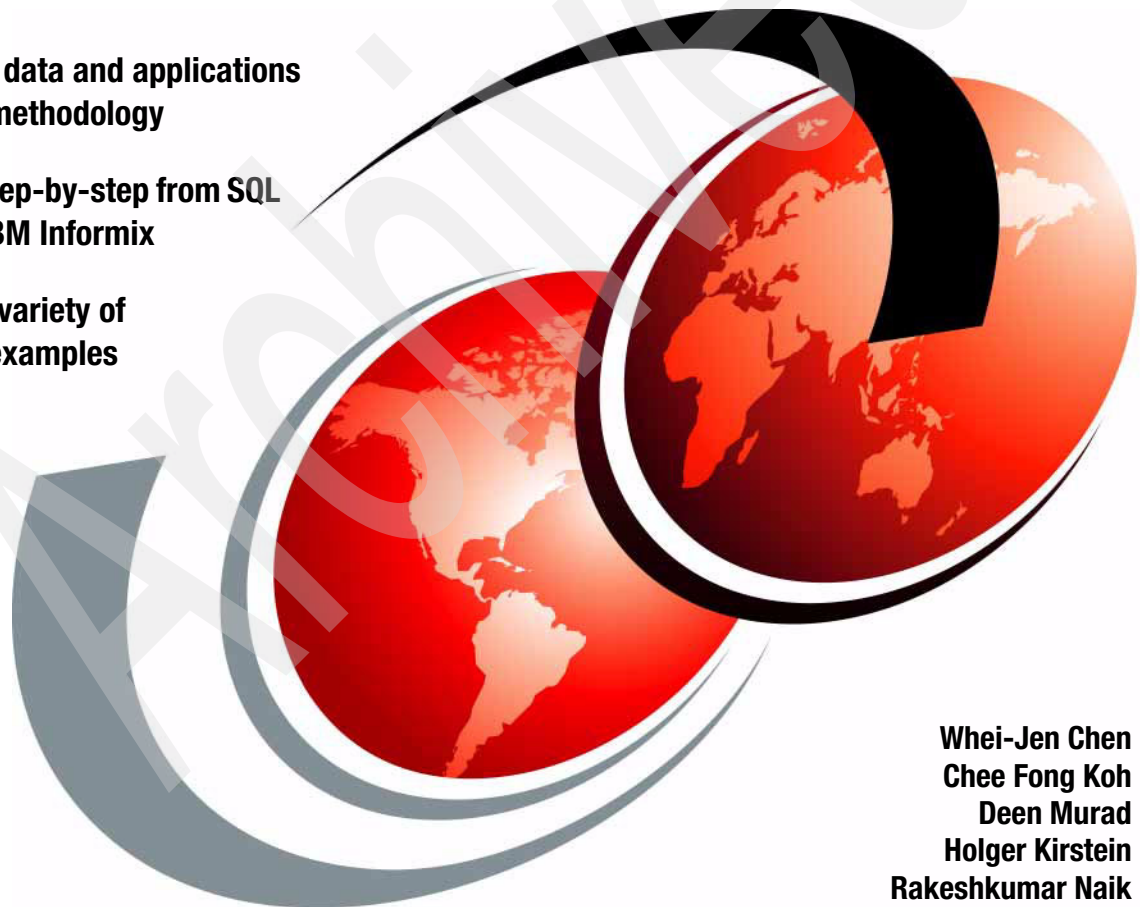


Migrating from Microsoft SQL Server to IBM Informix

Develops a data and applications migration methodology

Migrates step-by-step from SQL Server to IBM Informix

Provides a variety of migration examples



Whei-Jen Chen
Chee Fong Koh
Deen Murad
Holger Kirstein
Rakeshkumar Naik



International Technical Support Organization

**Migrating from Microsoft SQL Server to IBM
Informix**

July 2010

Archived

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

First Edition (July 2010)

This edition applies to IBM Informix Version 11.5 and Microsoft SQL Server 2008.

© Copyright International Business Machines Corporation 2010. 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	xi
Trademarks	xii
 Preface	 xiii
The team who wrote this book	xiii
Acknowledgements	xv
Become a published author	xv
Comments welcome	xvi
Stay connected to IBM Redbooks	xvi
 Chapter 1. Introduction	 1
1.1 Migration considerations	2
1.2 Informix position	4
1.3 IBM Informix editions	5
1.3.1 No-charge editions	5
1.3.2 For-purchase editions	6
1.4 Informix functionality and features	9
1.4.1 Replication and high availability	9
1.4.2 Performance	11
1.4.3 Security	12
1.4.4 Administration	13
1.4.5 Warehouse	14
1.4.6 Application development	14
1.4.7 Extensibility	15
 Chapter 2. Architecture overview	 19
2.1 Process	20
2.1.1 SQL Server	20
2.1.2 Informix	20
2.2 Memory architecture	24
2.2.1 SQL Server	24
2.2.2 Informix	25
2.3 Disk I/O architecture	29
2.3.1 SQL Server	29
2.3.2 Informix	30
2.4 Physical database structures	31
2.4.1 SQL Server	31
2.4.2 Informix	32
2.5 Logical database structures	34

2.5.1 SQL Server	34
2.5.2 Informix	35
2.6 Data dictionary and system catalog	38
2.6.1 SQL Server	38
2.6.2 Informix	38
2.7 Transactional model	39
2.7.1 SQL Server	40
2.7.2 Informix	40
2.8 Database server communications	42
2.8.1 SQL Server	42
2.8.2 Informix	43
Chapter 3. Migration methodology	47
3.1 An IBM migration methodology	48
3.2 Preparing for the migration	49
3.2.1 Performing the migration assessment	49
3.2.2 Understanding and selecting migration tools	50
3.2.3 Estimating the required effort	51
3.2.4 Preparing the environment	52
3.2.5 Receiving education about Informix	53
3.3 Migrating	53
3.3.1 Migrating and designing the database	53
3.3.2 Calibrating the planned strategy	54
3.3.3 Migrating applications	55
3.4 Testing the migration	56
3.4.1 Refreshing the migration	56
3.4.2 Migrating the data	56
3.4.3 Testing	57
3.5 Implementing the new environment	61
3.6 Related information resources	61
3.6.1 IBM training	61
3.6.2 IBM professional certification	63
3.6.3 Information Management Software Services	63
3.6.4 IBM Software Accelerated Value program	64
3.6.5 Protect your software investment with Software Subscription and Support	64
Chapter 4. SQL considerations	67
4.1 DDL	68
4.1.1 Database creation	68
4.1.2 Tables	71
4.1.3 Indexes	91
4.1.4 Views	98

4.1.5 Schemas	100
4.1.6 Sequences	102
4.1.7 Synonyms	103
4.1.8 Procedures and functions	105
4.1.9 Triggers	121
4.2 DML	130
4.2.1 SELECT statements	130
4.2.2 INSERT statements	147
4.2.3 UPDATE statements	149
4.2.4 DELETE statements	151
4.2.5 TRUNCATE statement	151
4.2.6 MERGE statement	153
4.2.7 SQL in system catalog tables	154
4.3 Transact-SQL statements	155
4.3.1 Variables	156
4.3.2 Conditional statements	157
4.3.3 GOTO statements	159
4.3.4 Statements maintaining loops	160
4.3.5 Dynamic SQL using the EXECUTE statement	162
4.3.6 Error handling and status handling	163
4.3.7 Try and catch	167
4.3.8 Debugging	168
4.3.9 More function mapping	170
4.4 SQL-based security considerations	170
4.4.1 Database user management	171
4.4.2 Database server and database permissions	171
4.4.3 Database object permissions	174
4.4.4 Roles	175
4.4.5 Column-level encryption	176
4.5 Development considerations	177
4.5.1 Concurrency and transaction	177
4.5.2 Isolation level	179
4.5.3 Locking	184
4.5.4 Cursors	187
Chapter 5. Database schema and data migration	193
5.1 Migration methodology	194
5.2 Database schema movement	197
5.2.1 Creating a database	198
5.2.2 Extracting table statements using SQL Server scripting facilities	198
5.2.3 Generating table statements with a T-SQL script	210
5.2.4 Extracting index statements using SQL Server scripting facility	213
5.2.5 Generating index statements with a T-SQL script	215

5.2.6	Implementing the views	218
5.2.7	Implementing the schema	219
5.2.8	Granting privileges and roles	221
5.2.9	Creating synonyms	222
5.2.10	Rules for migrating schema files to Informix	224
5.3	Data movement	226
5.3.1	Unloading tables with the Import/Export wizard	227
5.3.2	Generating an SQL Server Integration Services package to perform the unload task	228
5.3.3	Unloading data with SQL Server command-line tools	229
5.3.4	Changing the data representation	233
5.3.5	Loading data into the Informix database server	236
5.3.6	Data movement with distributed access	251
Chapter 6.	Application conversion	257
6.1	Heterogeneous application environments	258
6.2	Informix 11 support for native client development APIs	258
6.2.1	Embedded ESQL/C	259
6.2.2	Embedded ESQL/COBOL	259
6.2.3	IBM Informix JDBC 3.50 Driver	259
6.2.4	IBM Informix .NET Provider	260
6.2.5	IBM Informix ODBC 3.0 Driver	261
6.2.6	IBM Informix OLE DB Provider	262
6.2.7	IBM Informix Object Interface for C++	262
6.2.8	Additional APIs for accessing Informix 11	263
6.3	IBM Data Server Common Client for Informix	265
6.3.1	IBM Data Server Driver for JDBC and SQLJ	266
6.3.2	IBM Data Server Driver for ODBC and CLI	267
6.3.3	IBM Data Server Driver Package	268
6.3.4	IBM Data Server Runtime Client	269
6.3.5	IBM Data Server Client	270
6.4	Visual Studio Add-ins	271
6.4.1	Overview of Visual Studio Add-ins	271
6.4.2	Installing Visual Studio Add-ins	272
6.4.3	Configuring Visual Studio Add-ins	272
6.4.4	Server Explorer integration	275
6.4.5	Windows application development	278
6.4.6	Windows application development through DataSet creation	283
6.5	Migrating in-house applications	284
6.5.1	Application migration planning for in-house applications	285
6.5.2	Converting C/C++ applications	288
6.5.3	Converting Visual Basic applications	294
6.5.4	Converting ODBC applications	295

6.5.5	Converting Visual C++ applications	295
6.5.6	Converting .NET applications	298
6.5.7	Converting ASP applications	300
6.5.8	Converting Java applications	301
6.5.9	Converting PHP applications	308
6.6	Migrating third-party applications	317
6.6.1	Planning packaged application migration	318
6.6.2	Migrating applications based on ODBC	318
6.6.3	Migrating database applications based on JDBC	320
Chapter 7.	Informix configuration and administration.	323
7.1	Installation considerations	324
7.2	Database server setup and initialization	325
7.2.1	Configuring the database server	325
7.2.2	Set environment variables	326
7.2.3	Configure connectivity	327
7.2.4	Starting and administering the database server	330
7.2.5	Creating additional storage spaces and chunks	332
7.2.6	Configuring client connectivity	333
7.3	Informix administration utilities	335
7.3.1	OpenAdmin Tool for Informix	336
7.3.2	Command-line utilities	339
7.3.3	ON-Monitor utility	348
7.4	Database server monitoring	350
7.4.1	Memory monitoring	351
7.4.2	Process utilization and configuration	354
7.4.3	Disk space monitoring	356
7.4.4	Session monitoring	358
7.4.5	Cache monitoring	360
7.4.6	Non-Informix system monitoring tools	363
7.5	High availability and disaster recovery	363
7.5.1	Backup and restore	364
7.5.2	Fast recovery	368
7.5.3	Autonomic features	368
7.5.4	Dynamic configuration	369
7.5.5	Mirroring	369
7.6	Data replication to enable business continuity	370
7.6.1	High Availability Data Replication	371
7.6.2	Shared disk secondary server	372
7.6.3	Remote stand-alone secondary server	374
7.6.4	Enterprise Replication	376
7.6.5	The Connection Manager	378
7.7	Automatic administration and task management	379

7.7.1 The sysadmin database	380
7.7.2 Administration API.	380
7.7.3 The Scheduler.	382
7.7.4 Query drill-down	384
7.8 Informix database server security	386
7.8.1 Server utility and directory security	386
7.8.2 Network data encryption	387
7.8.3 User authentication	388
7.8.4 Single sign-on authentication	390
7.8.5 Auditing	391
7.9 Database-specific security	391
7.9.1 Database-level privileges	392
7.9.2 Table-level privileges	393
7.9.3 Other privileges	395
7.9.4 Role privileges.	395
7.9.5 Label-based access control (LBAC)	395
7.10 Informix performance enhancing features.	396
7.10.1 Compression and storage optimization.	396
7.10.2 Fragmentation	399
7.10.3 Parallelism in Informix	400
Appendix A. Terminology mapping	401
A.1 SQL Server to Informix terminology comparison	402
Appendix B. Data types	409
B.1 Supported SQL data types in C/C++	410
B.2 Supported SQL data types in Java	412
B.3 Supported SQL data types in Informix ADO.NET	414
B.4 Mapping SQL Server data types to Informix	416
Appendix C. Function mapping	419
C.1 Mathematical functions	420
C.2 Character and string functions	421
C.3 Boolean functions	424
C.4 Date and time functions	424
C.5 Metadata functions	425
C.6 Aggregate functions	427
C.7 System functions	428
C.8 Security functions	430
C.9 Miscellaneous functions	431
C.10 Implementation of new C-based functions in Informix	432
Appendix D. Operator mapping	437
D.1 Arithmetic operators	438

D.2 Variable assignment and declaration operators	438
D.3 String concatenation operators	438
D.4 Comparison operators	439
D.5 Logical operators	439
D.6 Bitwise operators	440
Appendix E. Administration and monitoring task mapping	443
E.1 Database administration	444
E.2 Database monitoring	447
Appendix F. Database server utilities	449
F.1 Informix utilities	450
Appendix G. SQL limits	453
G.1 Identifier length limits	454
G.2 Database limits	454
Related publications	457
IBM Redbooks publications	457
Other publications	457
Online resources	458
Educational support	459
How to get IBM Redbooks publications	461
Help from IBM	461
Index	463

Notices

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

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

Trademarks

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

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

AIX®	IBM®	Redbooks®
C-ISAM®	Informix®	Redbooks (logo)  ®
DataBlade®	InfoSphere™	System z9®
DataStage®	iSeries®	System z®
DB2®	Optim™	Tivoli®
developerWorks®	PowerPC®	WebSphere®
Distributed Relational Database Architecture™	POWER®	z9®
DRDA®	pSeries®	zSeries®
	Rational®	

The following terms are trademarks of other companies:

Snapshot, and the NetApp logo are trademarks or registered trademarks of NetApp, Inc. in the U.S. and other countries.

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

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

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

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

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

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

Preface

In this IBM® Redbooks® publication, we discuss considerations and describe a methodology for transitioning from Microsoft® SQL Server 2008 to IBM Informix®. We focus on the topic areas of data, applications, and administration, providing information about the differences in features and functionality, including data types, data manipulation language, data definition language, and stored procedures. Understanding the features and functionality of the two products can assist you develop a migration plan.

We provide a conversion methodology and discuss the processes for migrating the database objects and data from SQL Server to Informix using multiple methods. We show the SQL differences between SQL Server and Informix and illustrate, with examples, how to convert tables, views, stored procedures, functions, and triggers. We provide script conversion samples for data loading. We describe application programming and conversion considerations. With this information, you can gather and document your conversion requirements, develop your required transition methodology, and plan and execute the conversion activities in an orderly and cost-effective manner.

In addition, we discuss the Informix database server configuration, as well as the administration features and functions that Informix provides to help DBAs manage the Informix database server after it has been migrated.

The team who wrote this book

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

Whei-Jen Chen is a Project Leader at the International Technical Support Organization, San Jose Center. She has extensive experience in application development, database design and modeling, and DB2® system administration. Whei-Jen is an IBM Certified Solutions Expert in Database Administration and Application Development, and an IBM Certified IT Specialist.



Chee Fong Koh is a Level 2 Support Engineer with the Singapore IBM Informix support team. He joined the Informix support team in 2000 and has 10 years experience in supporting Informix products for clients in the Asia Pacific region. Chee Fong graduated from the University of Southern Queensland, Australia, with a degree in Information Technology, majoring in Applied Computer Science.



Deen Murad is an Advisory Software Engineer with Informix Advanced Support - Down System and Diagnostics group. He has been working with Informix products since 1997. He is a Certified System Administrator for IBM Informix Version 11 and participated in developing the IBM Informix Version 10 Certified System Administrator Examination. Deen has published many technical articles on a variety of Informix topics, as well as developing Informix courses and presenting at International Informix User Group conferences. He is a designated archiving and migration subject matter expert.



Holger Kirstein is a Resolution Team Engineer with the European Informix support team. He joined the Informix support team in 1996 and has over 15 years experience in application development and support for Informix database servers and Informix clients. He holds a Masters of Applied Computer Science from Technische Universität, Dresden.



Rakeshkumar Naik is an Informix Enablement Consultant with IBM America. Rakesh has worked more than 19 years in the information technology industry in various roles. His first four years were spent as a Lead 4GL/C developer with a manufacturing company where he assisted in developing a complete end-to-end 4GL-based ERP system using Informix 4.1. For the next three years, Rakesh worked in several leading roles ranging from developer, Informix DBA, and system administrator. Rakesh joined Informix, the company, in 1998, and for the next ten years, he worked on-site at several customer locations across the United States, as an Informix Advanced Support Engineer, and later, added DB2 Advanced Support to his profile. Rakesh is proficient in many programming languages, operating systems, and databases, is a member of the Informix Down System Group, and is also affiliated with the IIUG and local Informix groups.

Acknowledgements

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

Anup Nair
Cindy Fung
Dan Simchuk
Frederick Ho
Guy Bowerman
Gustavo Castro
Jacques Roy
Sally Hartnell
Ted Wasserman
IBM Software Group

Emma Jacobs
International Technical Support Organization, San Jose Center

Become a published author

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

participate either in person or as a remote resident working from your home base.

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

ibm.com/redbooks

- ▶ Send your comments in an email to:

redbooks@us.ibm.com

- ▶ Mail your comments to:

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

Stay connected to IBM Redbooks

- ▶ Find us on Facebook:

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

- ▶ Follow us on twitter:

<http://twitter.com/ibmredbooks>

- ▶ Look for us on LinkedIn:

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

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

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

- Stay current on recent IBM Redbooks publications with RSS feeds:

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

Archived

Introduction

The database management system (DBMS) is now the core of enterprise computing. It must be capable of supporting a large number of concurrent users, various types of applications, and huge volumes of data. It is, therefore, critical that the DBMS can house, organize, manage with integrity, and deliver on demand the data that is collected and stored within an organization. The DBMS also has to change with the business requirements, due to changes in client demands, and has to be enabled with a robust set of tools and applications that can accommodate the change and growth in the organization easily.

Coping with this changing environment requires speed and flexibility. Critical to meeting these changing requirements is maintaining a dynamic IT support infrastructure. IBM Informix is designed to work in such an environment to help your organization meet all these challenges.

This IBM Redbooks publication provides information that can help Microsoft SQL Server customers understand, plan for, and execute a migration to Informix in the easiest possible manner. We describe the differences between the two architectures, the tools for administration and development, the data definition language (DDL), SQL considerations, data conversion, and application conversion. With this information, you can gather and document your migration requirements, develop your required transition methodology, and plan and execute the migration activities in an orderly and cost-effective manner.

1.1 Migration considerations

Changing your DBMS platform can be a big challenge. Complexity, total cost, and the risk of downtime are key considerations when deciding whether to start such a project. However, with good planning, education, and product knowledge, you can minimize these concerns.

Migrating from SQL Server to Informix requires a certain level of knowledge of both environments. Because you are reading this book, you are likely already using, and are familiar with, SQL Server. Therefore, the purpose of this section is to introduce Informix. This section does not include an exhaustive description of Informix but, instead, includes a high-level overview of the functions and features to help you understand the structure and capabilities that are available, so that you can more easily relate them to the SQL Server environment. Subsequent sections will highlight additional features of Informix.

Many of the Informix features are unique in the industry today, enabling clients to use information in new and more efficient ways to create a business advantage. These features are designed to help businesses better use existing information assets as they move into an on-demand business environment. In this environment, the following operations for mission-critical database management applications are supported:

- ▶ Online transaction processing (OLTP) operations

OLTP applications are often used to capture new data or to update existing data. An order-entry system is a typical example of an OLTP application. OLTP applications have the following characteristics:

- Transactions that involve small amounts of data
- Indexed access to data
- Many users
- Frequent queries and updates
- Fast response times

- ▶ Batch and decision-support system (DSS) operations, including online analytical processing (OLAP)

DSS applications often report on or consolidate data that OLTP operations have captured over time. These applications provide information that is often used for accounting, strategic planning, and decision making. Typical DSS applications include payroll, inventory, and financial reports. DSS applications have the following characteristics:

- Many queries that process and return a large number of rows
- Sequential access to data
- Few users

- Small number of transactions
- Queried by front-end tools

Informix offers capabilities to support OLTP and DSS operations, to minimize database downtime, and to enable a fast and full recovery if an outage occurs.

The new suite of business availability functionality in Informix provides greater flexibility and performance in backing up and restoring an instance, automated statistical and performance metric gathering, improvements in administration, and reductions in the cost to operate the data server. Informix is flexible and can accommodate change and growth in the applications, data volumes, and number of users. It can scale in performance, as well as in functionality.

The technology that is used by Informix also enables efficient use of existing hardware and software, including single and multiprocessor architectures. It helps you keep pace with technological growth, including the requirement for improvements, such as more complex application support, which often calls for the use of nontraditional, or *rich*, data types that cannot be stored in simple character or numeric form.

Informix delivers proven technology that efficiently integrates new and complex data directly into the database. It handles time-series, spatial, geodetic, Extensible Markup Language (XML), video, image, and other user-defined data side-by-side with traditional data to meet today's most rigorous data and business demands. It also helps businesses lower total cost of ownership (TCO) by taking advantage of its general ease of use and administration, as well as its support of existing standards for development tools and systems infrastructure.

In addition to the DBMS, you also need to consider the migration of applications. Converting an application across separate platforms and separate databases is certainly not a trivial task. The decision to convert is generally made at a high level and when there is full justification in terms of costs and expected return on investment. The following issues indicate (and are the main components to build a business case for) the need to convert applications:

- Performance

Aspects of performance include scalability, availability, data movement, response time, and the ability to support multiple query workloads.

- Configuration costs

Realistic cost assessment is based on overall development, maintenance, tuning cost, and the full exploitation of current investments, both in terms of skill sets and reduced license costs.

- Data integration

Market trends highlight the importance of enterprise servers and the need to avoid data structure fragmentation to increase the value of business-critical systems. Fragmentation can cause cross-functional currency and consistency issues and can hamper innovation.

- Data infrastructure

Data is no longer an application-specific resource but an enterprise-wide tool that provides critical business information for a competitive advantage. Often, it is not enough to navigate through the data, but it is necessary to invest in the infrastructure to integrate enterprise views of data more easily.

Informix is a development-neutral environment and supports a comprehensive array of application development tools for the rapid deployment of applications under Linux®, Microsoft Windows®, and UNIX® operating environments.

Understanding the features and functionality of Informix, compared to the features and functionality of SQL Server, can assist you in developing a migration plan, which is necessary for a successful migration.

1.2 Informix position

IBM Informix Dynamic Server 11 (Informix 11) combines the robustness, high performance, availability, and scalability that are needed in businesses today.

Informix is built on the IBM Informix Dynamic Scalable Architecture (DSA) and provides one of the most effective solutions available. It includes next-generation parallel data server architecture that delivers mainframe-caliber scalability, manageability, and performance; minimal operating system overhead; automatic distribution of workload; and the capability to extend the server to handle new types of data.

