is the better product. They are both excellent products and could be used in most any implementation. The intent is only to provide information that will help in making a decision as to which would satisfy your requirements in a particular situation. It may be that you will use both at one time or another.

The function summary is at a high level and only based on specific functions and features that were evaluated during the development of this publication. This should help in the positioning of the products and determining where each fits best.

Data federation summary table

We provide below, in Table A-1, a list of functions and features to be considered when implementing a federated database environment. With each function or feature we give a brief evaluation, based on our experience, of the capabilities of DB2 Information Integrator and Informix Enterprise Gateway Manager. It will provide you with a good high-level overview of their capabilities.

Table A-1 *Functions and features*

Functions and features	Informix Enterprise Gateway Manager	DB2 Information Integrator
Provides optimization in a federated environment.	Provides pushdown of predicates to joins to relational data sources. It does not evaluate database statistics or network capabilities. Pushdown joins with IDS and/or XPS uses Informix I-STAR. It does not support the use of MQTs.	DB2 II provides robust optimization that enables maximum pushdown of predicates and joins to relational data sources. The optimizer evaluates database statistics and network capabilities in determining the lowest cost solution. It will also automatically use MQTs if determined that they will improve performance.
Supports both relational and nonrelational data sources.	Yes.	Yes.
Supports structured and unstructured data.	Yes.	Yes.
Supports pushdown of predicates.	Yes.	Yes.
Supports pushdown of joins to data sources.	Yes, with all relational data sources.	Yes, with all relational data sources.
Supports output of data in an XML format.	Yes.	Yes.

Functions and features	Informix Enterprise Gateway Manager	DB2 Information Integrator
Can replicate data to and from relational sources.	No. However, Enterprise Replication capability is included in Informix Dynamic Server— but only to and from other IDS instances.	Yes, the DB2 II Replication Edition supports movement of data between mixed relational data sources. DB2® Universal Database, Informix Dynamic Server, Microsoft® SQL Server, Oracle, Sybase SQL Server, and Sybase Adaptive Server Enterprises are supported as replication sources and targets. Informix Extended Parallel Server and Teradata are supported as replication targets.
Provides native client interface access to data sources.	Native to Informix IDS and XPS. Uses ODBC for others.	Yes, to many through the use of wrappers. For those not natively supported, it uses ODBC.
Is packaged with a database server.	No, can reside independently on a client.	Yes, Standard Edition has a limited-use license of DB2 V8.1 for use of the DB2 catalog only. Enterprise Edition is packaged with a full-use license of DB2.
Supports 2-phase commit protocol.	Yes.	No.
Support for scrolling cursors.	Yes.	Yes.
Can send DDL to data sources.	No.	Yes.
How are data sources accessed?	By synonym or view name.	By DB2 II nicknames.
Can execute SQL operations at the remote data source that are not natively supported by it.	No.	Yes. Valuable to compensate for SQL limitations of the data source being accessed.

Functions and features	Informix Enterprise Gateway Manager	DB2 Information Integrator
Uses a global catalog to store information about the federated environment.	No.	Yes.
Has a query optimizer.	Yes.	Yes, it generates local and remote access plans for processing a query fragment, based on resource cost.
Supports SQL pass-through.	Yes.	Yes.



B

Nonrelational wrappers

This appendix provides two examples of access to nonrelational data sources using DB2 Information Integrator wrappers. They are Microsoft Excel and XML. The ability to access nonrelational data sources is also supported by Informix Enterprise Gateway Manager, as long as there is ODBC support for those data sources.

Using DB2 Information Integrator you can process SQL statements that query data in nonrelational data sources, such as Excel spreadsheets, flat files, XML, and others. Also, you can do so as if it were contained in an ordinary relational table or view. In this section we demonstrate the use of the Excel and XML wrappers.