Informix can adjust dynamically as requirements change from accommodating larger amounts of data, to changing query operations, to increasing numbers of concurrent users. It is designed to use efficiently all the capabilities of the existing hardware and software configuration, including single and multiprocessor architectures.

You also have the choice of installing and running Informix 11 on UNIX, Windows, Linux, and Apple. Informix 11 is supported on all major UNIX platforms, including SUN, HP, and IBM. Informix 11 is also supported on the many versions of Linux, including Red Hat, SUSE, Debian, Asianux, and Ubuntu.

With Version 11, Informix increases its lead over the data server landscape with even faster performance, a new suite of business availability functionality, greater flexibility and performance in backing up and restoring an instance, automated statistical and performance metric gathering, new autonomic and dynamic features to support changing workloads, improvements in administration, significant reductions in storage costs through data compression, cost-effective infrastructure extensions to support warehouse workloads, and many more features.

The maturity and success of Informix is built on many years of widespread use in critical business operations, which attests to its stability, performance, and usability. Informix 11 moves this already highly successful enterprise relational database server to a significantly higher level.

The addition of an extremely large number of features in the latest releases of Informix, Informix 11.10, and Informix 11.50 also demonstrates the commitment by IBM to support this product well into the future.

1.3 IBM Informix editions

IBM has packaged Informix from a price and functionality perspective into “for-purchase” and “no-charge” editions. These Informix editions are designed to help you maximize your investment and minimize your costs while helping you benefit from enterprise-class functionality. Regardless of which edition you purchase, each edition comes with the full implementation of Dynamic Scalable Architecture (DSA) and its unmatched performance, reliability, ease of use, and availability (depending on bundle-driven hardware, connection, and scalability restrictions).

Not all license terms and conditions are contained in this book. See an authorized IBM marketing representative, IBM Business Partner, or the following web page for details:

<http://www.ibm.com/developerworks/data/library/techarticle/dm-0801doe/>

1.3.1 No-charge editions

The following no-charge editions are available as separate offerings subject to the IBM International License Agreement for Non-Warranted Programs (ILAN):

- IBM Informix Innovator-C Edition:

This edition is for clients that are looking for a robust and powerful database environment that can support small production workloads. This edition provides the most widely used data processing functionality, including limited

Enterprise Replication and High Availability clustering. Available on all supported platforms, this edition is limited to one socket with no more than four cores and a total of 2 GB of RAM.

The Innovator-C no-charge Informix edition is an offering to be used for development, test, and user production workloads without a license fee. Only user organizations can use this edition. You cannot redistribute this edition without signing a redistribution contract. Support is community-based though an optional for-charge service and support package is available.

► **IBM Developer Edition:**

For application development and testing only, the Informix Developer Edition includes a full suite of functionality at no charge. Informix Developer Edition includes all the functionality that is available in Informix Ultimate Edition. It contains scalability limits for non-production use, including processing, memory, and storage limitations. It is available on a wide range of operating systems in 32-bit and 64-bit versions where appropriate. Its hardware support is limited to one physical CPU, 1 GB of RAM, and 8 GB in data storage.

Because Informix Developer Edition is offered at no charge, it comes without formal support from IBM. A number of forums exist in the Informix development community that users can join for help and support using Informix Developer Edition. You can upgrade Informix Developer Edition directly to any other edition simply by installing the new data server binaries.

1.3.2 For-purchase editions

For large companies that need all of the Informix features to handle high data loads, scalability, and 24x7x365 availability, the Informix Growth Edition and the Informix Ultimate Edition are ideal choices.

Informix Choice Edition

Informix Choice Edition is ideal for a small to medium-size business. This edition is available for both Microsoft Windows and Apple Macintosh operating systems. This edition is limited to a total of eight cores over a maximum of two sockets and 8 GB of RAM. The Informix Choice Edition includes limited Enterprise Replication (ER) clustering with 2-root nodes to send or receive data updates within the cluster. This edition also provides limited High Availability (HA) cluster functionality; along with the primary server, you can have one secondary node, either a High Availability Data Replication (HDR) secondary or Remote Standby Secondary (RSS). You can use the HA cluster secondary node for SQL operations. This edition does not support use of the Shared Disk secondary (SD secondary) node type.

Informix Growth Edition

Informix Growth Edition is for any business that needs additional power to process SQL operations, to manage extremely large databases efficiently, or to build a robust failover system to ensure database continuation in the event of a natural or human-caused outage. This edition provides unlimited Enterprise Replication (ER) functionality; however, the High Availability (HA) cluster functionality is limited to a primary server and up to two secondary node types. The Informix Growth Edition is available on all supported operating systems, including both 32-bit and 64-bit ports, and has all the components, utilities, availability features, and storage scalability. Its hardware support is limited, however, to a total of 16 cores over a maximum of four sockets and 16 GB of RAM. Further, Informix Growth Edition cannot be used to support Internet-based application connections.

To manage large databases, you need the ability to insert or extract data quickly, as well as to perform full or targeted backups. Informix Growth Edition includes functionality to do both tasks. For example, you can use the High-Performance Loader utility to execute bulk data load and unload operations. It uses the DSA threading model to process multiple concurrent input or output data streams, with or without data manipulation by other threads as the job executes. Also, it includes the ON-Bar utility suite for partial-instance or full-instance backups and restores. These backups and restores can be multithreaded, so you can send the output to multiple backup devices to reduce the amount of time that is required to create a backup or to perform a restore.

If you want to provide continuation of database services in the event of a natural or human-made outage, you can use the Informix High Availability Data Replication (HDR) option with Informix Growth Edition. With HDR, the results of data manipulation statements, such as inserts, updates, or deletes, are mirrored in real time to a hot standby server. When in standby mode, the mirror copy supports only query operations for reporting applications. Depending on the number of reporting applications, offloading the application execution to the mirror server can provide a measurable performance improvement to day-to-day operations. Furthermore, the secondary server can now also be enabled for data manipulation statements. This added functionality can provide the benefits of workload partitioning and capacity relief, in addition to high availability.

Informix Ultimate Edition

Informix Ultimate Edition includes the full feature set of the database server. In addition to the features of Informix Growth Edition, Informix Ultimate Edition includes the features that are required to provide the scalability to handle extremely high user loads and to provide 24x7x365 high availability.

You can deploy Informix Ultimate Edition in any size environment that requires the richest set of functionality that is supported by the most stable and scalable

architecture available in the market today. Informix Ultimate Edition has no processor, memory, or disk access limitations other than those limitations imposed by the operating system on which it is installed.

In addition to HDR, Informix Ultimate Edition also includes Informix Enterprise Replication (ER), an asynchronous mechanism for the distribution of data throughout the enterprise. ER uses simple SQL statements to define what database objects to replicate, to where the objects are replicated, and under what conditions replication occurs. ER preserves state information about all the servers and the data that they have received and guarantees delivery of data even if the replication target is temporarily unavailable. Data flow can be either unidirectional or bidirectional, and several conflict resolution rule sets are included to automatically handle near-simultaneous changes to the same object on separate servers.

ER is flexible and is platform-independent and version-independent. Data objects from an Informix 7 instance on Windows can be replicated to an Informix 11 instance on an AIX® or other operating system without issue. The replication topology is completely separate from the actual physical network topology and can be configured to support fully meshed, hierarchical, or forest of trees/snowflake connection paths. ER can scale easily to support hundreds of nodes, each with customized replication rules, without affecting regular transaction processing.

Informix Ultimate Edition supports concurrent operation of both HDR and ER, which gives a business the ability to protect itself from outages, as well as to migrate data automatically either for application partitioning or distribution and consolidation purposes. With the introduction of support for additional secondary server types in Informix 11, Informix Ultimate Edition has the option to provide a complete high availability, disaster recovery, workload balancing, and online capacity relief solution with minimal setup and cost.

In addition, Informix Ultimate Edition can efficiently process extremely complicated SQL operations on large database instances through the use of the Parallel Data Query (PDQ) and Memory Grant Manager (MGM) components. With these features, database server resources can be effectively divided to allow maximum resource utilization for large queries without starving and compromising non-PDQ queries. With these features, database server resources can be prereserved and then deployed fully without interruption to process any given SQL operation.

1.4 Informix functionality and features

In this section, we provide a brief overview of just a few of the Informix features and functionality. The subsequent chapters of this book provide more detailed information about both SQL Server and Informix. Understanding these environments will better equip you to make your migration project a successful one.

1.4.1 Replication and high availability

In many respects, the functionality included here is one of the key advantages of Informix. The HDR technology can create multiple layers of secondary copies of the primary server. You can add or remove these layers, depending on your network, server, or disk environments and the level of protection that is needed to support your business continuity plans.

The first of the new server types is not really a part of the HDR technology. Rather, it is an extension to the Informix ontape utility and the ON-Bar suite. We include it in our discussion here because it forms part of the availability fabric. Called a *Continuous Log Restore* (CLR) server, it supports the intermittent application of completed transactional logical log records from the primary to create a near-line copy. You can use this technology in environments where network connectivity is not constant or where the available throughput is too slow to support any other kind of replication technology.

Remote Standby Secondary (RSS) servers are full copies of the primary server but are maintained asynchronously, as opposed to the synchronous nature of communication between the primary and the HDR secondary, regardless of its replication mode. However, this feature is not just 1 to *N* HDR secondary. You can have as many RSS instances as you need. In addition, although an RSS instance cannot be promoted to become an HDR primary, it can become an HDR secondary, after which it can be promoted to the primary if needed.

Consider RSS instances as disaster recovery instances, not high availability (HA) instances. RSS instances are deployed to expand the real-time failover capability of an HDR environment. Another use is to provide promotable redundancy. In the event of a primary failure, the RSS instance can be promoted to the HDR secondary to protect the new primary instance. A third benefit to an RSS instance is where the one and only failover server must be located geographically distant from the primary and the network latency or throughput is too great to support normal HDR replication.

The last of the expanded HDR server types is called the *Shared Disk Secondary* (SDS) server. SDS instances can also participate in high-availability cluster

configurations. In such configurations, the primary server and the SDS server share the same disk or disk array. SDS instances do not maintain a copy of the physical database on their own disk space. Rather, an SDS instance shares disks with the primary server. SDS instances can help provide redundancy and failover options, because they can be anywhere in the network. However, they do not protect against disk-related failures.

SDS instances are beneficial when reporting capacity is to be increased, because they can off-load reporting functionality without affecting the primary server. Also, in the event of a failure of the primary server, an SDS instance can be promoted quickly and easily to a primary server. For example, if you use storage area network (SAN) devices that provide ample and reliable disk storage but you are concerned with server failure, SDS instance can provide a reliable backup.

In addition to providing availability and disaster recovery, all the instances can also participate in ER, further expanding your options for building a failure-resistant environment.

Beginning with Informix 11.5, you can enable applications that are connected to secondary servers to update data. You can configure secondary servers so that client applications can send transactions that update data. If you enable write operations on a secondary server, insert, update, and delete operations are propagated to the primary server. The ability to update secondary servers is not enabled by default. You must set an Informix configuration parameter to allow transactions on each secondary instance.

Informix 11.5 also includes the Connection Manager, which is a stand-alone program that routes client application connection requests dynamically to the most appropriate server in a high-availability cluster. Connection Manager connects to each of the servers in the cluster and gathers statistics regarding the type of server, unused workload capacity, and the current state of the server. Using this information, the Connection Manager redirects the connection to the appropriate server. This process is invisible to the application.

In addition, Connection Manager Arbitrator provides automatic failover logic for high-availability clusters. You configure Connection Manager Arbitrator using a configuration file to specify which secondary server will take over the role of the primary in the event of a failure of the primary server. The benefit of the Connection Manager and failover arbitrator is that it can automate the node failover while ensuring that it is a real failure. It also supports application failover.

You can configure the Connection Manager as a proxy server when clients connect to Informix database servers from outside a firewall. It can also work with other connection managers to insure that there is not a single point of failure. The

Connection Manager is a *free* product that is packaged with the Informix Client Development Software Kit (CSDK).

1.4.2 Performance

Informix provides many built-in performance features, including:

- ▶ The new data compression technology in Informix 11 can produce up to an 80% savings on disk space and I/O improvements of up to 20%. With less volume to move, you can achieve faster search times, more efficient use of memory, and reduced backup and recovery time. Informix provides full online support for turning on storage optimization and for compressing existing table data while applications continue to use the table. Thus, no system downtime is required to utilize the Informix storage optimization technology. Informix compression technology works by considering the entire row and all its columns as a single string of bytes. Informix identifies repeating patterns and stores those patterns as symbols in the dictionary.
- ▶ The High-Performance Loader utility can load data quickly, because it can read from multiple data sources (such as tapes, disk files, pipes, or other tables) and can load the data in parallel. You can configure a High-Performance Loader job so that normal load tasks, such as referential integrity checking, logging, and index builds, are performed either during the load or afterwards, which speeds up the load time. You can also use the High-Performance Loader to extract data from one or more tables for output to one or more target locations.
- ▶ Parallel Data Query (PDQ) takes advantage of the CPU power that is provided by symmetric multiprocessor (SMP) systems and an Informix virtual processor to execute fan-out parallelism. PDQ is of greatest benefit to more complex analytical SQL operations. The operation is divided into a number of subtasks, which are given higher or lower priority for execution within the data server's resources based on the overall PDQ priority level that is requested by the operation.
- ▶ The parallel scan feature takes advantage of table partitioning in two ways:
 - If the SQL optimizer determines that each partition must be accessed, a scan thread for each partition executes in parallel with the other threads to bring the requested data out as quickly as possible.
 - If the access plan only calls for 1 to $N-1$ of the partitions to be accessed, another access operation can execute on the remaining partitions so that two (or more) operations can be active on the table or index at the same time, which can provide a significant performance boost.
- ▶ The Informix cost-based optimizer determines the fastest way to retrieve data based on detailed statistical information about the data within the database

generated by the UPDATE STATISTICS SQL command. The optimizer uses this information to pick the access plan that provides the quickest access to the data while trying to minimize the effect on system resources. Using the new Auto Update Statistics (AUS) feature, you can improve the performance of database SQL by allowing Informix to generate table statistics automatically based on specified policies.

- ▶ Interval checkpoints are operations that are conducted by the data server to ensure the logical consistency of the data. However, with interval checkpoints, service interruptions are virtually eliminated. Memory writes occur in the background, allowing user transaction processing to continue. The net result is a significant improvement in the number of transactions that can be processed over time.
- ▶ SQL optimizer has the ability to rewrite SQL operations when it recognizes query plans that require entire index scans, enabling query results to be returned more quickly.

1.4.3 Security

Informix has many security features to help businesses meet regulatory requirements. Security within the database server can be implemented at various levels, including instance connection, database, table, table fragment, column, view, and routine. Privileges can be granted individually or grouped into roles and then granted.

The addition of Label-Based Access Control (LBAC) permits you to design and implement multilevel data access and control labels and policies. These labels and policies are enforced regardless of the method that is used to access the data. Policies can be as simple (public and private) or as complicated as needed for your environment.

Another capability enables the instance to invoke automatically a specific user-defined routine (UDR) whenever a session connects or disconnects from the instance. Thus, you can now write a series of UDRs that are executed when a user session connects and disconnects from the instance. These routines can perform any functionality, including setting roles, specifying session operating parameters, setting the isolation level or the output location of optimizer reports, turning on (or off) monitoring functionality, and sending an alert.

Informix also has the ability to use filters in backup and restore operations. For example, it provides the ability to use a filter in-line with the backup or restore operation. This filter can re-encrypt the data before it leaves the instance for the storage medium or can perform other operations, such as compression, if the hardware device does not support that functionality.

You can configure Informix to use the Secure Sockets Layer (SSL) protocol, which encrypts data in TCP/IP connections between two points over a network. The SSL protocol is an alternative to the Informix-specific encryption Communication Support Module (CSM) and simple password CSM for CSDK clients. You must use SSL to encrypt data in communications between Informix and Distributed Relational Database Architecture (DRDA®) clients.

Informix delivers support for single sign-on (SSO) in the Generic Security Services Communications Support Module (GSSCSM) and uses the Kerberos 5 security protocol. With SSO, users provide a valid user ID and password when they log in to a client computer, and they can access the database server and other SSO-enabled services without having to log in again. In the past, users had to log in multiple times. After you enable SSO, you benefit from centralized management of authentication.

1.4.4 Administration

Informix has a graphical DBA tool called *OpenAdmin Tool for Informix*. Designed to help DBAs answer the most commonly asked questions, it is written in Hypertext Preprocessor (PHP) and works in conjunction with an Apache (or similar) Web server to manage all instances in your environment without having to load anything on the target servers. It has tasks and sensors to gather information and execute operations.

Allowing a DBA more capability to maintain instances remotely is one of the administration goals. With the OpenAdmin Tool, a single connection to an instance can now permit a DBA to monitor and maintain multiple instances. The instances can be local, on the same machine as the DBA, or in remote locations.

Informix delivers what is referred to as an *administration free zone*. Database administration is integrated with the SQL application programming interface (API) to enable the DBA to have more control over the instance and to automate many tasks that typically require intervention. Many tasks can be scripted using SQL and stored procedures, making those tasks platform independent and allowing a DBA to consolidate many of the tasks into a single environment or to monitor many environments.

The new Scheduler allows you to create, manage, and run scheduled maintenance, monitoring, and administration tasks at predefined times or as

determined internally by the server. You can monitor activities (for example, space management) and create automatic corrective actions. Scheduler functions collect information as well as monitor and adjust the server, using an SQL-based administrative system and a set of tasks. A set of task properties, which define what needs to be collected or executed, control the Scheduler.

The Memory Grant Manager (MGM) works in conjunction with PDQ to control the degree of parallelism by balancing the priority of OLAP-oriented user requests with available system resources, such as memory, virtual processor capacity, and disk scan threads. The Informix administrator can set query-type priorities, adjust the number of queries allowed to run concurrently, and adjust the maximum amount of memory used for PDQ-type queries.

You can use the deployment utility *ifxdeploy* to rapidly deploy a configured Informix instance to multiple computers, avoiding the need to configure manually instances on each computer and wait for each instance to initialise. By setting configuration parameters, essential environment variables, and basic connectivity information in a deployment utility configuration template file, you can reuse the configuration file to deploy the instance multiple times. You can call the utility programmatically or from a script as part of an application installation that embeds Informix.

1.4.5 Warehouse

The new warehouse capabilities for Informix provide an integrated and simplified software platform. It is now easier to use your existing Informix infrastructure to power business intelligence solutions with tools that simplify warehouse design and deployment. You can use a single database for both transactions and analytics or build a separate data warehouse, depending on workload requirements. The new tools make it easy to transform your data to make smarter business decisions and to do more with lower cost hardware.

1.4.6 Application development

Distributed Relational Database Architecture™ (DRDA) is a database interoperability standard from The Open Group. It is a set of protocols that enable communication between applications and database systems on disparate platforms and enables relational data to be distributed among multiple platforms. You can configure Informix to use DRDA to respond to requests from DRDA common API. Prior to Informix 11, Informix responded only to SQLI connection requests.

XSL uses XML elements to describe the format of a document. You can use the built-in XSLT functions that Informix provides to apply XSL transformations

(XSLT) to XML documents, resulting in a document in a different XML schema, HTML, PDF, or any defined type. XSL and XSLT are standards defined by the World Wide Web Consortium, which you can find at:

<http://www.w3.org/TR/xslt>

XML publishing provides a way to transform results of SQL queries into XML structures. You can use the built-in XML publishing functions that Informix provides to transform the result set of an SQL query into an XML structure, optionally including an XML schema and header. You can store the XML in Informix for use in XML-based applications.

1.4.7 Extensibility

Informix provides a complete set of extensibility features, including support for new data types, routines, aggregates, and access methods. This object-relational extensibility supports transactional consistency and data integrity while simplifying database optimization and administration. Much of the functionality is delivered with IBM Informix DataBlade®.

IBM Informix DataBlade bring additional business functionality to the data server through specialized user-defined data types, routines, and access methods. Developers can use these new data types and routines to more easily create and deploy richer applications that better address a company's business needs. Informix provides the same level of support to DataBlade functionality that is accorded to built-in or other user-defined types and routines. With IBM Informix DataBlade, you can manage almost any kind of information as a data type within the data server.

Several DataBlade modules are bundled as part of the data server, enabling application developers to enrich applications quickly and easily:

- Binary DataBlade

The Binary DataBlade provides the ability to use two new indexable extensible data types: *binary18* and *binaryvar*. The *binary18* data type is a fixed-length data type that holds 18 bytes. Because this data type is fixed in length, unused space is right-padded with zeros until the column length reaches 18. The *binaryvar* data type is a variable-length type that can hold up to 255 bytes of information.

- Basic Text Search DataBlade

This DataBlade expands the text string matching capabilities of the data server through the creation of a specialized index that supports proximity (for example, are two words within *N* words of each other) and fuzzy text searches. More robust text search functionality is available through the IBM Informix Excalibur Text DataBlade.

There is a growing portfolio of third-party DataBlade modules, and developers can use the IBM Informix DataBlade Developer's Kit (DBDK) to create specialized blades for a particular business need.

The following IBM Informix DataBlade technologies are available:

► IBM Informix TimeSeries DataBlade

This DataBlade provides a better way to organize and manipulate any form of real-time, time-stamped data. Use this DataBlade for applications that use large amounts of time-stamped data, such as network analysis, manufacturing throughput monitoring, or financial tick data analysis.

► IBM Informix TimeSeries Real-Time Loader

A companion component to the IBM Informix TimeSeries DataBlade, the TimeSeries Real-Time Loader is designed to load time-stamped data and make it available to queries in real time.

► IBM Informix Spatial DataBlade and the IBM Informix Geodetic DataBlade

These DataBlades provide functionality to manage complex geospatial information intelligently, within the efficiency of a relational database model. The IBM Informix Geodetic DataBlade stores and manipulates objects from a *whole-earth* perspective using four dimensions: latitude, longitude, altitude, and time. The IBM Informix Spatial DataBlade is a set of routines that are compliant with open geographic information system (GIS) standards, which take a *flat-earth* perspective to mapping geospatial data points. This DataBlade is based on routines, utilities, and Economic and Social Research Institute (ESRI) technology.

► IBM Informix Excalibur Text DataBlade

This DataBlade performs full text searches of documents stored in database tables and supports any language, word, or phrase that can be expressed in an 8-bit, single-byte character set.

► IBM Informix Video Foundation DataBlade

This DataBlade allows strategic third-party development partners to incorporate specific video technologies, such as video servers, external control devices, codecs, or cataloging tools, into database management applications.

► IBM Informix Image Foundation DataBlade

This DataBlade provides functionality for the storage, retrieval, transformation, and format conversion of image-based data and metadata.

► IBM Informix C-ISAM® DataBlade

This DataBlade provides functionality to the storage and use of Indexed Sequential Access Method (ISAM)-based data.

The current list of available IBM Informix DataBlade technologies is available at this Web site:

<http://www.ibm.com/informix>

Archived

Architecture overview

In this chapter, we present architecture overviews of the Microsoft SQL Server (SQL Server) and Informix relational database management system (RDBMS) products. The information that we include in this chapter can facilitate your understanding of both architectures, while also taking into consideration that you might already be familiar with either SQL Server or Informix. Understanding the differences between the two architectures can help in the transition from SQL Server to Informix.

We discuss the following topics in this chapter:

- ▶ Process
- ▶ Memory architecture
- ▶ Disk I/O architecture
- ▶ Physical database structures
- ▶ Logical database structures
- ▶ Data dictionary and system catalog
- ▶ Transactional model
- ▶ Database server communications

In general, the Informix architecture is same on both the UNIX and Windows platforms. We describe the architecture based on the UNIX platform and point out the Windows specifics.