Microsoft Excel wrapper

One of the nonrelational modules offered with DB2 Information Integrator is the Excel wrapper. An Excel spreadsheet or workbook is a file with an extension of *xls*. DB2 Information Integrator supports spreadsheets from Excel 97 and Excel 2000 by creating a nickname to map the columns in your Excel spreadsheet to columns in your DB2 federated system, thus making the information available through standard SQL commands as if it were a standard relational data source. Because DB2 Information Integrator holds the file handler to the Excel spreadsheet, multiple users can access it through SQL at the same time. In Example B-1 you see DDLs used to define the wrapper, server, and a nickname to a spreadsheet called EXLW_REGION.

Example: B-1 Excel wrapper and nickname definition

```
CREATE WRAPPER "EXCEL"
  LIBRARY 'db21sx1s.d11'
  OPTIONS (DB2_FENCED 'N'
          );
CREATE SERVER "EXLWIN"
  WRAPPER EXCEL
  OPTIONS);

CREATE NICKNAME DB2DBAW.EXLW_REGION
  (REGION VARCHAR (10) ,
   STATE_CODE VARCHAR (2) ,
   STATE_NAME VARCHAR (20) ,
   SYSTEM VARCHAR (5) )
  FOR SERVER "EXLWIN"
  OPTIONS(RANGE 'A2:D53' , FILE_PATH 'c:\excel\region.xls');
```

Once defined, the nickname can be referenced and the data accessed through regular SQL. In Example B-1 on page 231 you see the results of a SELECT over the nickname EXLW_REGION shown in the DB2 Command Center.

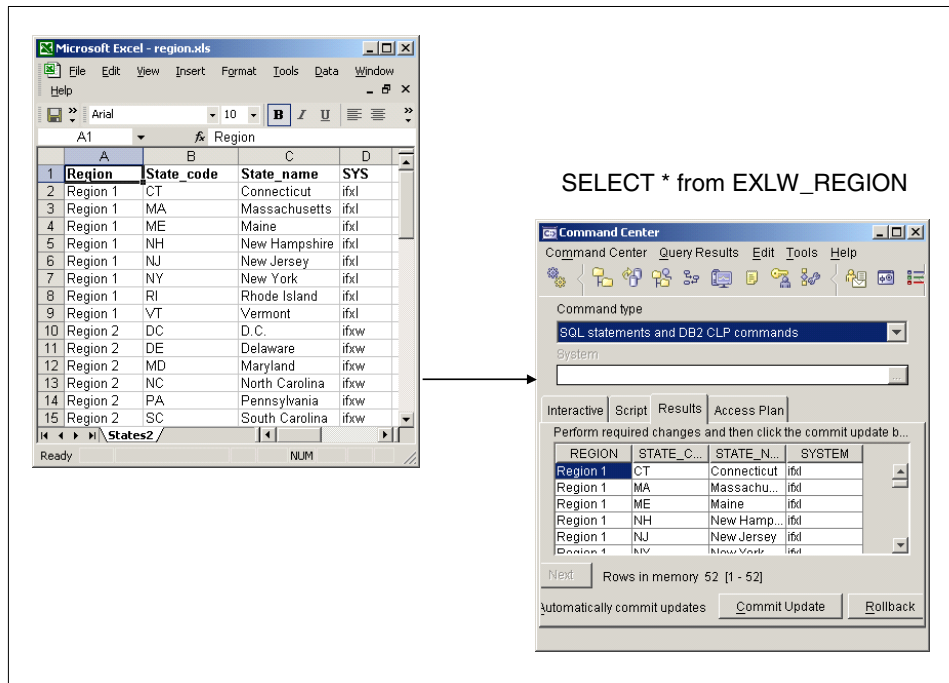


Figure B-1 An Excel spreadsheet accessed through SQL from DB2 Command Center

The query plan shown in Figure B-2 on page 232 is a result of a select between the customers table residing in the Linux server on an Informix database and an Excel spreadsheet accessed via the wrapper. Here is the SQL:

```
SELECT company, state, region
FROM ifxl_customer c, exlw_region r
WHERE c.state_code = r.state;
```

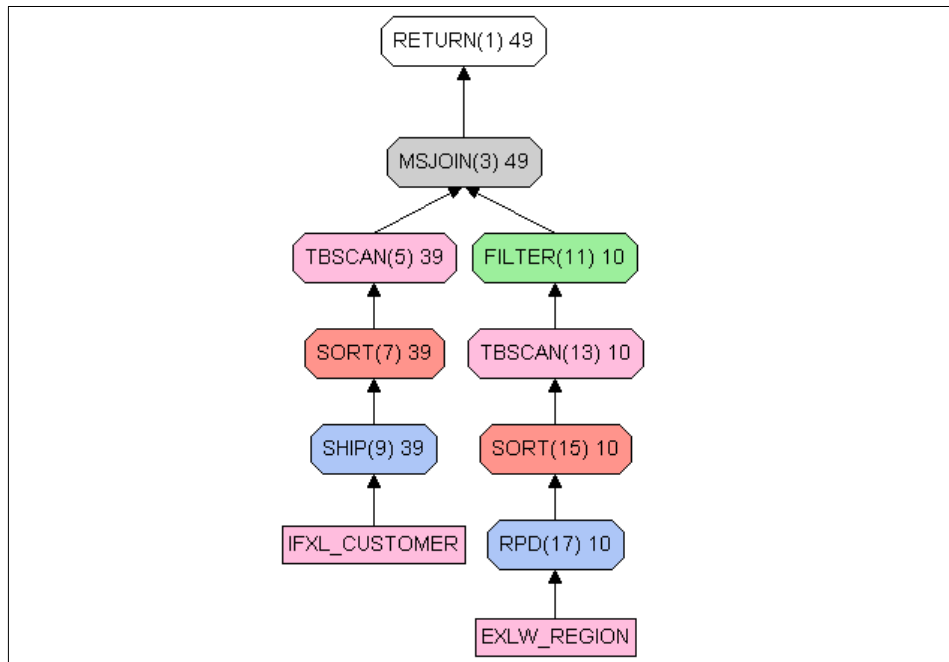


Figure B-2 Query plan produced for JOIN between Informix and Excel data sources

Note: The SHIP operator shows the data being retrieved from the relational data source and the RPD (remote pushdown) operator shows the data being retrieved from nonrelational sources.

XML wrapper

Another nonrelational module offered with DB2 Information Integrator is the XML wrapper. DB2 Extensible Markup Language (XML) is a universal format for structured documents and data, which has been widely used for inter-system communication. XML files have a file extension of *.xml*. Like HTML, XML uses tags for structuring data in a document turning it into *self-describing* data.

DB2 Information Integrator through the XML wrapper enables the use of SQL to query XML data from an external data source into a relational schema. The structure of an XML document is logically equivalent to a relational schema in which the nested and repeating elements are modeled as separate tables with foreign keys. The nicknames that correspond to an XML document are organized into a tree structure in which the child nicknames map to elements that are nested within the element that corresponds to the parent nickname. Child and

parent nicknames are connected by primary and foreign keys that are generated by the wrapper.

The XML wrapper includes support for the following types of data:

- ▶ External XML documents that are stored in a single file
- ▶ Multiple files in a directory path
- ▶ Remote XML files that are referenced with a Uniform Reference Identifier (URI)
- ▶ Relational columns

In Figure B-3 on page 234 a simple xml file with the REGION table content is shown along with the results of a query over the XMLW_REGION nickname created using the XML wrapper. The simple steps are listed in Example B-2.

Example: B-2 Defining XML access using the XML wrapper

```
CREATE WRAPPER "XML"
  LIBRARY 'db21sxml.d11'
  OPTIONS (DB2_FENCED 'N'
          );

CREATE SERVER "XMLWIN"
  WRAPPER XML
  OPTIONS
  );

CREATE NICKNAME DB2DBAW.XMLW_REGION
  (REGION    VARCHAR (48) OPTIONS(XPATH './REGION/text()'   ),
   STATE_CODE VARCHAR (48) OPTIONS(XPATH './STATE_CODE/text()'),
   STATE_NAME VARCHAR (48) OPTIONS(XPATH './STATE_NAME/text()'),
   SYSTEM    VARCHAR (48) OPTIONS(XPATH './SYSTEM/text()'   ))
  FOR SERVER "XMLWIN"
  OPTIONS(XPATH '//row' , FILE_PATH 'c:\xml\region.xml');
```