2.1 Process

In this section, we describe the differences between process implementation in SQL Server and Informix. Both SQL Server and Informix (Windows version) use a single process. Informix uses a multiprocess and multithreaded model on a multiprocessing system (such as Linux, UNIX, or Mac OS X).

2.1.1 SQL Server

SQL Server runs as a single process called `sqlservr.exe`, which represents a single instance. Within the SQL Server process, there are various threads running, such as the read-ahead manager, which retrieves data from disk to memory, and the backup process. There are also threads running for the transaction log writer and for user commands. The threads are managed by Open Data Services and have their CPU time scheduled by the User Mode Scheduler.

SQL Server also has a process for the SQL Agent (`sqlagent.exe`), Distributed Transaction Coordinator (`msdtc.exe`), and Active Directory Helper (`sqladhlp.exe`).

2.1.2 Informix

An Informix process is called a *virtual processor* (VP), because the way that it functions is similar to the way that a CPU functions in a computer. Just as a CPU runs multiple operating system processes to service multiple users, a database server virtual processor runs multiple threads to service multiple SQL client applications.

A virtual processor is a process that the operating system schedules for processing. Informix virtual processors are multithreaded because they run multiple concurrent threads.

Figure 2-1 on page 21 illustrates the relationship of client applications to virtual processors. A small number of virtual processors serve a much larger number of client applications or queries.

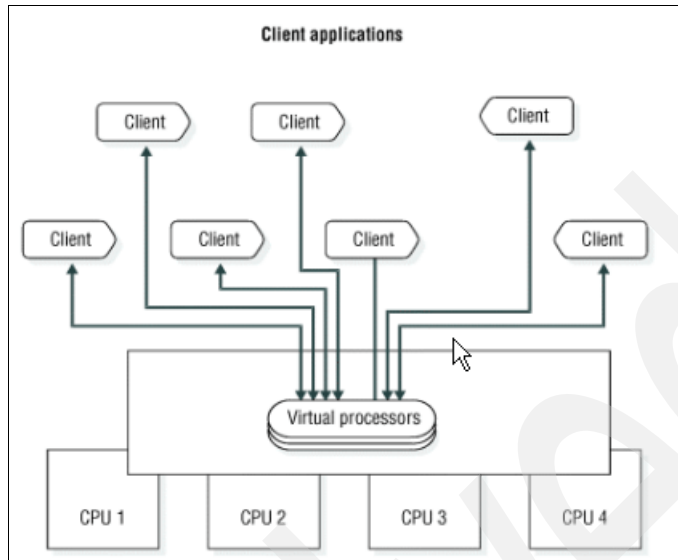


Figure 2-1 Virtual processors

Compared to a database server process that services a single client application, the dynamic, multithreaded nature of a database server virtual processor provides the following advantages:

- ▶ Virtual processors can share processing.
- ▶ Virtual processors save memory and resources.
- ▶ Virtual processors can perform parallel processing.
- ▶ You can start additional virtual processors and terminate active CPU virtual processors while the database server is running.
- ▶ You can bind virtual processors to CPUs.

A virtual processor runs threads on behalf of SQL client applications (session threads) and also to satisfy internal requirements (internal threads). In most cases, for each connection by a client application, the database server runs one session thread. The database server runs internal threads to accomplish, among other things, database I/O, logging I/O, page cleaning, and administration tasks.

A *user thread* is a database server thread that services requests from client applications. User threads include session threads, called *sqlexec threads*, which are the primary threads that the database server runs to service client applications. Figure 2-2 on page 22 illustrates virtual processors in an Informix instance.

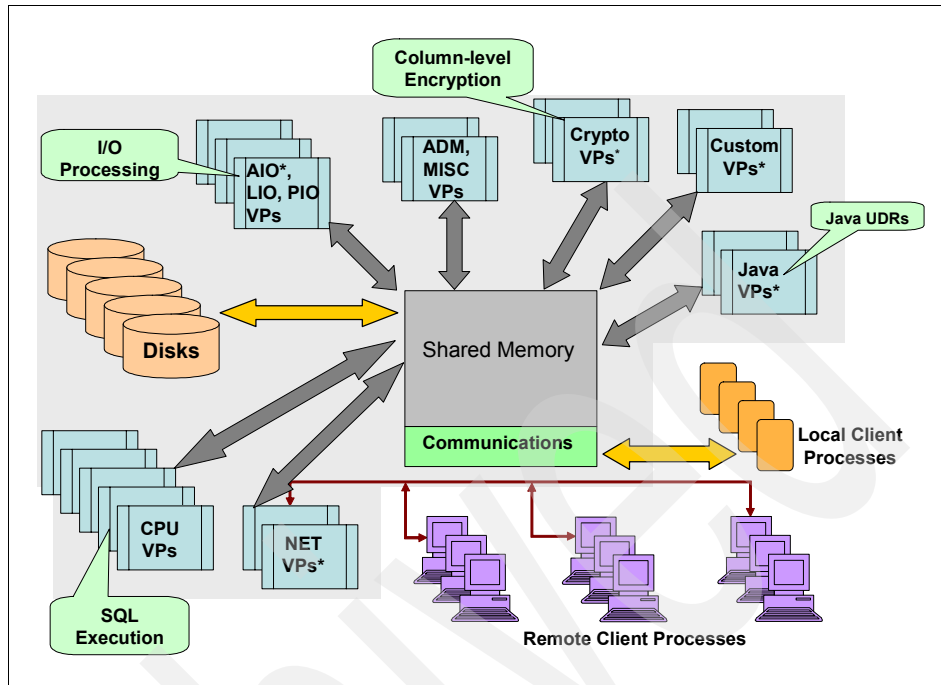


Figure 2-2 Informix virtual processors

Types of virtual processors in Informix

A virtual processor is a process that the operating system schedules for executing tasks. Informix has multiple classes of virtual processors, and each class is dedicated to processing certain types of threads. The number of virtual processors of each class that you configure depends on the availability of physical processors (CPUs), hardware memory, and the database applications in use.

On Windows, an Informix instance has a single `oninit` process, and each virtual processor is an operating system thread within that process. The Informix virtual processors can be viewed as operating system threads by cross-referencing the "pid" values with the thread IDs shown by a utility, such as Process Explorer.

Table 2-1 on page 23 lists the types of virtual processors in Informix.

Table 2-1 Informix virtual processors

Name	Description
CPU VP	Runs all session threads and certain system threads. Runs thread for kernel asynchronous I/O (KAIO) where available. Can run a single poll thread, depending on the configuration.
PIO VP	Writes to the physical log file (internal class) if it is in cooked disk space.
LIO VP	Writes to the logical log files (internal class) if they are in a cooked disk space.
AIO VP	Performs non-logging disk I/O. If KAIO is used, AIO virtual processors perform I/O to cooked disk spaces.
SHM VP	Performs shared memory communication.
TLI VP	Uses the transport layer interface (TLI) to perform network communication.
SOC VP	Uses sockets to perform network communication.
OPT VP	Performs I/O to optical disk in UNIX.
ADM VP	Performs administration functions.
ADT VP	Performs auditing functions.
MSC VP	Serves requests for system calls that require a large stack.
CSM VP	Communications support module performs communications support service operations.
Encrypt VP	Used by the database server when encryption or decryption functions are called.
Java™ VP (JVP)	Contains the Java Virtual Machine (JVM), and executes Java user-defined routines (UDRs).
User-defined VP	Runs user-defined routines in a thread-safe manner so that if the routine fails, the database server is unaffected. Specified with the VPCLASS configuration parameter.

For more information about the Informix virtual processors, see *Informix Dynamic Server Administrator's Guide*, G229-6359.

2.2 Memory architecture

Shared memory is an operating system feature that allows the database server threads and processes to share data by sharing access to pools of memory. The database server uses shared memory for the following purposes:

- ▶ Reducing memory usage and disk I/O
- ▶ Performing high-speed communication between processes

Shared memory enables the database server to reduce overall memory usage because the participating processes do not need to maintain private copies of the data that is in shared memory.

Shared memory reduces disk I/O, because buffers, which are managed as a common pool, are flushed on a database server-wide basis instead of a per-process basis. Furthermore, a virtual processor can often avoid reading data from disk because the data is already in shared memory as a result of an earlier read operation. The reduction in disk I/O reduces execution time.

In this section, we give an overview about how memory is allocated and used in an SQL Server and an Informix server.

2.2.1 SQL Server

Each instance of SQL Server has its own memory address space, which is divided into two areas:

- ▶ Executable code
- ▶ Memory pool

Dynamic link libraries (DLLs) and executable code used by the SQL Server Engine and Net-Libraries are loaded into the executable memory area. The memory pool is the main area of memory for SQL Server, and almost all objects are loaded into the memory pool. The memory pool includes the following objects:

- ▶ System data structure
- ▶ Buffer cache
- ▶ Procedure cache
- ▶ Log cache
- ▶ Connection context

The memory pool size and division change continuously, because the regions within the memory pool are adjusted constantly to optimize performance. In addition to SQL Server adjusting the memory allocation, it is possible for an administrator to manually allocate memory to the instance. SQL Server is also

able to utilize memory above the 2 GB to 4 GB limit through the use of the Address Windowing Extensions (AWE) application programming interface (API).

2.2.2 Informix

Informix uses the following shared memory portions:

- ▶ Resident portion
- ▶ Virtual portion
- ▶ Message portion (for Inter Process Communication on UNIX)
- ▶ Virtual Extension portion

Figure 2-3 illustrates the shared memory segments in Informix.

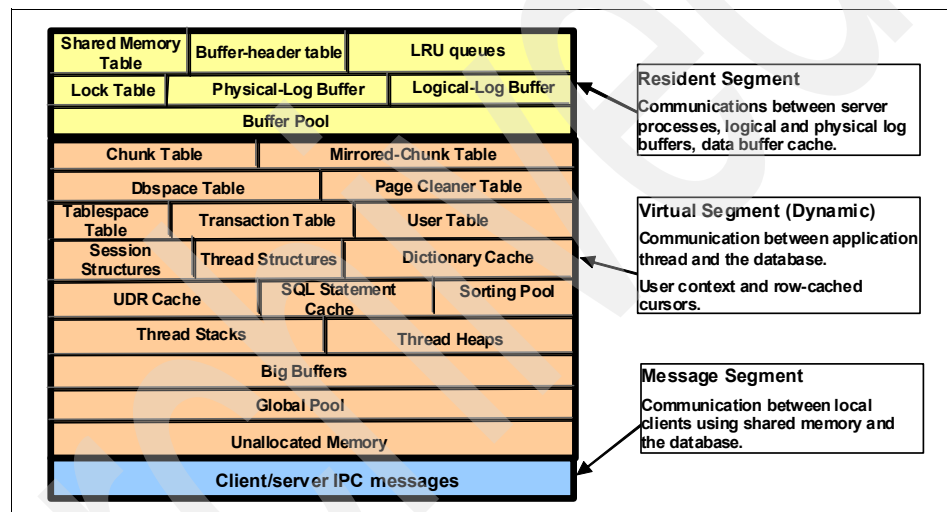


Figure 2-3 Shared memory segments in Informix

Resident portion of the shared memory

The resident portion of the Informix shared memory stores the following data structures that do not change in size while the database server is running:

- ▶ Shared memory header

The shared memory header contains a description of all other structures in shared memory, including internal tables and the buffer pool.

The shared memory header also contains pointers to the locations of these structures. When a virtual processor first attaches to shared memory, it reads address information in the shared memory header for directions to all other structures.

► Buffer pool

The buffer pool in the resident portion of shared memory contains buffers that store dbspace pages read from disk. The pool of buffers comprises the largest allocation of the resident portion of shared memory.

In Informix, the buffer pool is created at the instance level, and one instance can have one or many user databases. You use the BUFFERPOOL configuration parameter in the Informix instance ONCONFIG parameter file to specify information about a buffer pool, including the number of buffers in the buffer pool.

If you are creating a dbspace with a non-default page size, the dbspace must have a corresponding buffer pool. For example, if you create a dbspace with a page size of 8 KB, you must create a buffer pool with a page size of 8 KB.

If a buffer pool for a non-default page size does not exist, the database server will automatically create a large-page buffer.

► Logical log buffer

The database server uses the logical log to store a record of changes to the database server data since the last dbspace backup. The logical log stores records that represent logical units of work for the database server. The logical log contains the following types of log records, in addition to many others:

- SQL data definition statements for all databases
- SQL data manipulation statements for databases created with logging
- Record of a change to the logging status of a database
- Record of a checkpoint
- Record of a change to the configuration

The database server uses only one of the logical log buffers at a time. This buffer is the current logical log buffer. Before the database server flushes the current logical log buffer to disk, it makes the second logical log buffer the current one so that it can continue writing while the first buffer is flushed. If the second logical log buffer fills before the first one finishes flushing, the third logical log buffer becomes the current one.

The LOGBUFF configuration parameter in the ONCONFIG file specifies the size of the logical log buffers. There is one set of logical log buffers for all the databases in the Informix instance. The recommended value for the size of a logical log buffer is 64 KB.

► Physical log buffer

The physical log buffer holds before-images of the modified pages in the buffer pool. The before-images in the physical log and the logical log records enable the database server to restore consistency to its databases after a system failure.

The physical log buffer is actually two buffers. Double buffering permits the database server processes to write to the active physical log buffer while the other buffer is being flushed to the physical log on disk.

The PHYSBUFF parameter in the ONCONFIG file specifies the size of the physical log buffers. The default value for the physical log buffer size is 512 KB.

► Lock table

Locks can prevent sessions from reading data until after a concurrent transaction is committed or rolled back. A lock is created when a user thread writes an entry in the lock table. The lock table is the pool of available locks, and a single transaction can own multiple locks.

The lock table stores the following information:

- The address of the transaction that owns the lock
- The type of lock (exclusive, update, shared, byte, or intent)
- The page or ROWID that is locked
- The table space where the lock is placed
- Information about bytes locked (byte-range locks for smart large objects)

The LOCKS configuration parameter specifies the initial size of the LOCK table. If the number of locks allocated by sessions exceeds the value specified in the LOCKS configuration parameter, the database server doubles the size of the lock table, up to 15 times.

Virtual portion of the shared memory

The virtual portion of shared memory is expandable by the database server and can be paged out to disk by the operating system. As the database server executes, it attaches additional operating system segments automatically, as needed, to the virtual portion. The database server uses memory pools to track memory allocations that are similar in type and size.

The database server allocates virtual shared memory for each of its subsystems (session pools, stacks, heaps, control blocks, system catalog, Stored Procedure Language (SPL) routine caches, SQL statement cache, sort pools, and message buffers) from pools that track free space through a linked list. When the database server allocates a portion of memory, it first searches the pool free-list for a fragment of sufficient size. If it finds none, it brings new blocks into the pool from the virtual portion. When memory is freed, it goes back to the pool as a free fragment and remains there until the pool is destroyed. When the database server starts a session for a client application, for example, it allocates memory for the session pool. When the session terminates, the database server returns the allocated memory as free fragments.

The SHMVIRTSIZE parameter in the ONCONFIG file specifies the initial size of the virtual shared memory portion. SHMADD or EXTSHMADD configuration parameters specify the size of segments that are added later to the virtual shared memory.

The virtual portion of shared memory stores the following data:

- ▶ Internal tables
- ▶ Big buffers
- ▶ Session data
- ▶ Thread data (stacks and heaps)
- ▶ Data-distribution cache
- ▶ Dictionary cache
- ▶ SPL routine cache
- ▶ SQL statement cache
- ▶ Sorting pool
- ▶ Global pool

Message portion of the shared memory

The database server allocates memory for the Inter Process Communication (IPC) portion of shared memory, if you configure at least one of your connections as an IPC shared memory connection. The database server performs this allocation when you set up shared memory.

The communications portion contains the message buffers for local client applications that use shared memory to communicate with the database server. The size of the communications portion of shared memory equals approximately 12 KB multiplied by the expected number of connections that are needed for shared memory communications.

In a Windows environment, Informix uses shared memory segments for its internal memory. The Windows operating system implements shared memory as memory-mapped files using a structure called a *section object*. So, when you look at the details of a process, you will find that the Informix segments are called *sections*.

Figure 2-4 on page 29 shows that the shared memory segment output from the **onstat -g seg** command is contrasted with an operating system view of the same shared memory segments shown by the Windows Sysinternals Process Explorer utility. When the section object is mapped, the name of each file mapping object corresponds to the unique shared memory key (which is created from the Informix instance number SERVERNUM).

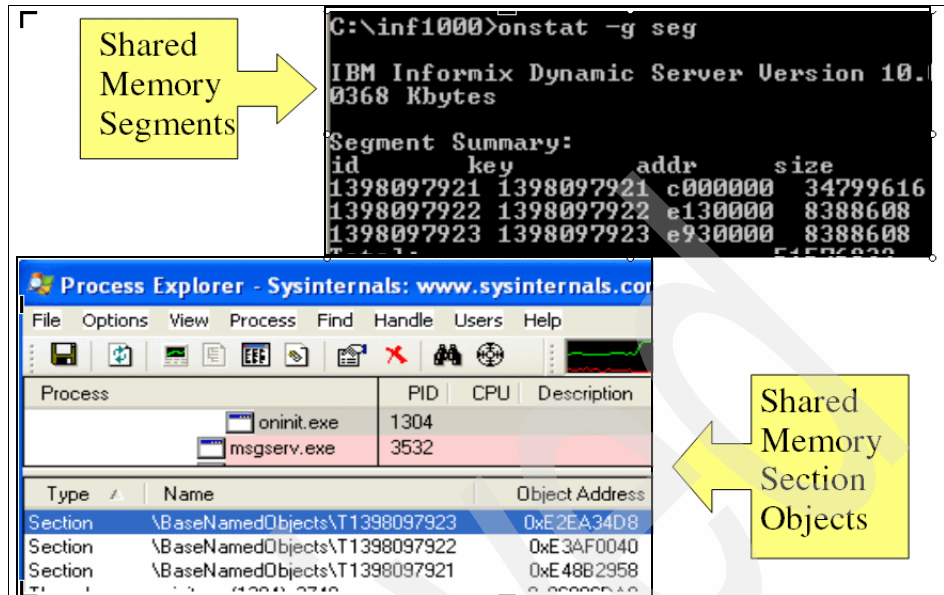


Figure 2-4 Shared memory segment

2.3 Disk I/O architecture

In this section, we explain how SQL Server and Informix differ in terms of disk I/O architecture.

2.3.1 SQL Server

SQL Server stores data into files and filegroups and relies heavily upon the operating system for all the file storage, access, and maintenance needs. Using a file system that limits the scalability and concurrency access can cause a considerable performance overhead.

There are several common issues when using a file system to store data:

- ▶ File system overhead: Extra overhead is needed for file creation and maintenance.
- ▶ Double caching: Duplicate efforts in caching and buffering.
- ▶ Slow write speeds: File system I/O performance is normally slower.
- ▶ Security: Database data files can be easily visible.

2.3.2 Informix

Informix allows you to use either raw devices or cooked files for the database. A *raw device* is a special block device file that allows direct access to a storage device, such as a hard drive, and bypasses the operating system's caches and buffers. Apart from that, Informix on most platforms (except Windows) is multithreaded and uses multiple virtual processors for disk I/O.

In this section, we describe in more detail the Informix virtual processor for disk I/O.

AIOVP

AIOVP stands for *asynchronous I/O virtual processors*. On a platform that does not support kernel asynchronous I/O (KAIO) or that does not use raw devices, Informix performs database I/O through the AIOVP class of virtual processors. The AIO virtual processors service all I/O requests equally within their class.

The database server assigns each disk chunk a queue, sometimes known as a *general file descriptor* (gfd) queue, based on the file name of the chunk. The database server orders I/O requests within a queue according to an algorithm that minimizes disk head movement. The AIO virtual processors service queues that have work pending in round-robin fashion.

All other non-chunk I/O is queued in the AIO queue.

You use the VPCLASS parameter with the `aio` keyword to specify the number of AIO virtual processors that the database server starts initially. For information about VPCLASS, refer to the chapter on configuration parameters in the *IBM Informix Dynamic Server Administrator's Reference*, SC23-7749.

You can start additional AIO virtual processors while the database server is in online mode. For more information, refer to the *IBM Informix Dynamic Server Administrator's Guide*, SC23-7748.

You cannot drop AIO virtual processors while the database server is in online mode.

KAIO

Informix implements KAIO by running a KAIO thread on the CPU virtual processor. The KAIO thread performs I/O by making system calls to the operating system, which performs the I/O independently of the virtual processor. The KAIO thread can produce better performance for disk I/O than the AIO virtual processor can, because it does not require a switch between the CPU and AIO virtual processors.

On Windows, chunk I/O always uses KAIO; therefore, the `VPCLASS aio` parameter setting defaults to 1 if not present. Usually, one AIO VP is all that is required on Windows to handle writing to log files, except on systems where a large amount of non-chunk I/O takes place. In that case, a small increase can improve performance.

Direct I/O

You can now improve the performance of cooked files by using direct I/O. In general, cooked files are slower because of the additional overhead and buffering provided by the file system. Direct I/O bypasses the use of the file system buffers and, therefore, is more efficient for reads and writes that go to disk. You specify direct I/O with the new `DIRECT_IO` configuration parameter. If your file system supports direct I/O for the page size used for the `dbspace` chunk and if you use direct I/O, performance for cooked files can approach the performance of raw devices used for `dbspace` chunks.

On Windows, the Informix I/O subsystem implementation is implemented using overlapped or asynchronous operations with no operating system buffering. Therefore, `DIRECT I/O` is not required on Windows.

2.4 Physical database structures

In this section, we discuss the physical database structures of SQL Server and Informix.

2.4.1 SQL Server

The physical database structures of SQL Server are organized into files and filegroups. A database consists of one primary data file but can have secondary data files, with the log files having their own database files. Database files can grow automatically at a specified increment until they reach a defined limit or until the disk runs out of available space. The data files can then be grouped in filegroups. SQL Server uses filegroups to simplify data organization, enabling an administrator to group tables that hold certain types of data.

SQL Server has a set of six error logs per instance. These error logs are typically found in the `Microsoft SQL Server\MSSQL\LOG` directory or the `\MSSQL$instanceName` directory if more than one instance is installed. Note that only one log is active at any one time. The error logs can be gathered with the `sqldiag` utility. Information is also recorded in the Windows application event log.

You can view the SQL Server error log using SQL Server Management Studio or a text editor. The error log is normally located at the Program Files\Microsoft SQL Server\MSSQL.*n*\MSSQL\LOG\ERRORLOG and ERRORLOG.*n* files.

2.4.2 Informix

The physical structures of Informix include chunks, the ONCONFIG configuration file, logical log files, a physical log file, and a message file, which we describe in this section.

Chunks

A *chunk* in Informix is the same as a data file in SQL Server. A chunk is the largest unit of physical disk that is dedicated to database server data storage. The maximum size of an individual chunk is four terabytes. A chunk can be a regular operating system file (cooked file) or a raw disk device.

Configuration file (ONCONFIG file)

Each Informix configuration file consists of configuration parameters that are used to initialize the Informix instance. The Informix instance configuration file is also known as the ONCONFIG file.

You can set the ONCONFIG environment variable to specify the name and location of the Informix instance configuration file, as shown in the following command:

```
export ONCONFIG=$INFORMIXDIR/etc/ONCONFIG.prod
```

If you start the database server with the **oninit** command and do not explicitly set the ONCONFIG environment variable, the database server looks for configuration values in the \$INFORMIXDIR/etc/onconfig.std file.

Logical log files

To keep a history of transactions and database server changes since the time of the last storage space backup, the database server generates log records. The database server stores the log records in the logical log, a circular file that is composed of three or more logical log files. The log is called *logical*, because the log records represent logical operations of the database server, as opposed to physical operations. At any time, the combination of a storage space backup plus logical log backup contains a complete copy of your database server data.

When you initialize or restart the database server, it creates the number of logical log files that you specify in the LOGFILES configuration parameter. The size of the logical log files is specified with LOGSIZE parameter in the ONCONFIG parameter file.

When the database server initializes disk space, it places the logical log files in the default root dbspace. You have no control over this action. To improve performance (specifically, to reduce the number of writes to the root dbspace and to minimize contention), move the logical log files out of the root dbspace to a dbspace on a disk that is not shared by active tables or the physical log.

To improve performance further, separate the logical log files into two groups and store them on two separate disks (neither of which contains data). For example, if you have six logical log files, you might locate files 1, 3, and 5 on disk 1 and files 2, 4, and 6 on disk 2. This arrangement improves performance, because the same disk drive never has to handle writes to the current logical log file and backups to tape at the same time.

The logical log files contain critical information and must be mirrored for maximum data protection. If you move logical log files to a separate dbspace, plan to start mirroring on that dbspace.

Physical log file

The physical log is used to record “before” images (first copy) of pages that have been modified in shared memory.

When the database server initializes disk space, it places the physical log in the default root dbspace. To improve performance, you can move the physical log out of the root dbspace to another dbspace, preferably to a disk that does not contain active tables or the logical log files.

The value that is stored in the ONCONFIG parameter PHYSFILE defines the size of your physical log when the database server is initially created. When the database server is online, use the onparams utility to change the physical log location and size.

For more information about logical and physical log files, see *Informix Dynamic Server Administrator's Guide*, G229-6359.

Message log file

The database server writes status and error information to the message log file. You specify the file name and location of the message log with the MSGPATH configuration parameter.

The default name and location of the message log on UNIX is
`$INFORMIXDIR/online.log`.

On Windows, the default message log is `online.log`, which normally resides in the installation directory, for example, `C:\Program Files\IBM\IBM Informix Dynamic Server\11.50\`.

Reserved pages in root DB space

The first 12 pages of the initial chunk of the root dbspace are reserved pages. Each reserved page contains specific control and tracking information that is used by the database server. You can obtain a listing of the contents of the reserved pages by executing the **oncheck -pr** command.

2.5 Logical database structures

In this section, we describe the objects in the SQL Server and Informix logical database structures.

2.5.1 SQL Server

SQL Server logical database structures consist of pages and extents:

- ▶ Page

The page size inside an SQL Server data file is 8192 bytes (8 KB). The page size is extendable in SQL, Version 2005 and later. The data file is logically divided into pages. Disk I/O operations are performed at the page level.

- ▶ Extent

Pages are organized into extents that have a fixed size of eight contiguous pages (64 KB). One page within an extent is allocated whenever a table or index is created in an SQL Server database.

There are two type of extents in SQL Server:

- Uniform extents are owned by a single object; all pages in the extent can only be used by the owning object.
- Mixed extents are shared and owned by a separate object.

Figure 2-5 shows the concept of extents in SQL Server.

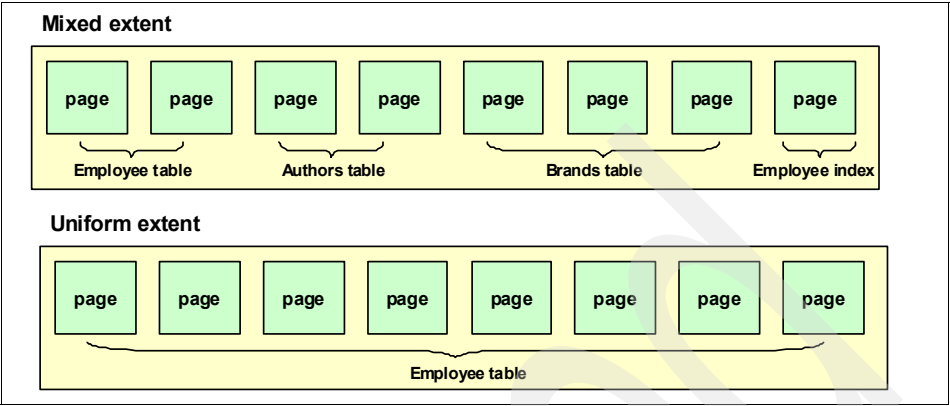


Figure 2-5 SQL Server extents and pages

2.5.2 Informix

Informix logical database structures include pages, extents, table spaces, and dbspaces. Figure 2-6 illustrates an overview of the Informix logical database structure.

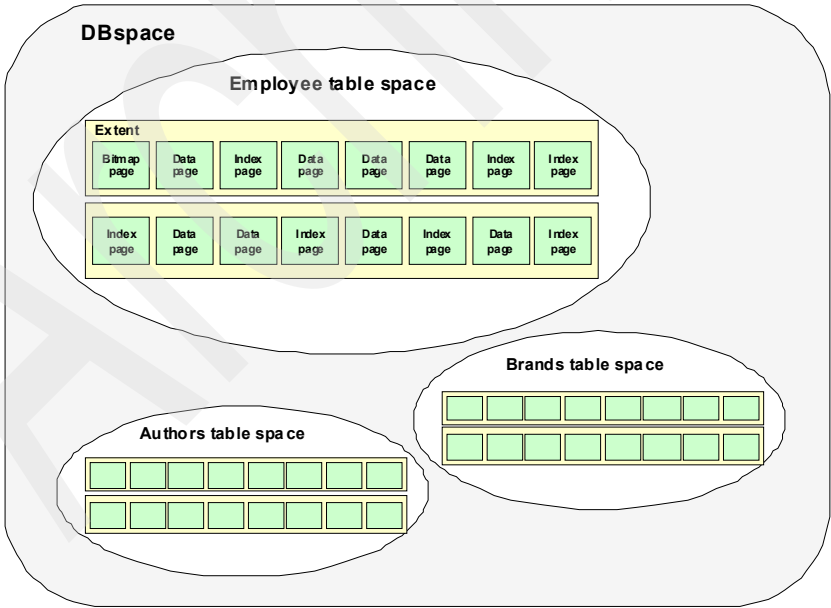


Figure 2-6 Informix logical database structure overview

Page

A *page* is the minimum I/O that is read/written to disk. The default size of a page is 2 KB on most UNIX systems and 4 KB on AIX and Windows. Informix introduced configurable page sizes in Informix Version 10 and later. Configurable page sizes allow users to create dbspaces with multiple page sizes up to 16 KB.

Extent

Disk space for a table or index is allocated in units called *extents*. An extent is an amount of contiguous pages on disk. Extent size is specified for each table or index when the table or index is created.

The default size of the initial extent and the next extent is four times the size of the page in the dbspace. When the number of extents within a table space becomes large, Informix attempts to compensate by doubling the value on the partition page. It doubles the next extent size after every 16 extents.

Table space

An Informix *table space* is a logical collection of extents that are allocated for a database object. An Informix thread searching for a place to insert data within a table space uses that table's bitmap pages to quickly identify a page with sufficient space. Along with inserts, the following Informix actions require bitmap searches:

- ▶ Updates whose resulting *tuple* (the internal term for a row) no longer fits into the existing slot
- ▶ Index operations that allocate new or free table space pages, such as B-tree splits and index creations
- ▶ Table space usage reports
- ▶ Index and data checkers

Dbspace

A *dbspace* is a logical collection of one or more chunks. It can have between one and 32,767 chunks, and a dbspace chunk can be a cooked file or a raw device.

The Informix dbspace characteristics are important:

- ▶ Informix dbspaces can contain the objects of one or more databases.
- ▶ Every Informix instance contains a default dbspace called the *root* dbspace, and logical and physical logs are created in the root dbspace, by default.

- ▶ A dbspace can be of the type dbspace, temporary dbspace, simple blobspace, smartblob space, and external dbspace.
- ▶ An Informix dbspace is a regular dbspace that can contain tables, indexes, logical logs, and physical logs. For better performance, move the logical and physical logs from root dbspace to user-defined dbspace for better performance.

You need to create at least one user dbspace before you create an Informix database. Specify the name of the default dbspace for each database, with the `<dbspace>` option, when you create the database (with the CREATE DATABASE command). If you do not specify the in `<dbspace>` option at the time of the database creation, the database is created in the default root dbspace. You must always specify a dbspace for the database instead of using the default.

Informix has the following types of dbspace:

- ▶ A *temporary dbspace* is a dbspace reserved for the storage of temporary tables. This temporary dbspace is temporary only in the sense that the database server does not preserve any of the dbspace contents when the database server shuts down abnormally.

The database server does not perform logical or physical logging for temporary dbspaces. Because temporary dbspaces are not logged physically, fewer checkpoints and I/O operations occur, which improves performance.

- ▶ A *blobspace* is a type of dbspace that is composed of one or more chunks that store only simple large objects. Simple large objects consist of TEXT or BYTE data types from any columns or any tables (from any database).

A blobpage is the basic unit of storage for blob data types stored in a blobspace. It is configured to be a multiple of the system page size. The database server writes data stored in a blobspace directly to disk. That is, this data does not pass through resident shared memory.

- ▶ An *SBspace* is a special type of dbspace composed of one or more chunks that store smart large objects that are of type BLOB (binary large object), CLOB (character large object), or a user-defined data type (UDT).

Smart large object pages have the same size and format as standard system data pages. If buffering is turned on in the database, the database server uses private buffers in the virtual portion of shared memory for buffering sbospace pages.

2.6 Data dictionary and system catalog

Every RDBMS has a form of metadata that describes the database and its objects. Essentially, the metadata contains information about the logical and physical structure of the database, integrity constraints, user and schema information, authorization, privilege information, and so on.

2.6.1 SQL Server

In SQL Server, metadata is stored in the master database and is accessible using a special schema called INFORMATION_SCHEMA.

2.6.2 Informix

In Informix, the data dictionary is divided into two parts:

- ▶ System catalog tables
- ▶ System monitoring interface (SMI) database

System catalog tables

In Informix, the system catalog tables are automatically created when you create a database. Each system catalog table contains specific information about elements in the database.

System catalog tables track objects, such as the following database objects:

- ▶ Tables, views, sequences, synonyms, and sequence objects
- ▶ Columns, constraints, indexes, and fragments
- ▶ Triggers
- ▶ Procedures, functions, routines, and associated messages
- ▶ Authorized users and privileges
- ▶ User-defined routines
- ▶ Data types and casts
- ▶ Aggregate functions
- ▶ Access methods and operator classes
- ▶ Inheritance relationships
- ▶ External optimizer directives

System monitoring interface

In addition to the system catalog tables, Informix also maintains a separate system database called the *system monitoring interface* (SMI) database. Each Informix instance contains one SMI database. It is created when you initialize the database server instance for the first time.

The SMI database consists of tables and pseudo-tables that the database server maintains automatically. While the SMI tables appear to the user as tables, they are not recorded on disk like normal tables. Instead, the database server constructs the tables in memory, on demand, based on information in shared memory at that instant. When you query an SMI table, the database server reads information from these shared memory structures. Because the database server updates the data in shared memory continually, the information that the SMI provides lets you examine the current state of your database server.

The SMI tables provide information about the following topics:

- ▶ Auditing
- ▶ Checkpoints
- ▶ Chunk I/O
- ▶ Chunks
- ▶ Database-logging status
- ▶ Dbspaces
- ▶ Disk usage
- ▶ Environment variables
- ▶ Extents
- ▶ Locks
- ▶ Networks
- ▶ SQL statement cache statistics
- ▶ SQL tracing
- ▶ System profiling
- ▶ Tables
- ▶ User profiling
- ▶ Virtual-processor CPU usage

The data in the SMI tables changes dynamically as users access and modify databases that the database server manages.

You can query SMI tables directly with select statements. Informix system monitoring utilities, such as **onstat**, also read data from SMI tables.

2.7 Transactional model

The transactional model is a collection of SQL statements that are treated as a single unit of work. All the SQL statements that you issue in an American National Standards Institute (ANSI)-compliant database are contained automatically in transactions. With a database that is not ANSI compliant, transaction processing is an option.

In a database that is not ANSI compliant, a transaction is enclosed by a `BEGIN WORK` statement and a `COMMIT WORK` or a `ROLLBACK WORK` statement. However, in an ANSI-compliant database, the `BEGIN WORK` statement is unnecessary, because all statements are contained automatically in a transaction. You need to indicate only the end of a transaction with a `COMMIT WORK` or `ROLLBACK WORK` statement.

This section describes the transactional models of both SQL Server and Informix.

2.7.1 SQL Server

In SQL Server, explicit transactions are indicated using the following key words:

- ▶ `BEGIN TRANSACTION`
- ▶ `SAVE TRANSACTION`
- ▶ `COMMIT TRANSACTION`
- ▶ `ROLLBACK TRANSACTION`

An implicit transaction is started automatically when those explicit key words are omitted. Such transactions need to be explicitly completed with a `COMMIT` or `ROLLBACK` statement.

SQL Server allows you to roll back a transaction using savepoints with the `SAVE TRANSACTION` statement.

2.7.2 Informix

The Informix transaction boundaries are marked explicitly with the following reserved words:

- ▶ `BEGIN WORK`
- ▶ `COMMIT WORK` or `ROLLBACK WORK`

Note that after a transaction begins for a given connection, it stays open until a `COMMIT` or `ROLLBACK` is executed. In other words, if an ESQL program begins a transaction and then calls a stored procedure, the stored procedure executes within the same transaction. If the stored procedure issues a `COMMIT`, the whole transaction is committed. The calling ESQL program must not issue a `COMMIT` or `ROLLBACK`. An Informix transaction that begins with a `CONNECT` statement explicitly must also have a `BEGIN WORK` statement that follows the `CONNECT` statement.

Multiple concurrent transactions

The WITH CONCURRENT TRANSACTION clause supports the concept of multiple concurrent transactions, where each connection can have its own transaction and the COMMIT WORK and ROLLBACK WORK statements affect only the current connection.

You can use the Informix SET CONNECTION *connection_name* statement to switch between transactions. Committing or rolling back each named transaction does not commit or roll back other active transactions.

Example 2-1 shows a simple example. In this example, we commit transaction A and roll back transaction B.

Example 2-1 Informix multiple concurrent transactional models

```
$CONNECT TO "stores7@test_100" AS "A" WITH CONCURRENT TRANSACTION;
$CONNECT TO "stores8@test_100" AS "B" WITH CONCURRENT TRANSACTION;

$SET CONNECTION "A";
$BEGIN WORK;
$SET CONNECTION "B";
$BEGIN WORK;

$SET CONNECTION "A";
$SELECT fname, lname INTO :fname, :lname FROM customer where customer_num=101;
printf("%s %s\n",fname, lname);

$SET CONNECTION "B";
$SELECT fname, lname INTO :fname, :lname FROM customer where customer_num=101;
printf("%s %s\n",fname, lname);
$ROLLBACK;

$SET CONNECTION "A";
$COMMIT WORK;
```

Savepoints

Informix 11.50 introduces the *savepoint* feature, which enables applications to roll back a transaction to a predetermined marker. Savepoints are named markers within a database transaction. In the case of an error, the transaction logic can specify that the transaction roll back to a savepoint. The Informix implementation of savepoints follows the SQL-99 standard.

Example 2-2 on page 42 demonstrates the usage of the SAVEPOINT statement in a transaction connected to the stores_demo database. The example sets a savepoint named *newcustomer* before inserting a new customer and sets another savepoint named *neworder* before adding the new order for this new

customer. Setting another savepoint named *newcustomer* with the UNIQUE keyword results in an error.

Example 2-2 Sample code using SAVEPOINT statement

```
Database stores_demo;  
Begin work;  
  
Savepoint newcustomer;  
Insert into customer (customer_num, fname, lname)  
values (0, "John","Doe");  
// Next statement results in an error because there is already a  
// newcustomer savepoint  
Savepoint newcustomer UNIQUE;  
Savepoint neworder;  
Insert into orders (order_num, customer_num) values (2000, 125);  
Insert into items (item_num, order_num, quantity, stock_num, manu_code)  
values (3, 2000, 2, 4, "HSK");
```

The ROLLBACK statement with the optional TO SAVEPOINT clause undoes changes that occurred since a particular savepoint was set. The statement has the following syntax:

```
ROLLBACK TO SAVEPOINT svpt1
```

For more information about savepoints, refer to “Expand transaction capabilities with savepoints in Informix Dynamic Server” by Uday B. Kale in IBM developerWorks®, 26 March 2009, which is available at this website:

<http://www.ibm.com/developerworks/data/library/techarticle/dm-0903idssavepoints/index.html>

2.8 Database server communications

In this section, we give an overview of the communication architecture that enables client/server communication in SQL Server and Informix environments. We also describe the tools that are used to connect to the database.

2.8.1 SQL Server

SQL Server provides the SQL Server Management Studio as the major interface for accessing, configuring, managing, and administering all components of SQL Server. PowerShell and Transact-SQL are also common ways to interface and communicate with SQL Server.

2.8.2 Informix

Informix provides the dbaccess utility, a cursor-based utility installed on the database server for local SQL connections to run ad hoc queries. Application programmers can install the IBM Informix Client Software Development Kit (Client SDK) on the remote clients to connect to Informix databases remotely. You can also connect to Informix databases using Java Database Connectivity (JDBC).

The dbaccess utility

An Informix installation includes the dbaccess utility. The dbaccess utility is a cursor-based tool that provides a user interface for entering, executing, and debugging SQL statements and Stored Procedure Language (SPL) routines. You can also start dbaccess in menu mode and select options from the menus.

As examples, you can use dbaccess for the following tasks:

- ▶ Ad hoc queries that are executed infrequently
- ▶ Connecting to one or more databases and displaying information about a database
- ▶ Displaying system catalog tables and the information schema
- ▶ Practicing the statements and examples that are provided in the *IBM Informix Guide to SQL: Tutorial*, G229-6427

IBM Informix Client Software Development Kit (Client SDK)

The IBM Informix Client Software Development Kit (Client SDK) includes several application programming interfaces (APIs) that developers can use to write applications for Informix database servers in ESQL, C, Java, and .Net.

Applications that run on client computers require Informix Connect to access database servers. Informix Connect is a connectivity product that is composed of runtime libraries that are included in the Client SDK. All client/application access to Informix database servers using Client SDK is through the Structured Query Language Interface (SQLI) protocol.

The Client SDK consists of the following components:

- ▶ IBM Informix .NET provider (Windows only)
- ▶ IBM Informix add-ins for visual studio 2003 and 2005
- ▶ IBM Informix ESQL/C with XA support
- ▶ The finderr utility on UNIX systems
- ▶ The Informix error messages utility on Windows systems

- ▶ The global security kit
- ▶ IBM Informix Object Interface for C++
- ▶ IBM Informix Open Database Connectivity (ODBC) driver with MTS support
- ▶ IBM Informix Object Linking and Embedding (OLE) DB provider (Windows only)
- ▶ The ILogin utility (Windows only)
- ▶ IBM Informix password communications support module (CSM)

SQLHOSTS file

Informix supports TCP/IP, IPX/SPX, sockets, shared memory, stream pipe, and named pipe connections. These connections are stored in the Informix SQLHOSTS file (on UNIX).

The `sqlhosts` file resides in the `$INFORMIXDIR/etc` directory, by default. As an alternative, you can set the `INFORMIXSQLHOSTS` environment variable to the full path name and file name of a file that contains the `sqlhosts` file information. On UNIX, each computer that hosts a database server or a client must have an `sqlhosts` file. Each entry (each line) in the `sqlhosts` file contains the `sqlhosts` information for one database server.

Example 2-3 depicts an `sqlhosts` file on a UNIX server.

Example 2-3 Sample sqlhosts file entries

dbservername	nettype	hostname	servicename	options
inst_tcp	onsoctcp	localhost	inst_svc	b=8192
inst_tcp	onsoctcp	192.1.1.11	1523	

On Windows, SQLHOSTS is registry keys. It contains client/server connectivity information that enables a client application to find and connect to any Informix database server in the network. When you install the database server, the setup program creates the following key in the Windows registry:

HKEY_LOCAL_MACHINE\SOFTWARE\INFORMIX\SQLHOSTS

This branch of the HKEY_LOCAL_MACHINE subtree stores the `sqlhosts` information. Each key on the SQLHOSTS branch is the name of a database server. When you click the database server name, the registry displays the values of the HOST, OPTIONS, PROTOCOL, and SERVICE fields for that particular database server.

Each computer that hosts a database server or a client must include the connectivity information either in the `sqlhosts` registry key or in a central registry.

When the client application runs on the same computer as the database server, they share a single sqlhosts registry key.

On Windows, use setnet32 to manage the sqlhosts information. Figure 2-7 shows the sample of sqlhosts information on a Windows server.

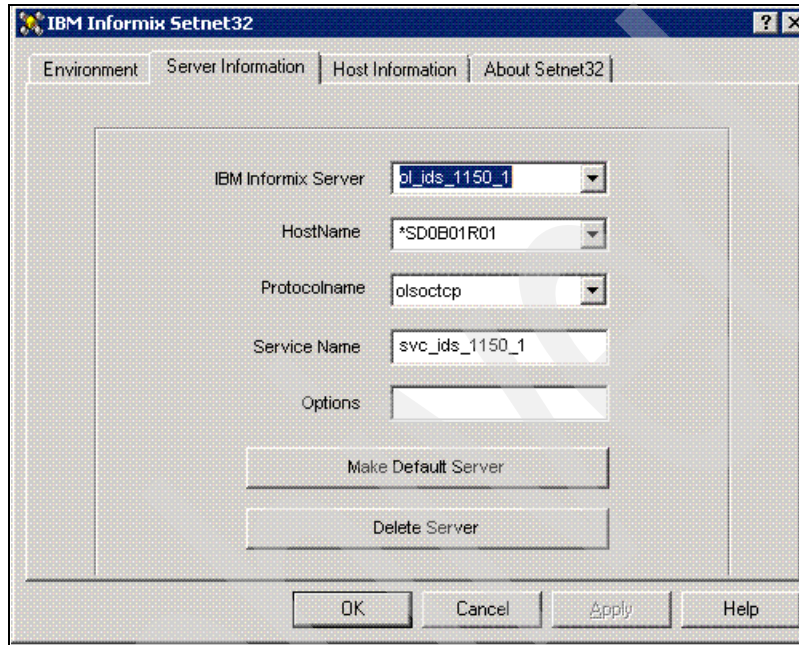


Figure 2-7 Sample sqlhosts information on a Windows server

Migration methodology

Database migration requires proper planning, regardless of how it is performed or the resources that are used. You can avoid unexpected delays in project execution by following well-planned migration procedures. In this chapter, we describe an IBM migration methodology and the resources that are available to assist in making that migration successful.

We discuss the following topics in this chapter:

- ▶ An IBM migration methodology
- ▶ Preparing for the migration
- ▶ Migrating
- ▶ Testing the migration
- ▶ Implementing the new environment
- ▶ Related information resources

3.1 An IBM migration methodology

The IBM Data Management Services Migration Center has developed a best-practices methodology to help clients develop a plan for how to best migrate their database environment to Informix. You can use this methodology to perform your own migration project, or you can contract the fee-based services of the IBM Migration Center.

The migration team provides assistance with the following tasks:

- ▶ Selection of application areas to migrate
- ▶ Assessment of migration complexity
- ▶ Ballpark migration estimates delivered in hours
- ▶ Sample database and code migrations
- ▶ Migration tool selection and demonstrations
- ▶ Problem resolution related to the migration
- ▶ Selection of migration services
- ▶ Database administration
- ▶ SQL application development comparisons

The IBM Data Management Services Migration Center consists of a team of migration specialists who can perform the entire migration or who can partner with your team for a more cost-effective solution. Alternatively, the Migration Center can simply provide direction and advice on an as-needed basis at an hourly rate.