h

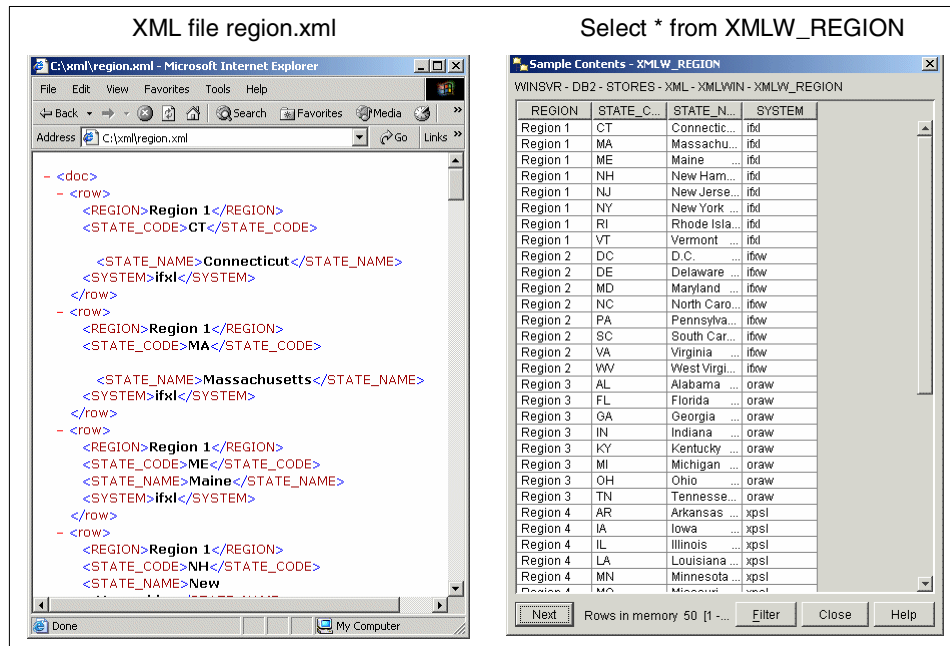


Figure B-3 XML file and query results

The query plan shown in Figure B-4 on page 235 is a result of a select between the customers table residing in the Linux server on an Informix database and the XML file accessed via the wrapper. Here is the SQL:

```
SELECT company, state, region
FROM ifxl_customer c, xmlw_region r
WHERE r.state_code = c.state;
```