You can contact the IBM Data Management Services Migration Center team through your IBM marketing representative.

You can obtain more information about the IBM Data Management Services Center at this website:

<http://www.ibm.com/software/solutions/softwaremigration/dbmigteam.html>

The IBM migration methodology consists of the following primary phases:

- ▶ Preparation
- ▶ Migration
- ▶ Test
- ▶ Implementation and cutover

Typically, a migration project is an iterative process that consists of multiple rounds of migration, testing, and refinement. Each phase has specific objectives and deliverables, which we describe in the following sections of this chapter.

3.2 Preparing for the migration

The *migration preparation* phase includes all the activities that take place before you set up the system and migrate the database, database objects, and data. In this phase, it is essential to focus on system architecture analysis and information gathering to make decisions about how the migration will be performed.

The preparation phase includes these primary objectives:

- ▶ Developing an overall migration strategy
- ▶ Performing the initial project planning
- ▶ Assessing the benefits and risks of various migration options

The preparation phase includes these major tasks:

- ▶ Performing the migration assessment
- ▶ Understanding and selecting migration tools
- ▶ Estimating the required effort
- ▶ Preparing the environment
- ▶ Receiving education about Informix

3.2.1 Performing the migration assessment

Planning a migration begins with an assessment of the current system and environment, as well as an understanding of the resources that can be used.

An accurate profile of the system-wide architecture is key to the success of the project. The assessment begins with collecting information about both the source and target environments. This detailed information is used to determine the scope of the migration and is the major input for the project planning process.

Before undertaking a migration project, you need to complete several planning activities:

- ▶ Perform a hardware and software architectural assessment:
 - Decide on the target hardware platform for the production system.
 - Understand the workload characteristics and requirements.
 - Know the number of concurrent users.
 - Take an inventory of software and hardware upgrades.
 - Determine the machine on which the migration will be performed.
- ▶ Investigate the scope, duration, and cost of the project:
 - Which application areas will be migrated?
 - How complex is the migration?
 - How many database objects and applications will be migrated?

- What are the differences between the source and target databases?
- How long will the migration take?
- What is an estimate of the proposal for work?
- ▶ Identify business dependencies:
 - Are there timeline or business cycle requirements?
 - Do licensing agreement renewals affect schedules?
- ▶ Identify the people resources and skills required:
 - Will resources be in-house, outsourced, or a combination of both?
 - Do in-house resources have skills for the source and target databases?
 - Are in-house resources available to perform the migration?
 - Are in-house resources available to support an outsourced migration?
- ▶ Identify the services and tools that can be used during the migration.

As you plan for the migration, you also need to devise a strategy for the following key issues:

- ▶ Coexistence of source and target databases
- ▶ Ongoing application development
- ▶ Change management
- ▶ Performance requirements
- ▶ Tool selection
- ▶ Naming conventions
- ▶ Database physical design
- ▶ Standards definitions
- ▶ Data migration strategy
- ▶ Security planning
- ▶ Cutover strategy

3.2.2 Understanding and selecting migration tools

A tool that is specifically designed for migration usually provides functions for creating Data Definition Language (DDL) scripts for the target database objects, unloading data, and creating DDL scripts for loading data. A good migration tool can save a significant amount of time in a migration process.

IBM Migration Toolkit is designed specifically to make a migration as simple as possible. IBM Migration Toolkit helps in the migration from SQL Server to Informix. You can use this toolkit to generate DDL scripts that create database objects, including tables, indexes, views, triggers, stored procedures, and user-defined functions. The toolkit also aids in moving the data from the original source system to the target Informix database. For example, the IBM Migration Toolkit can either connect directly to the source system and extract the database

structure and database object definitions, or it can use a valid database schema script that is extracted by other tools.

The IBM Migration Toolkit is a free migration tool that you can download from this website:

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

Note that the IBM Migration Toolkit is a no-cost, as-is tool and is currently in maintenance mode.

3.2.3 Estimating the required effort

An accurate estimate of the effort, resources needed, and total costs requires knowledge of the products, applications, and migration experience.

You can use a migration tool as an assessment tool to determine the complexity of the migration. Using a good migration tool in this manner can highlight complex stored procedures or SQL statements that might require manual migration effort. A working knowledge of the migration tool is mandatory when using it in this manner. Thus, you need to factor training costs and the time that is required for DBAs and users into your estimates.

Each migration differs; however, general signs can indicate the overall complexity and effort expected for a migration. For instance, in applications that use stored procedures frequently, the number and complexity of the stored procedures to be migrated greatly affect the migration time. The same concept applies to the use of special data types and large objects. Physical requirements (use of raw or formatted disk areas, spaces, and nodes) can also represent a large amount of work, especially if the data grows significantly over time. If there is replication or high availability involved, the migration needs a thorough study to understand the existing database environment, current topology, and storage requirements to ensure that you design the database properly and implement similar technology with the target database system.

The IBM migration team migration estimate

Clients frequently request that the IBM migration team provide a time estimate for the migration of database objects and applications. The team uses prior experiences to deliver an estimate for a minimum and maximum range for the number of hours that the project is expected to take. To deliver that estimate, the team provides a questionnaire, and the client collects and returns metric information for the objects to be migrated.

Clients collect and return the following types of metrics:

- ▶ Number of database objects, including tables, indexes, and views
- ▶ Number of database application objects (including packages, procedures, triggers, and user defined functions), average lines of code, and SQL statements for each
- ▶ Number and language of application programs, including lines of code and average number of SQL statements per module
- ▶ Volume of production data to be migrated

In addition to metrical information and project tasks, the following additional factors can influence the size and complexity of a migration:

- ▶ Amount and type of proprietary SQL used
- ▶ Quality of data
- ▶ Existence of system documentation
- ▶ Database design requirements, such as high availability and replication
- ▶ Third-party software dependencies
- ▶ Operating system and hardware platform changes
- ▶ Availability of a dedicated machine for migration development
- ▶ Extent of code freeze enforcement during migration
- ▶ Length of the cutover window
- ▶ Skill level of individuals performing the migration

3.2.4 Preparing the environment

Before starting the migration, you need to set up the target development environment. Ensure that all hardware and software prerequisites are met, that the network is configured properly, and that there is enough hardware to support the migration and development properly.

In this step, you complete the following tasks:

- ▶ Configure the operating system
- ▶ Configure disk storage
- ▶ Install Informix
- ▶ Create an Informix instance
- ▶ Install and configure tools
- ▶ Configure Informix
- ▶ Configure the database environment and registry variables
- ▶ Test the environment with a sample application

3.2.5 Receiving education about Informix

It is important that members of the migration team have a good understanding of important Informix features, such as transaction management, locking, authorities, data recovery techniques, and memory management. Although most database vendors implement similar sets of database features, there are often substantial differences in how they behave.

The fastest way to prepare your technical staff to work effectively with Informix is through education. The following web page provides numerous references to sites within IBM that offer classroom training, online tutorials, and reference materials:

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

3.3 Migrating

This phase is the core of the migration project. We introduce the steps that are required to migrate a database and application to Informix in this section. The methods employed for a migration can vary, but they are generally divided into the following primary tasks:

- ▶ Migrating and designing the database
- ▶ Calibrating the planned strategy
- ▶ Migrating applications

3.3.1 Migrating and designing the database

The first step in the migration process involves moving or duplicating the structure of the source database into an Informix database. In this process, you must address the differences between the source and destination structures. These differences can result from various interpretations of SQL standards or from the addition and omission of particular functions. You can address many differences by altering the existing syntax. However, in certain cases, you will need to add custom functions and modify the application.

In this phase, you plan and design the final Informix physical database design for the production system. The physical layout of the Informix database is critical to the success of the project. You must implement the layout according to best practices to ensure optimal performance. In addition, you must structure the layout to support future maintenance and to provide maximum flexibility to meet future requirements, without compromising existing business processes and program logic.

The most popular means of migrating a database structure is through the use of a migration tool. These tools connect to and extract structural information from the source database, and they can also convert it into a format acceptable by the target database system. You can use the IBM Migration Toolkit to perform the migration using this method.

Capturing the definition of database objects can often occur at the same time that the database structure is captured, if the objects are written in an SQL-like procedural language and stored within the database. A good portion of the database object migration is automated.

The IBM Migration Toolkit is a no-cost tool for converting database objects, such as tables, data types, constraints, indexes, views, stored procedures, user-defined functions, built-in functions, and sequences. However, the tool is in maintenance mode; therefore, certain database features are not supported in the tool. In addition, the creation of dbspaces, and the placement of tables and indexes within dbspaces, along with disk layout is a manual part of the physical design process. While the IBM Migration Toolkit can help convert procedures and user-defined functions to Informix, the migration of these application objects usually requires a certain amount of manual intervention, depending on the SQL constructs that are used. The IBM Migration Toolkit reports those statements that cannot be converted automatically.

The migration of database objects requires testing of the resulting objects, implying that test data needs to be available before testing. A preliminary data migration effort, therefore, is required to complete this migration step. After the object migration is completed, adjustments might still be required. As an example, you might need to address issues, such as identifier length. During this phase, you set up and use a test database for migration development and functional testing of converted objects.

3.3.2 Calibrating the planned strategy

In the *calibration phase*, you validate the planned strategy for the migration, including quality assurance review, standards compliance, and performance. During the calibration, you migrate a few selected programs as samples to ensure that the code produced complies with requirements and possesses characteristics that ensure the future maintainability of the applications. Typically, 20 to 60 programs of a selected group are migrated.

There are several considerations in defining the system to use in the calibration. The primary requirement, however, is that the chosen programs are representative of how the entire application portfolio uses the source database and application languages. These applications are also expected to provide baseline performance and data usage that can be generalized easily. The

generalized results are then used to predict the ability to migrate each of the other application subsystems, as well as their likely post-migration performance characteristics.

During this phase, there is a review of the data transfer requirements and design of a data transfer strategy. The data transfer strategy must ensure that all the production data can be transferred into the production system within the cutover time frame.

In addition, detailed resource and schedule planning is finalized in the calibration, as well as the test strategy for user acceptance and parallel testing.

3.3.3 Migrating applications

You incorporate insights that are gained during the calibration phase into the migration strategy. In this phase, you complete the migration of the entire portfolio of objects and applications.

Although converting the database structure and objects can be automated to a certain extent using migration tools, application code changes mostly require manual effort. Aside from the SQL language differences, you also need to understand the transaction, locking, and isolation-level differences between SQL Server and Informix. These differences influence how the application is coded and can require adjustments to run as expected.

Applications written in a high-level language with embedded SQL can be converted automatically by extracting SQL statements from the source code, adjusting them manually or with a migration tool, and reinserting them into the source.

Because this phase is mostly executed using manual techniques, expertise with both the source server and Informix is essential. You might not need to convert all SQL, because the same SQL syntax often runs on both the source and Informix databases. However, you must examine all SQL to determine whether migration is needed. In addition to syntax, you need to examine the semantics of the SQL to ensure that the SQL statement behaves the same way and returns the same results on Informix as on the source database. We discuss the complete details of application migration in the later chapters.

3.4 Testing the migration

In this section, we discuss the *test phase*, which consists of the following sub-phases:

- ▶ Refreshing the migration
- ▶ Migrating the data
- ▶ Testing

3.4.1 Refreshing the migration

Because most clients cannot freeze an application for the duration of a migration, and because the calibration programs are typically migrated early in the process, a migration refresh is required just prior to client testing and production implementation. Based on change control logs, those programs, applications, and data structures that changed during the migration phase are reconverted. This process relies heavily upon the inventory and analysis activities that took place during the assessment phase, because that effort cross-referenced programs, procedures, functions, data structures, and application code. Using the baseline as a guide, the migration team reconverts those migrated objects that were affected by ongoing development.

3.4.2 Migrating the data

Data migration is usually performed quite late in the migration process. However, you need to migrate subsets of the data with the database structure to verify that data is migrated correctly.

The process of data migration consists of the following activities:

- ▶ Unload
- ▶ Transform
- ▶ Cleanse
- ▶ Reload

This phase is accomplished by a combination of automated and manual methods.

In this phase, you can use a tool to automate the migration of the data or to generate scripts that can be used to migrate the data manually. You can also create your own data moving scripts.

In many cases, as the data is moved, it is also converted to a format that is compatible with the target database (date data and time data are good examples). This process can be quite lengthy, especially when there is a large

amount of data. Thus, it is imperative to have the data migrations well-defined and tested. In certain cases, it is still necessary to perform customized migrations of specialized data, such as time series and geospatial data. This specialization can sometimes be accomplished through the creation of a small program or script.

It is a good idea to perform tests after migrating subsets of the production data. These tests include performing simple comparisons against the source and target databases, such as row counts. You can also identify whether any transformations or adjustments are required and then implement those changes before migrating the full set of data.

When you are completely satisfied with the results of your tests, you can migrate the full set of production data. After moving all the production data from the source to the target system, correct any referential integrity issues, then perform a thorough verification of the migrated data against the source data for accuracy and completeness.

Important: If the data is encrypted, it must be decrypted manually first using the source database application programming interfaces (APIs), transferred to target database, and then encrypted.

3.4.3 Testing

The purpose of the *testing phase* is to determine the differences between the expected results (the source environment) and the observed results (the migrated environment). Then, you need to synchronize the detected changes with the development stages of the project. In this section, we describe the test objectives and a generic testing methodology that you can use to test migrated applications.

Testing methodology

The first step in the testing process is to prepare a thorough test plan. The test planning and documentation are important parts of this testing process that detail the activities, dependencies, and effort that is required to conduct the test of the migrated solution. A detailed test plan needs to describe all the test phases, the scope of the tests, validation criteria, and the time frame for the tests. It needs to list the features of the migration that are tested, as well as those features that are not tested, from both the user viewpoint of what the system does and a configuration management viewpoint. Plans need to consider the environmental needs (hardware, software, and any tools) for testing, test deliverables, pass/fail criteria for tests, and any required skills to implement the testing.

For large projects, it might be necessary to use support software to improve testing productivity (for example, the IBM Rational Functional Tester and IBM Rational Performance Tester). The IBM Rational Functional Tester provides testers with automated capabilities for data-driven testing and a choice of scripting language and an industrial-strength editor for test authoring and customization. IBM Rational Performance Tester can create, execute, and analyze tests to validate the reliability of complex e-business applications. Additionally, there are many other Rational products that might suit your testing needs. For more information about testing products, see this website:

<http://www.ibm.com/software/rational>

You need to validate all stages of the migration process by running a series of carefully designed tests. We discuss the phases and techniques to consider for testing in the remainder of this section.

Functional testing

Functional testing involves a set of tests in which the new functionality and the existing functionality of the system are tested after migration. Functional testing includes all components of the relational database management system (RDBMS), such as stored procedures, triggers, and user-defined functions, networking, and application components. The objective of functional testing is to verify that each component of the system functions as it did before the migration and to verify that new functions work properly.

Integration testing

Integration testing examines the interaction of each component of the system. All modules of the system and any additional applications (such as web, supportive modules, and Java programs) running against the target database instance must be verified to ensure that there are no problems with the new environment. The tests also need to include GUI and text-based interfaces with local and remote connections.

Performance testing

Performance testing of a target database compares the performance of various SQL statements in the target database with the statement performance in the source database. Before migrating, you need to understand the performance profile of the application under the source database. Specifically, you need to understand the calls that the application makes to the database engine. If performance issues are found, it might be necessary to revisit the physical database design and implement changes.

For detailed information about performance measurement and tuning methods that are relevant to daily database server administration and query execution, see *IBM Informix Dynamic Server Performance Guide*, SC23-7757.

Volume and load stress testing

Volume and load stress testing tests the entire migrated database under high volume and loads. The objective of volume and load testing is to emulate how the migrated system might behave in a production environment. These tests determine whether any database or application tuning is necessary.

Acceptance testing

Acceptance tests are typically carried out by the users of the migrated system. Users are simply asked to explore the system, test usability and system features, and give direct feedback. Acceptance tests are usually the last step before going into production with the new system.

Post-migration tests

Because a migrated database can be a completely new environment for the IT staff, the test plan needs to also encompass examination of new administration procedures, such as database backup/restore, daily maintenance operation, and software updates.

Data migration testing considerations

The extracting and loading process entails migration between source and target data types. You need to verify the migrated database to ensure that all data is accessible and was imported without any failure or modification that can cause applications to function improperly.

Any migration first needs to focus on data movement. Without having all the tables and data moved properly, all other migration testing is in vain. The test process needs to detect if all rows were imported into the target database, to verify that data type migrations were successful, and to check random data byte-by-byte. The data checking process must be automated by appropriate scripts. When testing data migration, perform the following tasks:

- ▶ Check IMPORT/LOAD messages for errors and warnings.
- ▶ Count the number of rows in source and target databases and compare them.
- ▶ Prepare scripts that perform data checks.
- ▶ Involve data administration staff who are familiar with the application and its data to perform random checks.

Code and application testing considerations

The most important part of the testing process is to verify that each component of the system functions as it did before migrating. You need to verify all components of the RDBMS, including views, stored procedures, triggers, application components, and security systems as to whether they work properly.

Views

After data migration testing, check all migrated views. After data is migrated, you can test the views to make sure that they work in the same way as in the source database. Depending upon the view, you can create various verification scenarios. Views can be checked easily by counting the number of rows that the views return or by comparing calculated values and, similarly, the tests performed on regular tables. More complicated views need a little more thought about how they need to be checked. However, you need to review all views by executing queries against them. You also need to test the views that are created with the intention to modify data for inserting, updating, and deleting.

Procedures and user-defined functions

You need to test all stored procedures, user-defined functions, and triggers individually before promoting them for further testing. Thus, after objects are migrated (either manually, with a migration tool, or a combination), you need to check them to make sure that they function properly.

Application code check

The scope of application testing depends on the migrated application. For custom, in-house-developed applications, start the testing process with the application queries. Test all queries independently to ensure that they return the expected results. With the application queries migrated successfully, rebuild the surrounding client programs, and test the application against the target database. Run each module of the application, and possibly each panel form, and check for errors or improper functionality. Check all the supported application connectivity interfaces, also.

Security considerations

Before going into production, check security in detail. Each database system handles security quite differently, so it is not trivial to compare the user rights between the two systems. Verify the existence and authorities of all required users to allow the appropriate persons to connect to the database. Test the access privileges and restrictions on the database objects to make sure that only users with proper authority can access them.

3.5 Implementing the new environment

After the testing phase is complete, you need to perform final testing to validate implementation of the new production environment and to gain user acceptance. Upon acceptance of the system, an additional test run of the cutover procedure is performed, usually over a weekend prior to the final cutover weekend. You need to correct any issues that are discovered during the test run. On the cutover weekend, the database is refreshed with production data, and the system goes live.

In preparation for this phase, you need to take a backup of the source production system in case there is an unexpected need to back out the new system. In addition, all database maintenance procedures, such as backups, are put into production.

In certain cases, the final cutover phase does not allow for downtime of the production system. In these cases, special planning and procedures are developed to accommodate the phasing in of the new system without affecting the production environment.

3.6 Related information resources

IBM provides information about Informix and offers various training courses. The following websites are good sources for Informix information:

- ▶ IBM database migration:
<http://www.ibm.com/software/solutions/softwaremigration/dbmigteam.html>
- ▶ IBM Migration Toolkit:
<http://www.ibm.com/software/data/db2/migration/mtk/>
- ▶ IBM Informix Information Center:
<http://publib.boulder.ibm.com/infocenter/idshelp/v115/index.jsp>
- ▶ IBM Redbooks publications:
<http://ibm.com/redbooks>

3.6.1 IBM training

The newest offerings are available from IBM training to support your training needs, enhance your skills, and boost your success with IBM software. IBM offers a complete portfolio of training options, including traditional classroom, private on-site, and eLearning courses. Many of our classroom courses are part

of the IBM “Guaranteed to run” program that ensures your course will never be canceled. We have a robust eLearning portfolio, including instructor-led online (ILO) courses, self-paced virtual courses (SPVC), and traditional web-based training (WBT) courses. A perfect complement to classroom training, our eLearning portfolio offers something for every need and every budget; simply select the style that suits you.

Be sure to take advantage of our custom training plans to map your path to acquiring skills. Enjoy further savings when you purchase training at a discount with an IBM Education Pack online account, which is a flexible and convenient way to pay, track, and manage your education expenses online.

Table 3-1 shows Informix courses that IBM offers. Check your local Information Management Training website or ask your training representative for the most recent training schedule.

Table 3-1 Informix courses

Course name	Classroom code	Instructor-led online code	Self-paced virtual classroom code
Informix System Administration	IX81	3X81	2X81
Informix Database Administration: Managing and Optimizing Data	IX22	3X22	2X22
Informix Structured Query Language	IX13	3X13	N/A
Informix 11 Performance Tuning	IX40	3X40	N/A
Informix 11 Replication	IX42	3X42	N/A
Informix 11 New Features	IX30	3X30	2X30
Informix 11 Internal Architecture	IX91	3X91	N/A
Informix Warehouse: SQL Warehousing Tool and Admin Console	IXA5	3XA5	N/A

You can access the course descriptions for IT professionals and managers at this website:

http://www.ibm.com/services/learning/ites.wss/tp/en?pageType=tp_search

For scheduling and enrollment, call IBM training at 1-800-IBM-TEACH (426-8322) or go to this website:

<http://www.ibm.com/training>

3.6.2 IBM professional certification

Information Management Professional Certification is a business solution for skilled IT professionals to demonstrate their expertise to the world. Certification validates skills and demonstrates proficiency with the most recent IBM technology and solutions. Table 3-2 lists the Informix certification exam.

Table 3-2 Informix certification exam

Exam number	Exam name	Certification title
918	System Administration for IBM Informix Dynamic Server V11	IBM Certified System Administrator - Informix Dynamic Server V11

For additional information about this exam, go to this website:

<http://www.ibm.com/certify/tests/ovr918.shtml>

3.6.3 Information Management Software Services

When implementing an Information Management solution, it is critical to have an experienced team involved to ensure that you achieve the results you want through a proven, low risk delivery approach. The Information Management Software Services team has the capabilities to meet your needs, and it is ready to deliver your Information Management solution in an efficient and cost-effective manner to accelerate your return on investment.

The Information Management Software Services team offers a broad range of planning, custom education, design engineering, implementation and solution support services. Our consultants have deep technical knowledge, industry skills, and delivery experience from thousands of worldwide engagements. With each engagement, our objective is to provide you with reduced risk and an expedient means of achieving your project goals. Through repeatable services offerings, capabilities, and best practices leveraging our proven methodologies for delivery, our team has been able to achieve these objectives and has demonstrated repeated success on a global basis.

Table 3-3 on page 64 shows the key services resources that are available for Informix V11.

Table 3-3 Informix key services

Information Management Services offering	Description
Quick Start Services for Informix 11	A Quick Start Service is designed to get a new database environment up and running quickly. Additionally, assistance is provided with installation and configuration to help you exploit the newest functionality in IBM Informix 11.

For more information, visit our website:

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

3.6.4 IBM Software Accelerated Value program

The IBM Software Accelerated Value program provides support assistance for issues that are outside the normal “break-fix” parameters that are addressed by the standard IBM support contract. This program offers clients a proactive approach to support management and issue resolution assistance. An assigned senior IBM support expert who knows your software and understands your business needs will help you. The Accelerated Value Program offers the following benefits:

- ▶ Priority access to assistance and information
- ▶ Assigned support resources
- ▶ Fewer issues and faster issue resolution times
- ▶ Improved availability of mission-critical systems
- ▶ Problem avoidance through managed planning
- ▶ Quicker deployments
- ▶ Optimized use of in-house support staff

To learn more about the IBM Software Accelerated Value program, contact your local Accelerated Value marketing representative at this website:

<http://www.ibm.com/software/support/acceleratedvalue/contactus.html>

3.6.5 Protect your software investment with Software Subscription and Support

Complementing your software purchases, Software Subscription and Support gets you access to our world-class support community and product upgrades, with every new license. Extend the value of your software solutions and transform your business to be smarter, more innovative, and cost-effective when you renew your Software Subscription and Support. Making sure that you renew

on time ensures that you maintain uninterrupted access to innovative solutions that can make a real difference to your company's bottom line.

Go to this website to learn more about Software Subscription and Support:

<http://www.ibm.com/software/data/support/subscriptionandsupport>

Archived

SQL considerations

In this chapter, we discuss the SQL functionality and syntax for SQL Server and the Informix. In particular, we cover:

- ▶ Data Definition Language (DDL) basics and differences
- ▶ Data Manipulation Language (DML) basics and differences
- ▶ Differences between SQL Server Transact-SQL (T-SQL) and Informix Stored Procedure Language (SPL)
- ▶ Available security options, including differences in implementation
- ▶ Database management system (DBMS) behavioral differences specific to application programming

Although the focus for this chapter is SQL Server 2008 and Informix Version 11.50, you can use the information in this chapter for migrations that are at other version levels.

4.1 DDL

The American National Standards Institute (ANSI) and the International Organization of Standardization (ISO) have jointly established a set of industry standards for the Structured Query Language (SQL). Most relational database management systems (RDBMSs) are either fully compliant or conform to one of the SQL standards.

Many features of Informix comply with the SQL-92 Intermediate and Full Level and X/Open SQL Common Applications Environment (CAE) standards. Informix also contains many core features of the SQL99 standard, and many features that are beyond the SQL99 Core are partially or completely supported. However, there are no commands to change the SQL compliance level.

SQL Server complies with the entry level of the ANSI SQL-92 standard and with the Federal Information Processing Standards (FIPS 127-2), as established by the US National Institute of Standards and Technology (NIST). You can use the SET FIPS_FLAGGER and SET ANSI_DEFAULTS commands to change the level of compliance and SQL92 behavior.

Because both SQL Server and Informix are SQL-92 compliant at a certain level, you can convert many SQL statements and queries written for SQL Server to Informix without modification. However, certain SQL syntax and semantic differences do exist across both DBMSs.

In this section, we describe how SQL Server and Informix handle Data Definition Language (DDL) statements, such as create database, create and alter tables, views, indexes, sequences, synonyms, procedures, functions, and triggers.

4.1.1 Database creation

SQL Server supports the creation of databases through client/application application programming interfaces (APIs), Object Explorer, and T-SQL. Informix supports the creation of databases through dbaccess DDL, client/application APIs, and the OpenAdmin Tool (OAT).

SQL Server automatically creates four default system databases for internal use when the product is installed. They are called master, model, msdb, and tempdb. Informix also creates four default system databases for internal/DBA use upon installation:

- **sysmaster**

The sysmaster database contains the system-monitoring interface (SMI) tables. The SMI tables provide information about the state of Informix. You

can query these tables to identify processing bottlenecks, determine resource usage, track session or Informix instance activity, and so on.

- **sysadmin**

The **sysadmin** database includes the tables that contain and organize the Scheduler tasks and sensors, store data collected by sensors, and record the results of Scheduler jobs and SQL administration API functions. A database administrator can create, manage, and monitor DBA tasks contained in this database.

- **sysuser**

The **sysuser** database is used for Pluggable Authentication Module (PAM) authentication in Informix server-to-server communication.

- **sysutils**

The Informix utility ON-Bar maintains a history of backup and restore operations in the **sysutils** database and an extra copy of the backup history in the emergency boot file. ON-Bar uses the **sysutils** database in a warm restore when only a portion of the data is lost.

To create a database in both SQL Server and Informix, you use the **CREATE DATABASE** statement. Both DBMSs do not require any options to this command aside from the database name. However, because the notion of a logged database is fundamental to SQL Server, all databases that are created in SQL Server are, by default, logged. In Informix, by default, database are not logged unless explicitly specified. For example, consider the following command:

```
CREATE DATABASE BOOKDB;
```

In SQL Server, the previous command created a logged database called **bookdb**. However, in Informix, the command creates a non-logging database; if you lose the database because of a hardware failure, you lose all data alterations since the last backup.

To create a logged database in Informix, use the **WITH LOG** clause:

```
CREATE DATABASE bookdb WITH LOG;
```

The previous command creates a database with unbuffered logging, which is the default. In the event of a failure, you lose only uncommitted transactions.

Informix also lets you create a database using buffered logging. If you lose the database, you lose few or possibly none of the most recent alterations in the database; only transactions contained in the Informix Log buffer are lost. In return for this small risk, performance during alterations improves slightly.

The following command is an example of how to create a database with buffered logging:

```
CREATE DATABASE bookdb WITH BUFFERED LOG;
```

Buffered logging is best for databases that are updated frequently (so that speed of updating is important), but you can re-create the updates from other data in the event of a failure. You can use the SET LOG statement to alternate between buffered and regular logging.

Informix supports both ANSI and non-ANSI databases. However, SQL Server transaction behavior is more closely aligned with ANSI databases. In SQL Server and Informix ANSI databases, a transaction implicitly begins at the start of the program and ends after execution of a commit or rollback statement.

To create an ANSI-compliant database in Informix, you must add the WITH LOG MODE ANSI clause to the CREATE DATABASE statement.

The following command creates an ANSI-complaint database called bookdb:

```
CREATE DATABASE bookdb WITH LOG MODE ANSI;
```

The CREATE DATABASE statement is an extension to the ANSI/ISO standard for SQL. ANSI-compliant databases differ from databases that are not ANSI-compliant in several ways:

- ▶ All SQL statements are automatically contained in transactions.
- ▶ All databases use unbuffered logging.
- ▶ For sessions, the default isolation level is REPEATABLE READ.
- ▶ Default privileges on objects differ from those in databases that are not ANSI compliant. When you create a table or a synonym, other users do not receive access privileges (as members of the PUBLIC group) to the object, by default.
- ▶ All DECIMAL data types are fixed-point values. If you declare a column as DECIMAL(p), the default scale is zero, meaning that only integer values can be stored. In a database that is not ANSI-compliant, DECIMAL(p) is a floating-point data type of a scale large enough to store the exponential notation for a value.
- ▶ Owner naming is enforced. You must qualify with the owner name any table, view, synonym, index, or constraint that you do not own. Unless you enclose the owner name between quotation marks, alphabetic characters in owner names default to uppercase. To prevent this upshifting of lowercase letters in unlimited owner names, set the ANSIOWNER environment variable to 1.

By default, SQL Server creates a database in the <Drive>:\Program Files\Microsoft SQL Server\MSSQL10.<SERVERNAME>\MSSQL\DATA folder. The *Drive* and *SERVERNAME* are specific to your SQL Server installation and setup. By default,

Informix creates a database in the path for the device containing the root dbspace, which is referenced by the ROOTPATH instance configuration parameter.

However, in Informix, to avoid possible disk I/O contention and bottlenecks, create all user databases in a separate dbspace using the `IN` clause of the `CREATE DATABASE` statement. For example, the following command creates a logged database called `bookdb` in a dbspace called `redspace`:

```
CREATE DATABASE bookdb WITH LOG IN redspace;
```

Like SQL Server, the database storage files/folder associated with the dbspace must exist prior to creating the database.

Just as an SQL Server instance can have more than one user-defined database, an Informix instance can also have more than one user-defined database. However, it is important to note that transaction log files in Informix are associated with the whole instance whereas log files in SQL Server are associated at the database level.

Also, by default, when an Informix instance is created, the log files reside in the root dbspace. Just as SQL Server allows you to change the location of the log files using the `LOG ON` clause of the `CREATE DATABASE` statement, it is advisable to change the location of the log files in Informix to a separate dbspace, which contains no user data. The `onparams` utility allows you to change the location of log files in Informix.

SQL Server, by default, creates the user `dbo` (database owner). When you create a database without specifying an owner, user `dbo` is automatically assigned as the owner of the database. In Informix, the user ID of the user creating the database automatically becomes the owner if no owner is explicitly specified.

4.1.2 Tables

When creating tables, the SQL Server `CREATE TABLE` statement must be converted to Informix, taking advantage of many Informix options in the following areas:

- ▶ Data types
- ▶ Default values
- ▶ Constraints
- ▶ Identity columns
- ▶ Computed columns
- ▶ Storage options
- ▶ Lock mode
- ▶ Logging mode

Data types

There is a well-defined mapping between SQL Server and Informix data types. However, the risk is not in the conversion of the data, but with ensuring that the chosen conversions work well with the code in both the C++ and the stored procedures. This section highlights the important differences between various data types in SQL Server and Informix.

Data type mapping

There are differences between the data types supported in SQL Server and Informix. Notable data type mappings include the following examples:

- ▶ **UNIQUEIDENTIFIER** is a data type unique to SQL Server. It is used to provide a value (with **NEWID()**) that is guaranteed to be unique across all systems on a network. Because there is no equivalent data type within Informix, one suggestion here is to use a user-defined function (UDF) that parallels the **UNIQUEIDENTIFIER**-related functions. To ensure uniqueness, you can create a similar algorithm with the resulting values either stored in a string or in an **int8** data type.
- ▶ Informix allows a maximum of 255 characters in **NVARCHAR** data types. Because SQL Server allows its equivalent data type to be greater than 255 characters in length, you can redefine these column types as **TEXT** or **LVARCHAR**, or as **NCHAR** if multi-byte characters are to be supported.
- ▶ You can convert the SQL Server data types **DATETIME** and **DATETIME YEAR TO FRACTION** to the Informix data types **SMALLDATETIME** and **DATETIME YEAR TO DAY** respectively. However, this conversion requires you to examine the application logic in detail to compensate for the fact that SQL Server and Informix retain entirely separate date formats. Any code that manipulates date information might require logic modifications.
- ▶ You can convert the SQL data type **VARBINARY** to the **BYTE** data type in Informix.
- ▶ You can convert the SQL Server data type **BIT** to either the **SMALLINT** or **BOOLEAN** data types in Informix.
- ▶ You can convert the SQL Server data type **BIGINT** to **INT8** in Informix.
- ▶ The SQL Server data type **BINARY** can become a **BYTE** column in Informix.

Most of the data types that are supported in SQL Server can be mapped to equivalent Informix data types. Refer to Appendix B, “Data types” on page 409 for a complete data type mapping.

Strong type casting

When comparing, concatenating, assigning, or returning database expressions, columns, or variables of separate data types, you must convert the value of one data type to the other data type before the operation can be performed.

SQL Server can implicitly cast certain expressions or values to another data type. For example, it has a built-in cast that can implicitly cast DATE data types to VARCHAR. You can also create explicit casts using one of two functions: CAST or CONVERT. For example, the following statements explicitly cast a DATETIME value to a DECIMAL:

```
DECLARE @DECVAL DECIMAL;  
DECLARE @DATETIMEVAL DATETIME;  
SET @DATETIMEVAL = '2010-03-01T10:10:10.100';  
SET @DECVAL = CAST(@DATETIMEVAL AS DECIMAL);
```

Executing the previous statement without the CAST clause causes the last SET statement to fail.

Informix also provides strong type casting to avoid user mistakes during the assignment or comparison of separate types involving actual data.

For certain data types, you can choose whether Informix or the user controls casting. You can create an implicit cast if you want Informix to automatically convert the source data type. You can create an explicit cast if you want the user to specify the cast within an SQL statement.

Implicit casts allow you to convert a user-defined data type to a built-in type and vice versa. Informix automatically invokes a single implicit cast when needed to evaluate and compare expressions or pass arguments. Operations that require more than one implicit cast will fail. For example, you can compare a value of type INT with SMALLINT, FLOAT, or CHAR values without explicitly casting the expression, because Informix provides system-defined casts to transparently handle conversions between these built-in data types.

You can also use the IMPLICIT keyword of the CREATE CAST statement to create your own (user-defined) implicit cast. The following CREATE CAST statement creates an implicit cast from the percent data type to the DECIMAL data type:

```
CREATE IMPLICIT CAST (percent AS DECIMAL)
```

Explicit casts, unlike implicit casts or built-in casts, are never invoked automatically by Informix. Users must invoke them explicitly with the CAST AS keywords or with the double colon (::) cast operator. The SQL Server CAST function must therefore be replaced with the Informix “::” operator.

To create an explicit cast, specify the EXPLICIT keyword of the CREATE CAST statement. If you omit the keyword, the default is an explicit cast. Both of the following CREATE CAST statements create explicit casts from the percent data type to the INTEGER data type:

```
CREATE EXPLICIT CAST (percent AS INTEGER)
CREATE CAST (percent AS INTEGER)
```

Informix allows you to invoke an explicit cast using two supported methods. You can explicitly use the keyword CAST or use the cast operator ::, as shown in the following example:

```
... WHERE col1 > (CAST percent AS INTEGER)
... WHERE col1 > (percent::INTEGER)
```

You can also create implicit and explicit cast functions to cast types with dissimilar data structures. For example, suppose you want to compare values of two opaque data types: int_type and float_type. Both types have an external LVARCHAR format that you can use as an intermediate type for converting from one to the other.

You can use the CREATE FUNCTION statement that is referenced in Example 4-1 to create and register an SPL function. The function, int_to_float(), casts the input value of type int_type to a LVARCHAR data type, and then casts the LVARCHAR result to a float_type data type. It then returns the converted value, now of the float_type data type, to the statement that called the function.

Example 4-1 CREATE FUNCTION statement to create a cast function

```
CREATE FUNCTION int_to_float(int_arg int_type)
  RETURNS float_type
  RETURN CAST(CAST(int_arg AS LVARCHAR) AS float_type);
END FUNCTION;
```

After you create this cast function, you can use the CREATE CAST statement to register the function as a cast. You cannot use the function as a cast until you register it with the CREATE CAST statement. The following CREATE CAST statement creates an explicit cast that uses the int_to_float() function as its cast function:

```
CREATE EXPLICIT CAST (int_type AS float_type
  WITH int_to_float)
```

Differences in implicit type conversions between SQL Server and Informix can sometimes result in complications in the translation of DML statements. In these situations, you can use a built-in or user-defined conversion function anywhere that you use an expression.

The following examples are built-in conversion functions:

- The DATE function converts a character string to a DATE value. In the following query, the DATE function converts a character string to a DATE value to allow for comparisons with DATETIME values:

```
SELECT book_id, title, release_date
FROM booktab
WHERE release_date > DATE ('12/31/97');
```

The query retrieves DATETIME values only when release_date is later than the specified DATE. The release_date is defined as a DATETIME data type.

- The TO_CHAR function converts DATETIME or DATE values to character string values. The TO_CHAR function evaluates a DATETIME value according to the date-formatting directive that you specify and returns an NVARCHAR value. You can also use the TO_CHAR function to convert a DATETIME or DATE value to an LVARCHAR value. The following query uses the TO_CHAR function to convert a DATETIME value to a more readable character string:

```
SELECT book_id,
       TO_CHAR(release_date, "%A %B %d %Y") rel_date
FROM booktab
WHERE brand_id = 101;
```

- The TO_DATE function accepts an argument of a character data type and converts this value to a DATETIME value. The TO_DATE function evaluates a character string according to the date-formatting directive that you specify and returns a DATETIME value. You can also use the TO_DATE function to convert an LVARCHAR value to a DATETIME value. The following query uses the TO_DATE function to convert character string values to DATETIME values whose format you specify:

```
SELECT book_id, title
FROM booktab
WHERE release_date = TO_DATE("1998-07-07 10:24", "%Y-%m-%d");
```

Default values

In SQL Server, you can assign default values to columns of a table by specifying the DEFAULT keyword followed by the value. For example, Example 4-2 on page 76 creates a table called book_history with two columns that have specified defaults. Any records inserted into this table for which the values for the columns location and timestamp are not specified will be assigned their respective default values.

Example 4-2 CREATE TABLE statement with defaults

```
CREATE TABLE book_history
(
    book_id      INT          NOT NULL,
    book_no      CHAR(10)     NOT NULL,
    title        VARCHAR(80)  NOT NULL,
    brand_id     INT          NOT NULL,
    price        MONEY        ,
    notes        VARCHAR(200) ,
    location     VARCHAR(30)   DEFAULT ('N San Jose'),
    timestamp    DATE         NOT NULL
                                DEFAULT (getdate())
)
```

In Informix, you can also use the **DEFAULT** clause to specify the default value for Informix to insert into a column when no explicit value for the column is specified. However, you cannot specify default values for **SERIAL**, **BIGSERIAL**, or **SERIAL8** columns.

Note that the use of parentheses after the **DEFAULT** keyword is optional in SQL Server; however, in Informix, the use of parentheses is not allowed. For example, the following column definition is valid in Informix:

```
location      VARCHAR(30)   DEFAULT 'N San Jose'
```

If you do not specify a default value for a column, the default is **NULL** unless you place a **NOT NULL** constraint on the column. In this case, no default exists. If you specify **NULL** as the default value for a column, you cannot specify a **NOT NULL** constraint as part of the column definition. **NULL** is not a valid default value for a column that is part of a primary key. If the column is a **BYTE**, **TEXT**, **BLOB** (binary large object), or **CLOB** (character large object) data type, **NULL** is the only valid default value.

You can also designate a literal value as a default value in Informix. A literal value is a string of alphabetic or numeric characters.

Like SQL Server, you can specify a built-in function as the default column value in Informix. However, as Table 4-1 on page 77 shows, you are limited to certain built-in functions only.

Table 4-1 Allowed built-in functions as a DEFAULT column value in Informix

Built-in function	Data type requirement	Recommended size (bytes)
CURRENT, SYSDATE	DATETIME column with matching qualifier	Enough bytes to store the longest DATETIME value for the locale
DBSERVERNAME, SITENAME	CHAR, VARCHAR, NCHAR, NVARCHAR, or CHARACTER VARYING column	128 bytes
TODAY	DATE column	Enough bytes to store the longest DATE value for the locale
USER	CHAR, VARCHAR, NCHAR, NVARCHAR, or CHARACTER VARYING column	32 bytes

Using one of the built-in functions in Table 4-1, you can create the CREATE TABLE statement in our example in Informix, as shown in Example 4-3.

Example 4-3 CREATE TABLE statement with a default value and function

```
CREATE TABLE book_history2
(
  book_id      INT          NOT NULL,
  book_no      CHAR(10)     NOT NULL,
  title        VARCHAR(80)  NOT NULL,
  brand_id     INT          NOT NULL,
  price        MONEY        ,
  notes        VARCHAR(200) ,
  location     VARCHAR(30)   DEFAULT 'N San Jose',
  timestamp    DATE         DEFAULT today
)
```

Constraints

The SQL Server syntax for constraints must be changed to the Informix format for primary keys, foreign keys, unique, and others.

Primary key constraint

A *primary key* is a column (or a set of columns, if you use the multiple-column constraint format) that contains a non-NULL, unique value for each row in a table.

When you define a PRIMARY KEY constraint, Informix automatically creates an *internal* index on the column or columns that make up the primary key.

For efficiency, the primary key must be one of the following types:

- ▶ Numeric (INT or SMALLINT)
- ▶ Serial (BIGSERIAL, SERIAL, or SERIAL8)
- ▶ A short character string (as used for codes)

You can designate only one primary key for a table. If you define a single column as the primary key, it is unique by definition. You cannot explicitly give the same column a unique constraint.

Example 4-4 lists the difference in syntax between SQL Server and Informix when defining a primary key.

Example 4-4 Primary key definition

```
-- SQL Server Syntax:
CONSTRAINT name PRIMARY KEY(column(s))

-- SQL Server Example:
CONSTRAINT pk_acct_id PRIMARY KEY(acct_id)

-- Informix Syntax:
PRIMARY KEY (column(s)) CONSTRAINT name

-- Informix Example:
PRIMARY KEY(acct_id) CONSTRAINT pk_acct_id
```

In Informix, you cannot place a unique or primary key constraint on a BLOB or CLOB column. Also, null values are never allowed in primary key columns.

Foreign key constraint

A *foreign key* joins and establishes dependencies between tables. That is, it creates a referential constraint. A foreign key is a column or group of columns in one table that contains values that match the primary key in another table. A foreign key references a unique or primary key in a table. You use foreign keys to join tables.

Example 4-5 on page 79 lists the difference in syntax between SQL Server and Informix when defining a foreign key.

Example 4-5 Foreign key definition

```
-- SQL Server Syntax:
CONSTRAINT name FOREIGN KEY REFERENCES schema.table_name(column)
[ ON DELETE { NO ACTION | SET NULL | SET DEFAULT | CASCADE } ]
[ ON UPDATE { NO ACTION | SET NULL | SET DEFAULT | CASCADE } ]

-- SQL Server Example:
CONSTRAINT fk_acc_dept_code FOREIGN KEY
REFERENCES department ( dept_code ) ON DELETE CASCADE

--Informix Syntax:
FOREIGN KEY(column(s)) references_clause
[ON DELETE CASCADE] CONSTRAINT name

--Informix Example:
FOREIGN KEY ( dept_code ) REFERENCES department ( dept_code )
ON DELETE CASCADE CONSTRAINT fk_acc_dept_code
```

In Informix, you cannot specify BYTE, TEXT, BLOB, and CLOB columns as foreign keys.

Important: There is no ON DELETE SET NULL in Informix. However, you can write a stored procedure or a trigger to get that capability. Refer to the CREATE TABLE statement syntax in the *IBM Informix Guide to SQL: Syntax*, G229-6375, for information about the Informix referential integrity constraint.

Unique constraint

You can use the UNIQUE or DISTINCT keyword to require that a column or set of columns accepts only unique data values. You cannot insert values that duplicate the values of another row into a column that has a unique constraint. When you create a UNIQUE or DISTINCT constraint, Informix automatically creates an internal index on the constrained column or columns. (In this context, the keyword DISTINCT is a synonym for UNIQUE.)

Example 4-6 lists the difference in syntax between SQL Server and Informix when defining a unique key.

Example 4-6 Unique key definition

```
-- SQL Server Syntax:
CONSTRAINT name UNIQUE(column(s))

-- SQL Server Example:
CONSTRAINT uk_phone UNIQUE(phone)
```

```
-- Informix Syntax:  
UNIQUE (column(s)) CONSTRAINT name
```

```
--Informix Example:  
UNIQUE(phone) CONSTRAINT uk_phone
```

You cannot place a unique constraint on a column that already has a primary key constraint. You cannot place a unique constraint on a BYTE or TEXT column. You cannot place a unique or primary key constraint on a BLOB or CLOB column of Informix.

Check constraint

Check constraints specify a condition or requirement on a data value before data can be assigned to a column during an INSERT or UPDATE statement. If a row evaluates to false for any of the check constraints that are defined on a table during an insert or update, Informix returns an error. However, Informix does not report an error or reject the record when the check constraint evaluates to NULL. For this reason, you might want to use both a check constraint and a NOT NULL constraint when you create a table.

Example 4-7 lists the difference in syntax between SQL Server and Informix when defining a check constraint.

Example 4-7 Check constraint definition

```
--SQL Server Syntax:  
CONSTRAINT name CHECK (condition)
```

```
--SQL Server Example:  
CONSTRAINT chk_ord_month CHECK (order_month BETWEEN 1 AND 12)
```

```
--Informix Syntax:  
CHECK (condition) CONSTRAINT name
```

```
--Informix Example:  
CHECK (order_month BETWEEN 1 AND 12) CONSTRAINT chk_ord_month
```

Check constraints are defined with search conditions. The search condition cannot contain subqueries, aggregates, host variables, or SPL routines. In addition, it cannot include the built-in functions: CURRENT, USER, SITENAME, DBSERVERNAME, or TODAY. When you define a check constraint at the column level, the only column that the check constraint can check against is the column

itself. In other words, the check constraint cannot depend upon values in other columns of the table.