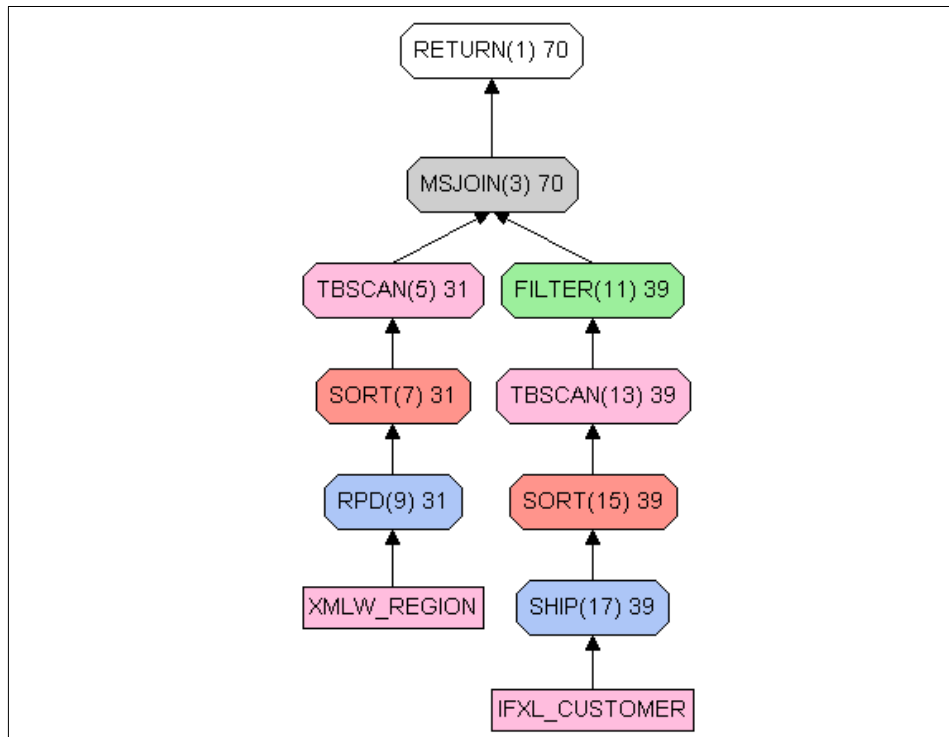


Figure B-4 Query plan produced for JOIN between Informix and XML data sources

Note: The SHIP operator shows the data being retrieved from the relational data source and the RPD (remote pushdown) operator shows the data being retrieved from nonrelational sources.

Glossary

Cluster. A group of records with similar characteristics.

Compensation. The ability of DB2 to process SQL that is not supported by a data source on the data from that data source.

Data Federation. The process of enabling data from multiple heterogeneous data sources to appear as if it is contained in a single relational database. Can also be referred to “distributed access”.

Database Instance. A specific independent implementation of a DBMS in a specific environment. For example, there might be an independent DB2 DBMS implementation on a Linux server in Boston supporting the Eastern offices, and another separate and independent DB2 DBMS on the same Linux server supporting the western offices. They would represent two instances of DB2.

DataBlades. These are program modules that provide extended capabilities for Informix databases, and are tightly integrated with the DBMS.

Dynamic SQL. SQL that is interpreted during execution of the statement.

Extenders. These are program modules that provide extended capabilities for DB2, and are tightly integrated with DB2.

Federated server. Any DB2 server where the DB2 Information Integrator is installed.

Federation. Providing a unified interface to diverse data.

Gateway. A means to access a heterogeneous data source. It can use native access or ODBC technology.

Instance. A complete database environment.

I-STAR. An Informix technology to enable data federation between Informix relational data sources.

Materialized Query Table. A table where the results of a query are stored for later reuse.

Nickname. An identifier that is used to reference the object located at the data source that you want to access.

Optimization. The capability to enable a process to execute and perform in such a way as to maximize performance, minimize resource utilization, and minimize the process execution response time delivered to the end user.

Pass-through. The act of passing the SQL for an operation directly to the data source without being changed by the federation server.

Pushdown. The act of optimizing a data operation by pushing the SQL down to the lowest point in the federated architecture where that operation can be executed. More simply, a pushdown operation is one that is executed at a remote server.

Shared-nothing. A data management architecture where nothing is shared between processes. Each process has its own processor, memory, and disk space.

Static SQL. SQL that has been compiled prior to execution. Typically provides best performance.

Type mapping. The mapping of a specific data source type to a DB2 UDB data type.

User mapping. An association made between the federated server user ID and password and the data source (to be accessed) used ID and password.

Virtual database. A federation of multiple heterogeneous relational databases.

Wrapper. The means by which a data federation engine interacts with heterogeneous sources of data. Wrappers take the SQL that the federation engine uses and map it to the API of the data source to be accessed. For example, they take DB2 SQL and transform it to the language understood by the data source to be accessed.

xtree. A query-tree tool that allows you to monitor the query plan execution of individual queries in a graphical environment.