Important: An SQL Server check constraint statement can contain functions that are not available in Informix. To get similar functionality, you can create stored procedures in Informix.

NOT NULL constraint

You can use the NOT NULL keywords to require that a column receives a value during insert or update operations. If you place a NOT NULL constraint on a column (and no default value is specified), you must enter a value into this column when you insert a row or update that column in a row. If you do not enter a value, Informix returns an error, because no default value exists.

One difference between SQL Server and Informix is in the placement of the NOT NULL clause when the column is defined. SQL Server allows the NOT NULL clause on either side of another constraint. However, Informix requires the NOT NULL clause *before* any constraint definition. Example 4-8 lists this difference.

Example 4-8 NOT NULL constraints

```
-- SQL Server Example:
... brand_id          INT          PRIMARY KEY NOT NULL,

--Informix Example:
... brand_id          INT          NOT NULL PRIMARY KEY,
```

IDENTITY columns

SQL Server supports the IDENTITY column feature, which allows you to create a column with system-generated values that uniquely identify each row in a table. While not a data type (rather, it is a column property), IDENTITY can be handled in Informix using the SERIAL, SERIAL8, and BIGSERIAL data types.

Whenever a new row is inserted into a table, Informix automatically generates a new value for BIGSERIAL, SERIAL, or SERIAL8 columns. When rows are deleted from the table, their serial numbers are not reused. Rows that are sorted on columns of these types are returned in the order in which they were created.

A table can have no more than one SERIAL column, but it can have a SERIAL column and either a SERIAL8 column or a BIGSERIAL column. Because Informix generates the values, the serial values in new rows always differ, even when multiple users add rows at the same time.

A BIGSERIAL, SERIAL, or SERIAL8 column is not automatically a unique column. If you want to be perfectly sure that no duplicate serial numbers occur, you must apply a unique constraint.

The BIGSERIAL, SERIAL, and SERIAL8 data types have the following advantages:

- ▶ They provide a convenient way to generate system-assigned keys.
- ▶ They produce unique numeric codes even when multiple users are updating the table.
- ▶ Separate tables can use separate ranges of numbers.

The SERIAL data type can yield up to $2^{31}-1$ positive integers.

Computed columns

SQL Server allows you to create virtual or physical columns whose values are derived from an expression using other columns in the same table, functions, constants, and variables. Computed columns are virtual unless the keyword PERSISTED is used.

SQL Server allows you to use computed columns in SQL statements; however, you cannot INSERT or UPDATE a computed column. Consider the SQL Server example in Example 4-9, which creates a table with a computed column called total_price.

Example 4-9 Computed column in SQL Server

```
CREATE TABLE books_order
( cust_id INT,
  quantity INT,
  price MONEY,
  total_price AS (quantity * price) PERSISTED
);
```

Although Informix does not support computed columns, you can use triggers to implement the same functionality. For example, in Informix, you first create the table that is shown in Example 4-10.

Example 4-10 CREATE TABLE in Informix

```
CREATE TABLE books_order
( cust_id INT,
  quantity INT,
  price MONEY,
  total_price MONEY
);
```

In Example 4-10 on page 82, notice that the column `total_price` is now defined as `MONEY`.

For `INSERT` statements, you can automatically update the `total_price` using an Informix stored procedure and trigger, as described in Example 4-11.

Example 4-11 Informix stored procedure and trigger create computed values for INSERT

```
CREATE PROCEDURE ins_tot_rb_ord()
    REFERENCING OLD AS pre NEW AS post FOR books_order;

LET post.total_price = post.quantity * post.price;

END PROCEDURE;

CREATE TRIGGER insert_books_order
INSERT ON books_order
FOR EACH ROW (
EXECUTE PROCEDURE ins_tot_rb_ord() WITH TRIGGER REFERENCES);
```

For `UPDATE` statements, you can automatically update the `total_price` using an Informix stored procedure and two triggers, as demonstrated in Example 4-12.

Example 4-12 Informix stored procedure and triggers create computed values for UPDATE

```
CREATE PROCEDURE upd_tot_rb_ord(val int)
    REFERENCING OLD as pre NEW as post for books_order;

    IF (val = 1)
    THEN
        UPDATE books_order
        SET total_price = post.quantity * pre.price
        WHERE cust_id = pre.cust_id;
    ELSE
        UPDATE books_order
        SET total_price = pre.quantity * post.price
        WHERE cust_id = pre.cust_id;
    END IF

END PROCEDURE;

CREATE TRIGGER update_books_order1
UPDATE OF quantity ON books_order
FOR EACH ROW (
EXECUTE PROCEDURE upd_tot_rb_ord(1) WITH TRIGGER REFERENCES);

CREATE TRIGGER update_books_order2
UPDATE OF price ON books_order
```

```
FOR EACH ROW (  
EXECUTE PROCEDURE upd_tot_rb_ord(2) WITH TRIGGER REFERENCES);
```

Informix stored procedures are a great way to embed and encapsulate business logic.

Storage options

As with file groups in SQL Server, before a dbspace can be created in Informix, the associated chunks must physically exist.

If all the data is stored in one file group for a table in SQL Server, converting the DDL to Informix is simply a matter of replacing the *ON filegroup* clause with the *IN dbspace* clause in Informix. Example 4-13 stores all data for the `brands` table in the `brands_storage_space` storage space.

Example 4-13 CREATE TABLE using a storage clause

```
-- SQL Server Example:  
CREATE TABLE brands  
(  
    brand_id      INT PRIMARY KEY,  
    brand_name    VARCHAR(20) NOT NULL,  
    brand_desc    VARCHAR(80) NOT NULL  
)  
ON brands_storage_space;  
  
--Informix Example:  
CREATE TABLE brands  
(  
    brand_id      INT PRIMARY KEY,  
    brand_name    VARCHAR(20) NOT NULL,  
    brand_desc    VARCHAR(80) NOT NULL  
)  
IN brands_storage_space;
```

In SQL Server, partitioning a table requires a partition function and scheme. A *partition function* identifies how to split the table data. A *partition scheme* identifies over which file groups the split data is to be stored. Example 4-14 details how to create a partitioned table in SQL Server.

Example 4-14 Create a partitioned table in SQL Server

```
CREATE PARTITION FUNCTION books_function (int)  
AS RANGE LEFT FOR VALUES (25000, 50000);  
  
CREATE PARTITION SCHEME books_scheme  
AS PARTITION books_function
```



```
TO (file_grp1, file_grp2, file_grp3);
```

```
CREATE TABLE book_history
(
    book_id      INT          NOT NULL,
    book_no      CHAR(10)     NOT NULL,
    title        VARCHAR(80)  NOT NULL,
    brand_id     INT          NOT NULL,
    price        money        NULL,
    notes        VARCHAR(200) NULL,
    release_date DATETIME     NOT NULL
                        DEFAULT (getdate()),
    change_date  DATETIME,
    timestamp    DATETIME     NOT NULL
                        DEFAULT (getdate())
)
ON books_scheme (book_id);
```

In Example 4-14 on page 84, the values for `book_id` are stored in one of the following locations:

- ▶ Less than or equal to 25000 is stored in the `file_grp1` file group.
- ▶ Greater than 25000 and less than or equal to 50000 is stored in `file_grp2` file group.
- ▶ Greater than 50000 is stored in `file_grp3` file group.

Informix has a simpler method to partition data, which does not require partition functions or partition schemes. Informix replaces the `ON` clause with the `FRAGMENT` clause, as detailed in Example 4-15.

Example 4-15 Create a fragmented table in Informix

```
CREATE TABLE book_history
(
    book_id      INT          NOT NULL,
    book_no      CHAR(10)     NOT NULL,
    title        VARCHAR(80)  NOT NULL,
    brand_id     INT          NOT NULL,
    price        money        NULL,
    notes        VARCHAR(200) NULL,
    release_date DATETIME     NOT NULL
                        DEFAULT (getdate()),
    change_date  DATETIME,
    timestamp    DATETIME     NOT NULL
                        DEFAULT (getdate())
)
FRAGMENT BY EXPRESSION
(book_id <= 25000) IN dbspace1,
```

```
(book_id > 25000 AND book_id <= 50000) IN dbspace2,  
REMAINDER in dbspace3;
```

Informix also has the round-robin distribution scheme, in which you specify at least two dbspaces where you want the fragments to be placed. As records are inserted in the table, they are placed in the first available fragment. Informix balances the load among the specified fragments as you insert records and distributes the rows in such a way that the fragments always maintain approximately the same number of rows.

You can also create Informix tables with initial and subsequent (*next*) storage values. When you create a table, Informix allocates a fixed amount of space to contain the data to be stored in that table. When this space fills, Informix must allocate space for additional storage. The physical unit of storage that Informix uses to allocate both the initial and subsequent storage space is called an *extent*.

An extent consists of a collection of contiguous pages. These contiguous pages store data for a given table. Every permanent database table has two extent sizes associated with it. The *initial-extent size* is the number of KBs allocated to the table when it is first created, specified by the `EXTENT` clause of the `CREATE TABLE` statement. The *next-extent size* is the number of KBs allocated to the table when the initial extent (and any subsequent extents) becomes full, specified by the `NEXT` clause of the `CREATE TABLE` statement.

For permanent tables and user-defined temporary tables, the next-extent size begins to double after 16 extents have been added. For system-created temporary tables, the next-extent size begins to double after four extents have been added. The default Informix `EXTENT` and `NEXT` sizes are four times the disk page size on your system. For example, if you have a 4 KB page system, the minimum length is 16 KB.

Logging mode

You cannot create logged and non-logged tables in SQL Server, because logging is defined at the database level.

You can create logging or non-logging tables in a logged database on Informix. Informix supports two table types:

- ▶ `STANDARD` (logging tables)
- ▶ `RAW` (non-logging tables)

The default standard table is like a table created in earlier versions without a special keyword specified. You can create either a `STANDARD` or `RAW` table and change tables from one type to another.

In a nonlogging database, both STANDARD tables and RAW tables are nonlogging. In a nonlogging database, the only difference between STANDARD and RAW tables is that RAW tables do not support primary constraints, unique constraints, and rollback. However, these tables can be indexed and updated.

A STANDARD table is the same as a table in a logged database that Informix creates. All operations are logged, record by record, so STANDARD tables can be recovered and rolled back. You can back up and restore STANDARD tables. Logging enables updates since the last physical backup to be applied when you perform a warm restore or point in time restore. A STANDARD table is the default type on both logging and non-logging databases.

RAW tables are intended for the initial loading and validation of data. To load RAW tables, you can use any loading utility, including dbexport or the *High-Performance Loader* in *express* mode. If an error or failure occurs while loading a RAW table, the resulting data is whatever was on the disk at the time of the failure. Update, insert, and delete operations on rows in a RAW table are supported but are not logged. If you update a RAW table, you cannot reliably restore the data unless you perform a full (level-0) backup after the update.

We do not suggest using RAW tables within a transaction. After you have loaded the data, use the ALTER TABLE statement to change the table to the STANDARD type and perform a level-0 backup before you use the table in a transaction.

Temporary tables

Informix and SQL Server both support temporary tables. Like Informix, SQL Server implements temporary tables just as regular database tables.

Temporary tables in SQL Server are stored in the tempdb database, which is re-created every time that the server is restarted. Two types of temporary tables are supported: private and global. *Private temporary tables* are visible only in the current session (connection) and are denoted with a single number sign (#) preceding the table name. *Global temporary tables* are visible in all sessions and are denoted with a double number sign (##) preceding the table name.

Informix temporary tables are only visible to the user that created them. However, Informix allows you to create one or more *user-defined* temporary dbspaces to store temporary tables.

Informix temporary tables can be created, manipulated, and dropped only within a session. After the session ends, any remaining temporary tables created by that session are automatically dropped. Temporary tables support indexes, constraints, and rollback. You cannot recover, back up, or restore temporary tables. Temporary tables support bulk operations, such as light appends, which

add rows quickly to the end of each table fragment. Example 4-16 creates a local (private) temporary table in SQL Server and Informix.

Example 4-16 Create a temporary table

```
-- SQL Server Example:
CREATE TABLE #brands
(
    brand_id      INT PRIMARY KEY,
    brand_name    VARCHAR(20)  NOT NULL,
    brand_desc    VARCHAR(80)  NOT NULL
)
-- Informix Example:
CREATE TABLE brands
(
    brand_id      INT PRIMARY KEY,
    brand_name    VARCHAR(20)  NOT NULL,
    brand_desc    VARCHAR(80)  NOT NULL
) WITH NO LOG
```

Using the WITH NO LOG option

When creating temporary tables in Informix, use the WITH NO LOG option to reduce the overhead of transaction logging. If you specify WITH NO LOG, operations on the temporary table are not included in the transaction-log operations. The WITH NO LOG option is required on all temporary tables that you create in temporary dbspaces. In Informix, you can also choose to create a temporary table in the dbspace of your choice.

Differences between temporary and permanent tables

Permanent tables differ from temporary tables in the following ways:

- ▶ They have fewer types of constraints available.
- ▶ They have fewer options that you can specify.
- ▶ They are not visible to other users or sessions.
- ▶ They do not appear in the system catalog tables.
- ▶ They are not preserved.

Duration of temporary tables in Informix

The duration of a temporary table depends on whether it is logged. A logged temporary table exists until one of the following situations occurs:

- ▶ The application disconnects.
- ▶ A DROP TABLE statement is issued on the temporary table.
- ▶ The database is closed.

A non-logging temporary table exists until one of the following events occurs:

- ▶ The application disconnects.
- ▶ A DROP TABLE statement is issued on the temporary table.

External tables

An *external table* is an operating system data file that is not managed by Informix. You can map external data to internal data so that Informix views the external data as an external table. Treating the external data as a table provides a powerful method for moving data into or out of the database and for specifying transformations of the data. SQL Server does not have functionality similar to the Informix external table.

You can use external tables to load and unload database data. You issue a series of SQL statements that perform the following functions:

- ▶ Transfer operational data efficiently to or from other systems
- ▶ Transfer data files across platforms in Informix internal data format
- ▶ Use Informix to convert data between delimited ASCII, fixed-ASCII, and Informix internal (raw) representation
- ▶ Use SQL INSERT and SELECT statements to specify the mapping of data to new columns in a database table
- ▶ Provide parallel standard INSERT operations so that data can be loaded without dropping indexes
- ▶ Use named pipes to support loading data to and unloading data from storage devices, including tape drives and direct network connections
- ▶ Maintain a record of load and unload statistics during the run
- ▶ Perform express (high-speed) and deluxe (data-checking) transfers

You can issue the SQL statements with DB-Access or embed them in an ESQL/C program.

To define an external table, you use SQL statements to describe the data file, define the table, and then specify the data to load or unload.

To set up loading and unloading tasks, you issue a series of SQL statements:

- ▶ CREATE EXTERNAL TABLE to describe the data file to load or unload.
- ▶ CREATE TABLE to define the table to load.
- ▶ INSERT...SELECT to load and unload.

Example 4-17, which performs the following tasks, is an example of the CREATE EXTERNAL TABLE syntax:

- ▶ An external table named `ext_brands` is created with three columns. The `DATAFILES` clause indicates the location of the data file, specifies that data in the file is delimited, indicates the location of the reject file, indicates that the reject file can contain no more than 50 errors, and specifies that data is to be loaded using `DELUXE` mode.
- ▶ A permanent Informix table called `brands` is then created.
- ▶ Data is loaded from the `ext_brands` external table into the `brands` Informix table.

Example 4-17 Create an external table

```
CREATE EXTERNAL TABLE ext_brands
(
    brand_id      INT          PRIMARY KEY,
    brand_name    VARCHAR(20)  NOT NULL,
    brand_desc    VARCHAR(80)  NOT NULL
)
USING
(DATAFILES
    (
        "DISK:/work/branddata.unl"
    ),
    FORMAT "DELIMITED",
    REJECTFILE "/usr/home/informix/branddata.rej",
    MAXERRORS 50,
    DELUXE
);

CREATE EXTERNAL TABLE brands
(
    brand_id      int          PRIMARY KEY,
    brand_name    VARCHAR(20)  NOT NULL,
    brand_desc    VARCHAR(80)  NOT NULL
) FRAGMENT BY ROUND ROBIN IN dbspace1, dbspace2, dbspace3;

INSERT INTO brands SELECT * FROM ext_brands;
```

Table 4-2 on page 91 provides a comparison of SQL Server and Informix table types.

Table 4-2 Mapping SQL Server and Informix tables

SQL Server	Informix	Observation
System	Catalog	Internal tables
Temporary	Temporary	N/A
Wide	None	Same as a regular table
Partitioned	Standard/Fragmented	N/A
N/A	External Table	Great for Migrations

4.1.3 Indexes

Both SQL Server and Informix use the CREATE INDEX statement to create an index on one or more columns in a table and, optionally, to cluster the physical table in the order of the index. In Informix, this statement is an extension to the ANSI/ISO standard for SQL.

Composite indexes

A *simple index* lists only one column (or for Informix, only one column or function) in its Index Key Specification. Any other index is a composite index. List the columns in a composite index in order from the most frequently used column to the least frequently used column.

The following example creates a composite index using the stock_num and manu_code columns of the stock table:

```
CREATE UNIQUE INDEX st_man_ix ON stock (stock_num, manu_code)
```

The UNIQUE keyword prevents any duplicates of a given combination of stock_num and manu_code. The index is in ascending order, by default.

You can include up to 16 columns in a composite index. The total width of all indexed columns in a single composite index cannot exceed 380 bytes.

In Informix, an index key part is either a column in a table, or the result of a user-defined function on one or more columns. A composite index can have up to 16 key parts that are columns, or up to 341 key parts that are values returned by a user-defined routine (UDR). This limit is language dependent and applies to UDRs that are written in SPL or Java; functional indexes based on C language UDRs can have up to 102 key parts.

A composite index can have any of the following items as an index key:

- One or more columns

- One or more values that a user-defined function returns (referred to as a *functional index*)
- A combination of columns and user-defined functions

Functional indexes

Functional indexes are UDRs that accept column names as arguments, and whose return values are specified as index keys in the `CREATE INDEX` statement. You can build a functional index on a user-defined function. The user-defined function can be either an external function or an SPL function.

Follow these steps to build a functional index on a user-defined function:

1. Write the code for the user-defined function if it is an external function, for example, the `darkness()` function that operates on the data type and returns a decimal value.
2. Register the user-defined function in the database with the `CREATE FUNCTION` statement, for example:

```
CREATE FUNCTION darkness(im image)
  RETURNS decimal
  EXTERNAL NAME '/lib/image.so'
  LANGUAGE C NOT VARIANT
```

In this example, you can use the default operator class for the functional index, because the return value of the `darkness()` function is a built-in data type - `DECIMAL`.

3. Build the functional index with the `CREATE INDEX` statement, for example:

```
CREATE TABLE photos
(
  name CHAR(20),
  picture image
  ...
);
CREATE INDEX dark_ix ON photos (darkness(picture));
```

In this example, assume that the user-defined data type of `image` has already been defined in the database.

The Informix optimizer can now consider the functional index when you specify the `darkness()` function as a filter in the following query:

```
SELECT count(*) FROM photos WHERE
darkness(picture) > 0.5
```

You can also create a composite index with user-defined functions. However, do not create a functional index using either the `DECRYPT_BINARY()` or

DECRYPT_CHAR() function. These functions store plain text data in the database, defeating the purpose of encryption.

Maximum key size

The total widths of all indexed columns in a single CREATE INDEX statement have a limitation based upon the page size of the dbspace in which the index resides. However, this limitation does not apply to functional indexes.

An enhancement called *configurable page sizes for dbspaces* was introduced with IBM Informix Version 10.00. Prior to this version, the page size of dbspaces was restricted to the page size of the operating system and was unchangeable; thus, Informix was limited to 390 byte indexes. With page sizes now configurable from 2 K to 16 K, wider indexes are supported.

Referring to Table 4-3, 16 K page sizes raise the limit on index width to over 3000 bytes. Lifting the 390 byte limit also permits LVARCHAR columns larger than 387 bytes to be included in an index definition.

Table 4-3 *Page sizes for dbspaces*

Page size	Maximum key size
2 KBs	387 bytes
4 KBs	796 bytes
8 KBs	1615 bytes
12 KBs	2435 bytes
16 KBs	3245 bytes

In addition to aiding the porting process, wider indexes have satisfied requirements for Unicode data, and have provided performance gains by reducing B-Tree index traversal costs. Now, indexing can be built with more items on an index page and produces fewer levels.

Index fragmentation

Both SQL Server and Informix support index fragmentation (partitioning). SQL Server indexes can be partitioned similarly to table partition schemes or can differ from the table partition strategy. This method is similar to Informix.

In SQL Server, if the partitioning scheme is not specified in the CREATE INDEX statement, but the table is partitioned, the index is partitioned using the partitioning scheme of the table. In Informix, which has similar functionality, these types of indexes are known as *attached indexes*. Informix fragments the attached index according to the same distribution scheme as the table by using the same

rule for index keys as for table data. As a result, attached indexes have the following physical characteristics:

- ▶ The number of index fragments is the same as the number of data fragments.
- ▶ Each attached index fragment resides in the same dbspace as the corresponding table data, but in a separate *tblspace*.
- ▶ An attached index or an index on a non-fragmented table uses 4 bytes for the row pointer for each index entry.

To create an index that follows the same strategy that is used when creating the fragmented table, create an index with a simple CREATE INDEX statement. Example 4-18 shows how to create an attached index in Informix. For additional information, refer to the *IBM Informix Dynamic Server Performance Guide, v11.10*, G229-6385-01.

Example 4-18 Informix attached index

```
CREATE INDEX idx_books_hist_local ON book_history (book_id);
```

In SQL Server, to create an index using a partitioning scheme other than that of the table, you have to first create a new partitioning scheme, based on an existing or new partition function, and then reference the scheme in the CREATE INDEX statement. Example 4-19 shows how to create an SQL Server index partition based on the SQL Server partition table called *book_history* that was created in “Storage options” on page 84.

Example 4-19 SQL Server partition scheme

```
CREATE INDEX book_history_part_index  
ON book_history (book_id)  
ON books_scheme (book_id);
```

Informix refers to the previous types of indexes as *detached fragmented indexes*. Informix detached indexes can also refer to indexes that do not follow the same fragmentation scheme as the table that they index. Example 4-20 shows how to create an Informix fragmented index based on the Informix fragmented table called *book_history* that was created in “Storage options” on page 84.

Example 4-20 Informix detached index

```
CREATE INDEX book_history_frag_index ON book_history (book_id)  
FRAGMENT BY EXPRESSION  
(book_id < 10000) IN dbspace1,  
(book_id > 10000 AND book_id <= 25000) IN dbspace2,  
(book_id > 25000 AND book_id <= 50000) IN dbspace3,  
REMAINDER IN dbspace4;
```

Clustered indexes

Both SQL Server and Informix offer the option to order records within a table according to an index. By default, both products create a non-clustered index if the keyword CLUSTERED (in SQL Server) or CLUSTER (in Informix) is not explicitly specified in the CREATE INDEX statement.

Use the Informix CLUSTER option when appropriate. This option orders the data within the table in the order that the index requires. It is important to note that the CLUSTER index is not maintained over time. The primary factor determining the frequency of CLUSTER index rebuilds is the amount of DML performed against the table. Frequent INSERTS, UPDATES, and DELETES make it necessary to closely monitor CLUSTER indexes. Only one clustered index exists per table, because table data can only be stored in one order. The CREATE CLUSTER INDEX statement fails if a CLUSTER index already exists on the same table.