Abbreviations and acronyms

MM			
API	Application Programming Interface	HPL	High Performance Loader
AST	Application Summary Table	I/O	Input/Output
BLOB	Binary Large Object	IBM	International Business Machines Corporation
CLI	Call Level Interface	IDS	Informix Dynamic Server
CLOB	Character Large Object	II	Information Integrator
CPU	Central Processing Unit	IMS	Information Management System
DB2 UDB	DB2 Universal DataBase	ISAM	Indexed Sequential Access Method
DB2II	DB2 Information Integrator	ISM	Informix Storage Manager
DBMS	DataBase Management System	ITSO	International Technical Support Organization
DCM	Dynamic Coserver Management	JDBC	Java DataBase Connectivity
DDL	Data Definition Language	JE	Java Edition
DLL	Dynamically Linked Library	LDAP	Lightweight Directory Access Protocol
DML	Data Manipulation Language	Mb	Mega bits
DRDA	Distributed Relational Database Architecture	MB	Mega Bytes
DSA	Dynamic Scalable Architecture	MQT	Materialized Query Table
DSN	Data Source Name	ODBC	Open DataBase Connectivity
DSS	Decision Support System	OLAP	OnLine Analytical Processing
EDA	Enterprise Data Architecture	OLE	Object Linking and Embedding
EGM	Enterprise Gateway Manager	OLTP	OnLine Transaction Processing
ER	Enterprise Replication	ORDBMS	Object Relational DataBase Management System
ESE	Enterprise Server Edition	RBW	Red Brick Warehouse
ETL	Extract Transform and Load	RDBMS	Relational DataBase Management System
FP	Fix Pack	SPL	Stored Procedure Language
FTP	File Transfer Protocol	SQL	Structured Query
Gb	Giga bits	TMU	Table Management Utility
GB	Giga Bytes		
HDR	High availability Data Replication		

UDB	Universal DataBase
UDB	Universal DataBase
UDF	User Defined Function
UDR	User Defined Routine
URL	Uniform Resource Locator
VLDB	Very Large DataBase
VSAM	Virtual Sequential Access Method
VTI	Virtual Table Interface
WWW	World Wide Web
XBSA	X-Open Backup and Restore APIs
XML	eXtensible Markup Language
XPS	Informix eXtended Parallel Server

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 242. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *DB2 UDB Exploitation of the Windows Environment*, SG24-6893
- ▶ *DB2 UDB Evaluation Guide for Linux and Windows*, SG24-6934
- ▶ *DB2 UDB WebSphere Performance Tuning Guide*, SG24-6417
- ▶ *Using Informix Dynamic Server with WebSphere*, SG24-6948

Other publications

These publications are also relevant as further information sources:

- ▶ *IBM DB2 Information Integrator Federated Systems Guide*, SC18-7364
- ▶ *IBM DB2 UDB Administration Guide: Performance V8*, SC09-4821
- ▶ *IBM Informix Backup and Restore Guide, Version 9.4*, G251-1240
- ▶ *IBM Informix High-Performance Loader User's Guide, Version 9.4*, G251-1255
- ▶ *Informix Enterprise Gateway Manager User Manual for WindowsNT*, Version 7.2, 000-8911
- ▶ *The Federation - Database Interoperability Parts I and II*, Articles by Marty Lurie, IBM, April 2003.
- ▶ *Data Integration Through Database Federation*, IBM Systems Journal, Volume 41, November 4, 2002.
- ▶ *Information Integration - Distributed Access and Data Consolidation*, by Dr. Barry Devlin, Information Integration Software Solutions, March 2003

Online resources

These Web sites and URLs are also relevant as further information sources:

- ▶ DB2 Command Reference manual at:
<http://www.ibm.com/software/data/db2/library/>
- ▶ IBM Informix Online documentation site
<http://www.ibm.com/software/data/informix/pubs/library/>
- ▶ IBM Red Brick Warehouse site
<http://www.ibm.com/software/data/redbrick/>
- ▶ IBM Data Management Online documentation site
<http://www.ibm.com/software/data/pubs/library/>
- ▶ IBM DB2 Information Integrator site
<http://www.ibm.com/software/data/integration/db2ii/>
- ▶ IBM Informix Enterprise Gateway Manager site
<http://www.ibm.com/software/data/informix/tools/egm/>
- ▶ IBM DB2 UDB site
<http://www.ibm.com/software/data/db2/udb/>
- ▶ IBM Database and Data Management site
<http://www.ibm.com/software/data/database/>
- ▶ DB2 Problem Determination Tutorial Series
http://www-3.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/tutorial_main.d2w/toc
- ▶ **runstats** and **getstats** utilities for download
<http://www.ibm.com/software/data/integration/db2ii/support.html>

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

A

- access plans 80
- aliases 77
- Architecture
 - co-server 62
 - data federation 17
 - DB2 UDB 49
 - EGM 135
 - Red Brick 66
 - XPS 62
- authenticate 36

B

- Backup and Restore 58
- Backup Services 58
- Binary join 66
- binary large objects 29
- BLOB. See binary large objects
- Brio Explorer 6, 34, 150, 160
- B-TREE 70
- Business intelligence 52

C

- Cache 60
- cardinality 174, 181, 195, 198, 210
- cartesian join 174, 181
- case study 31
- character large objects 29
- Client-server Connectivity 57
- CLOB. See character large objects
- collating sequence 200
- column options 77
- Compensation 80
- configuration 5
- Connection Concentrator 53
- Consolidation 11
- co-server 62

D

- Data consolidation 16, 18
- data federation ix, 1, 3, 7, 16, 18, 20, 24, 47
 - architecture 17

- products 7
- data mart 20
- Data Partitioning 55
- data source name 77, 138, 140
- data source objects 77
- data types 29, 53, 78
 - mapping 114
 - User Defined 59
- data warehousing 22
- Database Server 57
- DataBlade 54
- DB2 Command Center 6, 34, 169
 - interactive explain 169
 - zoom box 170
- DB2 Information Integrator ix, 4, 26, 73
 - Components 73
 - configuration 82
 - create a wrapper 98
 - Define a server 99
 - Define user mapping 102
 - Editions 81
 - Installation 83
 - packaging 81
 - Post-installation 93
- DB2 Life Sciences Data Connect 81
- DB2 Relational Connect 81
- DB2 UDB ix, 3, 47, 49
 - Agents 49
 - Architecture 49
 - Buffer pool 49
 - Deadlock 51
 - instance 37
 - Logs 51
 - Page cleaners 50
 - Prefetchers 50
- Decision Support Systems 48
- Distributed Relational Database Architecture 24
 - DRDA 75
 - wrapper 78
- DRDA. See Distributed Relational Database Architecture
- DSS. See Decision Support Systems
- Dynamic Scalable Architecture 55

E

EDA/SQL 134
EGM. See Enterprise Gateway Manager
Enterprise Gateway Manager
 architecture 135
 Authentication 143
 authorization 137
 configuration 139
 Configuring Informix 147
 connection support 134
 considerations 150
 coordinator 135
 gateway service 135
 Informix 4GL 151
 Object names 150
 Pre-installation 139
 Scrolling cursor 134
 statistics 151
 synonyms 150
 System requirements 134
 Troubleshooting 154
 User mapping 134
 views 150
Enterprise Replication 57
environment variables 94
ETL 18

F

federated ix
 data environment 3, 8
 database 35, 79
 coordinating server 37
 information environment 9
 server 37
 Informix 37
 systems 74
Fragmentation 57
Function mapping 78
Fuzzy Checkpoints 60

H

HDR. See High Availability
heterogeneous data 21
heterogeneous data sources 2, 47
High Availability 55, 57
High Performance Loader 58
hints and tips xi, 217

I

IMS. See Information Management System
Index specifications 79
information integration 2, 7, 13, 15, 52
Information Management System 24
Informix
 4GL 151
 Dynamic Server ix, 38, 47, 53
 Enterprise Gateway Manager ix, 4–5, 26
 Extended Parallel Server ix, 38, 47
 federated server 37
 OnLine 134
 Stores Demo x, 8
 wrappers 90
Informix federated server
 IDS instance 37
integrated information environment ix
Integration 11
I-STAR 4, 9, 27, 64, 135

J

join operation 20
joins 23

L

latency 205
LDAP 75
Linux 8, 37, 182
Lotus Notes 75

M

Materialized Query Table 10, 20, 23, 204
materialized union 180
maximal pushdown 199
Microsoft
 Access 75
 Excel x, 3, 48, 70
 Windows/2000 8
MQT. See Materialized Query Table
Multi threaded Implementation 56
Multimedia extensibility 52

N

naming conventions 28
nicknames 77
 creating 97
nonrelational xi

data sources xi, 10, 47, 78

O

Object-Relational 53
ODBC 17, 134
ODBC data source 146
OLTP 48
Omniback 58
ON-Bar 58
on-demand 13
Open Database Connectivity. See ODBC
optimization 3, 8
optimizer hint 212
Oracle ix, 3, 47, 134, 152
 date 152
 datetime 152
ORDBMS 53

P

Partitioning 57
pass-through 81
Project Environment 31
 case study 31, 41
 Client Configuration 34
 Database identifiers 44
 Database servers 44
 DB2 connectivity 35
 DB2 Information Integrator 35
 Enterprise Gateway Manager 36
 Informix Database connectivity 36
 naming conventions 43
 objectives 31
 operating environment 33
 populate the database tables 40
 Red Brick Warehouse 45
 schema 38
 Stores Demo 42
 Stores Demo database 38
project environment x
pushdown 80, 159, 195, 198

Q

query
 fragments 80
 optimization 5, 10
 optimizer 80
 rewrite 159

query fragments 184

R

Red Brick ix, 47, 65, 134, 145, 189
 Aggregate maintenance 69
 Linear scalability 69
 STARIndex 66
 STARJoin 67
 Table Management Utility 70
 TARGETIndex 68
 TARGETJoin 68
Red Brick Warehouse. See Red Brick
Redbooks Web site 242
 Contact us xv
refresh options 206
relational data sources 47, 79
Reliability 52

S

sample queries 160
 Query 1 160
 Query 2 163
 Query 3 167
Scalability 52
schema x, 8, 28, 32
 Stores Demo 40
Scroll cursors 153
server options 194
 collating sequence 200
 maximal pushdown 199
Server Studio Java Edition 6, 34, 182
Server Studio JE. See Server Studio Java Edition
Shared Memory 56
Snowflake Schema 64
STAR schema 67
STARIndex 67
STARJoin 67
Storage Manager 58
Stores Demo database 6
 Stores Demo 32, 168
structured data 25
summary tables xi, 20
SuSE Linux 44
Sybase 134
synonyms 38, 138, 184
system catalog 79, 152
system data source 139
systems architecture x

T

test environment 4
Tivoli Storage Manager 58
Transaction Processing Council 20
tuning 194
two-phase commit 8, 21, 27, 190

U

UDR. See User Defined Routines
unstructured data 25
User Defined Routines 54, 59
user mapping 77, 102, 144
userid 45

V

views 38, 138

W

Web-enablement 52
wrappers 2, 76, 79
 creating 97
 DRDA 78
 Informix 90
 nonrelational 83
 ODBC 179, 195, 197, 199
 relational 84, 88

X

XML x, 52, 70



IBM Informix: Integration Through Data Federation



Redbooks

IBM Informix: Integration Through Data Federation

**Managing
information and
protecting
development assets**

**Integrating
heterogeneous
sources of data**

**Data federation and
join optimization**

This IBM Redbook describes how to create and implement a federated data management environment. That environment can enable access to, and use of, multiple heterogeneous data sources as if they were all resident in the same data management system. We use Informix Dynamic Server (IDS) and DB2 as the primary data sources, along with Informix Extended Parallel Server (XPS), Red Brick Warehouse, Oracle9i, and Microsoft Excel as data sources. We also use data management tools, such as IBM DB2 Information Integrator and Informix Enterprise Gateway Manager, to provide the data federation. For data access tools, we use DB2 Command Center, Server Studio JE for Informix, and Brio Explorer to show data federation in action. With the combined functionality of the Informix and DB2 database management systems and the DB2 Information Integrator, you can implement a very powerful federated data management environment. Informix customers will be well positioned to take advantage of the expanded capabilities for integration through data federation, as well as future enhancements.

**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**

SG24-7032-00

ISBN 0738499897