The following example shows the use of the CLUSTER keyword in Informix to reorder the rows of the booktab table in the order of the book_no column and *not* of its primary key column book_id:

```
CREATE CLUSTER INDEX booktab_ci ON booktab (book_no);
```

The TO NOT CLUSTER option in Informix drops the cluster attribute on the index name without affecting the physical table. Because no more than one clustered index can exist on a given table, you must use the TO NOT CLUSTER option to release the cluster attribute from one index before you assign it to another index on the same table.

Automatic calculation of distribution statistics

Like SQL SERVER, Informix allows you to disable an existing index using the DISABLED option; SQL Server uses the keyword DISABLE. The index definition remains in the catalog table. Informix does not update the index after insert, delete, and update operations that modify the base table. The optimizer does not use the index during the execution of queries. However, to re-enable the index, SQL Server rebuilds the disabled index with the REBUILD option whereas Informix simply updates the index with the ENABLED option.

When the CREATE INDEX statement runs successfully, with or without the ONLINE keyword, Informix automatically gathers statistics for the newly created index. It uses the statistics to update the sysdistrib system catalog table with values that are equivalent to an UPDATE STATISTICS operation in a mode that depends on the type of index:

- Index level statistics, which are equivalent to the statistics gathered by UPDATE STATISTICS in the LOW mode, are calculated for most types of indexes, including B-tree, Virtual Index Interface, and functional indexes.

- ▶ Column distribution statistics, which are equivalent to the distribution generated in the HIGH mode, for a non-opaque leading indexed column of an ordinary B-tree index. The resolution percentage is 1.0 if the table has fewer than a million rows, and 0.5 for larger table sizes.

These distribution statistics are available to the query optimizer when it designs query plans for the table on which the new index was created. For composite key indexes, only distributions of the leading column are created implicitly by the CREATE INDEX statement.

ONLINE keyword

By default, in both SQL Server and Informix, the CREATE INDEX statement attempts to place an exclusive lock on the indexed table to prevent all other users from accessing the table while the index is being created. The CREATE INDEX statement fails if another user already has a lock on the table, or if another user is currently accessing the table at the Dirty Read isolation level.

The DBA can reduce the risk of non-exclusive access errors, and can increase the availability of the indexed table, by including the ONLINE keyword as the last specification of the CREATE INDEX statement. This keyword instructs Informix to create the index online while concurrent users can continue to access the table.

Informix builds the index, even if other users are performing Dirty Read and DML operations on the indexed table. Immediately after you issue the CREATE INDEX . . . ONLINE statement, the new index is not visible to the query optimizer for use in query plans or cost estimates, and Informix does not support any other DDL operations on the indexed table, until after the specified index has been built without errors. At this time, Informix briefly locks the table while updating the system catalog with information about the new index.

The indexed table in a CREATE INDEX . . . ONLINE statement can be permanent or temporary, logged or unlogged, and fragmented or non-fragmented. You cannot specify the ONLINE keyword, however, when you create a functional index, a clustered index, a virtual index, or an R-tree index.

The following statement instructs Informix to create a unique online index called idx_1 on the brand_name column of the brands table:

```
CREATE UNIQUE INDEX idx_1 ON brands (brand_name) ONLINE;
```

Text and XML search indexes

SQL Server also supports full-text and XML indexes. A *full-text index* allows the search of words and phrases in character strings while *XML indexes* allow the search of tags, paths, and values in XML columns. In Informix, both of these functions are included as part of the Basic Search Text (BTS) module.

The Informix Basic Text Search DataBlade module allows you to search words and phrases in a document repository stored in a column of a table. In traditional relational database systems, you must use a LIKE or MATCHES condition to search for text data and use Informix to perform the search. The Basic Text Search DataBlade module uses the open source CLucene text search package. This text search package and its associated functions, which is known as the *text search engine*, is specifically designed to perform fast retrieval and automatic indexing of text data. The text search engine runs in one of the Informix-controlled virtual processes.

The Basic Text Search DataBlade module has two principal components: the `bts_contains()` search predicate function and the Basic Text Search DataBlade module management functions.

When you execute searches with Basic Text Search, you use a predicate called `bts_contains()` that instructs Informix to call the text search engine to perform the search. For example, to search for the string `century` in the `brand_names` column in the `brands` table, you use the following statement:

```
SELECT brand_id FROM brands
WHERE bts_contains(brand_names, 'century')
```

The search predicate takes a variety of arguments to make the search more detailed than a search using a LIKE condition. Search strategies include single and multiple character wildcard searches, fuzzy and proximity searches, AND, OR and NOT Boolean operations, range options, and term-boosting.

You can search for unstructured text or, if you use XML index parameters, you can search columns with XML documents by tags, attributes, or XML paths. You can use Basic Text Search XML index parameters to manipulate searches of XML data in various ways.

When you do not use XML index parameters, XML documents are indexed as unstructured text. The XML tags, attributes, and values are included in searches and are indexed in a single field called `contents`. By contrast when you use XML index parameters, the XML tag and attribute values can be indexed in separate fields either by tag name, attribute name, or by path.

We use a simple example, given the following XML fragment:

```
<skipper>Captain Black</skipper>
```

You can create a BTS index for searching the text within the `<skipper>` `</skipper>` tags:

```
CREATE INDEX boats_bts ON boats(xml_data bts_lvarchar_ops)
USING bts(xmltags="(skipper)") IN bts_sbospace;
```

To search for a skipper's name that contains the word "Black", use the bts search predicate:

```
bts_contains(xml_data, 'skipper:black')
```

General index information

The SQL Server FILLFACTOR functionality is similar to the Informix index FILLFACTOR option of the CREATE INDEX statement or the FILLFACTOR configuration parameter in the ONCONFIG file.

In either case, the functionality is handled by the Informix engine after it is set, and can be removed from the application. FILLFACTOR specifies the degree of index-page compactness. A low value provides room for growth in the index, and a high value compacts the index. If an index is fully compacted (100%), any new inserts result in splitting nodes. The setting on the CREATE INDEX statement overrides the ONCONFIG file value. The FILLFACTOR default value for both the CREATE INDEX statement, as well as the ONCONFIG, is 90.

In SQL Server, you can use the INCLUDE option to add non-key columns to the leaf pages of an unclustered index. The inclusion of additional columns in the leaf pages can speed up query processing, because data for the non-key columns can directly be retrieved from the index, versus going to the data page. Informix does not provide this option; the include columns must be included as part of a composite index.

4.1.4 Views

A *view* is a virtual table that is based on a specified SELECT statement. You use views to restrict the access to the contents of the base tables. A view in Informix has the following characteristics:

- ▶ To create a view, the base table must exist and you must have the SELECT privilege on all columns from which the view is derived.
- ▶ Deletes, inserts, and updates can be performed through the view.
- ▶ Because a view is not really a table, it cannot be indexed and it cannot be the object of statements, such as ALTER TABLE and RENAME TABLE.
- ▶ You cannot rename the columns of a view with RENAME COLUMN. To change anything about the definition of a view, drop the view and recreate it.
- ▶ The view reflects changes to the underlying tables with one exception. If a SELECT * specification defines the view, the view has only the columns that existed in the underlying tables when the view was defined by a CREATE VIEW. Any new columns that are subsequently added to the underlying tables with the ALTER TABLE statement do not appear in the view.

- ▶ The SELECT statement on which a view is based cannot contain INTO TEMP and ORDER BY clauses.

The SQL Server CREATE VIEW statement is similar to the CREATE VIEW statement of Informix. Example 4-21 shows a view creation statement in SQL Server and Informix.

Example 4-21 Create view statement in SQL Server and Informix

```
-- SQL Server Example:
CREATE VIEW v_books
AS
SELECT title, authors.name, brand_name, price, release_date
FROM authors, booktab, brands
WHERE authors.author_id = booktab.author_id
AND booktab.brand_id = brands.brand_id

-- Informix Example:
CREATE VIEW v_books
AS
SELECT title, authors.name, brand_name, price, release_date
FROM authors, booktab, brands
WHERE authors.author_id = booktab.author_id
AND booktab.brand_id = brands.brand_id
```

If a view is built on a single table, you can update that view if the SELECT statement that defines the view does not contain any of the following elements:

- ▶ Columns in the projection list that are aggregate values
- ▶ Columns in the projection list that use the UNIQUE or DISTINCT keyword
- ▶ A GROUP BY clause
- ▶ A UNION operator
- ▶ A query that selects calculated or literal values

You can DELETE from a view that selects calculated values from a single table, but INSERT and UPDATE operations are not valid on such views.

In a view that can be updated, you can update the values in the underlying table by inserting values into the view. If a view is built on a table that has a derived value for a column, however, you cannot update that column through the view. You can update other columns in the view, however.

A view that contains a UNION or UNION ALL operator in its SELECT statement is known as a *union view* in Informix. In SQL Server, this type of view is called a *partitioned view*. Certain restrictions apply to union views in Informix:

- ▶ If a CREATE VIEW statement defines a union view, you cannot specify the WITH CHECK OPTION keywords in the CREATE VIEW statement.

- All restrictions that apply to UNION or UNION ALL operations in stand-alone SELECT statements also apply to UNION and UNION ALL operations in the SELECT statement of a union view.

4.1.5 Schemas

You can use the CREATE SCHEMA statement in Informix and SQL Server to issue a block of data definition language (DDL) and GRANT statements as a unit. The CREATE SCHEMA statement allows the DBA to specify an owner for all database objects that the CREATE SCHEMA statement creates. You cannot issue CREATE SCHEMA until you have created the database that stores the objects.

Users with the *Resource* privilege can create a schema for themselves. In this case, the user must be the name of the person with the Resource privilege who is running the CREATE SCHEMA statement. Anyone with the DBA privilege can also create a schema for someone else. In this case, the user can specify a user other than the person who is running the CREATE SCHEMA statement.

All database objects in Informix that a CREATE SCHEMA statement creates are owned by the *user*, even if you do not explicitly name each database object. If you are the DBA, you can create database objects for another user. If you are not the DBA, specifying an owner other than yourself results in an error message.

You can only grant privileges with the CREATE SCHEMA statement; you cannot use CREATE SCHEMA to revoke or to drop privileges.

If you create a database object or use the GRANT statement outside a CREATE SCHEMA statement, you receive warnings if you use the -ansi flag or set DBANSIWARN.

You can create the schema for an entire database or for a portion of the database.

You can put CREATE and GRANT statements in any logical order. Statements are considered part of the CREATE SCHEMA statement until a semicolon (;) or an end-of-file symbol is reached. Example 4-22 shows the CREATE SCHEMA statement for both SQL Server and Informix.

Example 4-22 Create Schema in SQL Server and Informix

Create Schema Statement in SQL Server and Informix:

```
CREATE SCHEMA AUTHORIZATION sarah
  CREATE TABLE mytable (mytime DATE, mytext TEXT)
  GRANT SELECT, UPDATE, DELETE ON mytable TO rick
```



```
CREATE VIEW myview AS
  SELECT * FROM mytable WHERE mytime > '12/31/2004'
CREATE INDEX idxtime ON mytable (mytime);
```

The options of the Informix dbschema utility allow you to perform the following actions:

- ▶ Display CREATE SYNONYM statements by owner, for a specific table or for the entire database.
- ▶ Display the CREATE TABLE, CREATE VIEW, CREATE FUNCTION, or CREATE PROCEDURE statement for a specific table or for the entire database.
- ▶ Display all GRANT privilege statements that affect a specified user or that affect all users for a database or a specific table. The user can be either a user name or role name.
- ▶ Display user-defined and row data types with or without type inheritance.
- ▶ Display the CREATE SEQUENCE statement defining the specified sequence object, or defining all sequence objects in the database.

When you use the dbschema utility and specify only the database name, it is equivalent to using dbschema with all its options (except for the **-hd** and **-ss** options). In addition, if Information Schema views were created for the database, this schema is shown. For example, the following two commands are equivalent:

```
dbschema -d bookdb
dbschema -s all -p all -t all -f all -d bookdb
```

The SERIAL fields that are included in CREATE TABLE statements that dbschema displays do not specify a starting value. New SERIAL fields created with the schema file have a starting value of 1, regardless of their starting value in the original database. If this value is not acceptable, you must modify the schema file.

You can edit dbschema output to change the owner of a new object.

The dbschema utility uses the *owner.object* convention when it generates any CREATE TABLE, CREATE INDEX, CREATE SYNONYM, CREATE VIEW, CREATE SEQUENCE, CREATE PROCEDURE, CREATE FUNCTION, or GRANT statement, and when it reproduces any unique, referential, or check constraint. As a result, if you use the dbschema output to create a new object (table, index, view, procedure, constraint, sequence, or synonym), the owner of the original object owns the new object. If you want to change the owner of the new object, you must edit the dbschema output before you run it as an SQL script.

You can use the output of dbschema to create a new function if you also specify the path name to a file in which compile-time warnings are stored. This path name is displayed in the dbschema output.

4.1.6 Sequences

A *sequence* (sometimes called a *sequence generator* or *sequence object*) is a database object from which multiple users, or transactions, can generate unique integers within a defined range. The sequence of numbers can run in either ascending or descending order, and is monotonic.

For example, you can use sequences automatically to generate primary key values. When a sequence object is queried, the sequence number is incremented and passed to the query, independently of the transaction committing or rolling back. If two applications increment the same sequence, the sequence numbers that each application acquires might not be sequential because sequence numbers are being generated by the other applications.

One user, or transaction, can never acquire the sequence number generated by another user, or transaction. After a user or transaction generates a sequence value, that value is never generated and passed to another user or transaction.

A database can support multiple sequences concurrently, but the name of a sequence (or in an ANSI-compliant database, the owner.sequence combination) must be unique within the current database among the names of tables, temporary tables, views, synonyms, and sequences.

Informix supports DML statements (CREATE SEQUENCE, ALTER SEQUENCE, RENAME SEQUENCE, and DROP SEQUENCE) for sequence objects that multiple users can access concurrently to generate unique integers in the 8-byte integer range.

You can use the CREATE SEQUENCE statement to create a sequence database object from which multiple users can generate unique integers. Only Informix supports this statement; SQL Server does not have a similar feature. Under certain circumstances, you can use the SQL Server IDENTITY data type as a sequence. This statement is an extension to the ANSI/ISO standard for SQL.

CREATE SEQUENCE can specify the following characteristics of a sequence:

- ▶ Initial value
- ▶ Size and sign of the increment between values
- ▶ Maximum and minimum values
- ▶ Whether the sequence recycles values after reaching its limit
- ▶ The number of values that are pre-allocated in memory for rapid access

Authorized users of a sequence can request a new value by including the sequence.NEXTVAL expression in DML statements. The sequence.CURRVAL expression returns the current value of the specified sequence. NEXTVAL and CURRVAL expressions are valid only within SELECT, DELETE, INSERT, and UPDATE statements; Informix returns an error if you attempt to invoke the built-in NEXTVAL or CURRVAL functions in any other context.

Generated values logically resemble the SERIAL8 data type, but they can be negative, and they are unique within the sequence. Because Informix generates the values, sequences support a much higher level of concurrency than a serial column can. The values are independent of transactions; a generated value cannot be rolled back, even if the transaction in which it was generated fails. Example 4-23 shows a CREATE SEQUENCE statement in Informix.

Example 4-23 CREATE SEQUENCE in Informix

```
CREATE SEQUENCE account_sequence
START WITH 1
MAXVALUE 99999999
INCREMENT BY 1
NOCACHE
NOCYCLE;
```

GRANT and REVOKE statements support access privileges on sequence objects, and you can use the CREATE SYNONYM and DROP SYNONYM statements to reference synonyms for sequence objects in the local database. An error occurs if you include contradictory options, such as specifying both the MINVALUE and NOMINVALUE options, or both CACHE and NOCACHE.

4.1.7 Synonyms

A *synonym* is a name that you can use in place of another SQL identifier. You use the CREATE SYNONYM statement to declare an alternative name for a table, a view, or a sequence object.

You can also create a synonym for any table or view in any database of Informix to which your session is currently connected. You can also create a synonym for an external table that the CREATE EXTERNAL TABLE statement registered in the current database.

You can create a synonym for a table, view, or sequence that exists in a database of an Informix instance that is not your current database server. Both Informix instances must be online when you create the synonym. In a network, the remote database server verifies that the table or view that is referenced by the synonym exists when you create the synonym.

Synonyms are always private in an ANSI-compliant database. If you attempt to create a public synonym or use the `PRIVATE` keyword to designate a private synonym in an ANSI-compliant database, you receive an error. A private synonym can be declared with the same name as a public synonym only if the two synonyms have separate owners.

If you use the `PUBLIC` keyword (or no keyword at all), anyone who has access to the database can use your synonym. If you use the `PRIVATE` keyword to declare a synonym in a database that is not ANSI compliant, the unqualified synonym can be used by its owner. Other users must qualify the synonym with the name of the owner.

Users have the same privileges for a synonym that they have for the database object that the synonym references. The identifier of the synonym must be unique among the names of tables, temporary tables, external tables, views, and sequence objects in the same database. After a synonym is created, it persists until the owner executes the `DROP SYNONYM` statement.

By default, SQL Server creates a private synonym, whereas Informix creates a public synonym. However, SQL Server does not provide an option to create a public synonym; to allow a user to access a synonym created by another user, the owner of the synonym, or the DBA, needs to `GRANT` permissions to that user.

Example 4-24 creates a private and public synonym named `rb_hist` for the `book_history` table in the `bookdb` database located on an Informix instance called `myinstance`. This example shows the difference in usage between the two synonym types in Informix.

Example 4-24 Synonyms in Informix

```
-- Private Synonym in Informix:
CREATE PRIVATE SYNONYM rb_hist
FOR bookdb@myinstance:appl.book_history;

SELECT * from appl.rb_hist;

--Public Synonym in Informix:
CREATE SYNONYM rb_hist FOR bookdb@myinstance:book_history;

SELECT * from rb_hist;
```

The following statement shows the equivalent SQL Server syntax:

```
CREATE SYNONYM rb_hist FOR myinstance.bookdb.book_history;
```

A short synonym makes it easier to write queries, but synonyms can play another role. They allow you to move a table to a separate database, or even to a separate computer, and keep your queries the same. If a synonym refers to a table, view, or sequence in the same database, however, the synonym is automatically dropped if the referenced table, view, or sequence object is dropped.

SQL Server allows you to create synonyms for user-defined functions whereas Informix does not, so you need to drop these synonyms prior to the migration to Informix.

4.1.8 Procedures and functions

Stored procedures are server-side database objects that are used to encapsulate frequently executed SQL statements with flow logic. Procedures are executed on Informix, which is often a much faster machine than the client. Stored procedures help reduce network traffic, because only the original request and the final output need to be transmitted between the client and the server. Both SQL Server and Informix provide the ability to create customized stored procedures. Although stored procedures are supported in both platforms, implementation differences exist.

Informix procedures and functions are considered to be user-defined routines (UDRs). A UDR can be written in the Informix Stored Procedure Language (SPL) or an external language, such as C or Java. A *procedure* is a routine written that does not return a value. A *function* is a routine written that returns a single value, a value with a complex data type, or multiple values.

Informix SPL is an extension to Informix SQL and provides procedural flow control, such as looping and branching. An SPL routine is a generic term that includes both SPL procedures and SPL functions, and you can use SQL and SPL statements to write an SPL routine. You can only use SPL statements inside the following statements:

- ▶ CREATE PROCEDURE
- ▶ CREATE PROCEDURE FROM
- ▶ CREATE FUNCTION
- ▶ CREATE FUNCTION FROM

SPL routines are parsed, optimized, and stored in the system catalog tables in executable format.

Like SQL Server, Informix stored procedures support input, output, and input/output parameters and can be used in SQL statements wherever expressions are allowed.

The way that stored procedures and functions are created differs between SQL Server and Informix. SQL Server uses an AS clause after the CREATE statement to encapsulate the body of the procedure, with an optional BEGIN and END clause. Informix does not have a specific clause to start the body of the procedure, but it uses the END PROCEDURE clause to mark the end of the procedure. However, both SQL Server and Informix use the DROP PROCEDURE statement to delete existing stored procedures.

There are marked differences between the syntax of Transact-SQL (as used in stored procedures) and Informix Stored Procedure Language (SPL). Unfortunately, no tools exist that perform the conversion automatically.

Types of functions and procedures

SQL Server stored procedures are coded using T-SQL and are stored in the database. The maximum number of parameters supported in a stored procedure is 1024. Nested stored procedures are supported up to a limit of 32 nesting levels. Procedure parameters can be declared with default values. Wildcard parameters for strings, using the percent (%) character, are also supported. Parameter values can be passed by explicitly naming the parameters or by providing all the parameter values anonymously but in correct positional order.

SQL Server extended stored procedures are routines that are written in a language, such as C. SQL Server automatically loads and execute these routines just like a regular stored procedure. These procedures run directly in the address space of SQL Server. The dynamic link libraries (DLLs) are created using the Open Data Services (ODS) API.

SQL Server also supports temporary stored procedures. Temporary stored procedures are stored in the tempdb database, which is re-created each time that the server is restarted. These objects do not exist after SQL Server is shut down. Similar to temporary tables, temporary stored procedures can be of two types: private and global. A private temporary stored procedure, identified by a single number sign (#) prefixing the procedure name, is only accessible to the connection that created it. A global temporary stored procedure, identified by a double number sign (##) prefixing the procedure name, is available for invocation by all connections.

SQL Server's user-defined database level functions, such as stored functions, stored procedures, and common language runtime (CLR) procedures all need to be converted to Informix stored procedures using the Informix Stored Procedure Language. SQL Server's stored functions and stored procedures are similar in concept to Informix's stored procedures. SQL Server's CLR routines, however, have no Informix equivalent.

Common language runtime (CLR) stored procedures are similar to database stored procedures and functions in that they accept and return values; however, they are integrated with the .NET Framework. All procedures that are written in a CLR must be rewritten as separate Informix procedures or implemented as external stored procedures referencing external library files. However, the Informix external languages that Informix supports are C and Java only. Therefore, you must first translate all C# and Visual Basic routines to either C or Java.

Because mapping constructs between programming languages is outside the scope of this book, we focus on Transact-SQL routines only.

Parameters

A *parameter* is a formal argument in the declaration of a stored procedure. (When you subsequently invoke a stored procedure that has parameters, you must substitute an actual argument for the parameter, unless the parameter has a default value.)

The name of the parameter is optional for external routines of Informix.

When you create a stored procedure, you declare a name and data type for each parameter. You can specify the data type directly, or you can use the LIKE or REFERENCES clause to specify the data type. You can optionally specify a default value.

You can define any number of SPL routine parameters, but the total length of all parameters passed to an SPL routine must be less than 64 KBs. SQL Server stored procedures greater than 64 K in size can be segmented into smaller Informix stored procedures communicating with each other as though they were one by passing and returning the necessary values.

Informix has a limit of 341 parameters for each stored procedure. This limit also applies to UDRs written in the Java language. UDRs written in the C language can have no more than 102 parameters. In SQL Server, a stored procedure can have up to 2100 parameters.

Both SQL Server and Informix allow you to pass arguments to stored procedure parameters by name, and supply a value for each, or by ordinal position (same order as they are defined). Both DBMSs also allow default values to be assigned to stored procedure parameters.

Declaration and assignment

Informix requires that all variables are defined at the beginning of each stored procedure control block:

```
DEFINE out_emp_id LIKE employee.emp_id;
DEFINE counter INTEGER;
```

SQL Server variables are declared using the DECLARE statement and preceding the variable name with an at sign (@). Informix variables are created using the DEFINE statement and can be any legal identifier (that is, no reserved words and less than 128 characters long.)

Important: Informix identifiers cannot begin with the at sign (@). You must establish standards for your organization when converting procedure variables, for example, precede all variables with “v_” or precede all input variables with “i_” and output variables with “o_”.

You must define all Informix stored procedure variables at the top of the procedure, *before* any other statements.

Example 4-25 highlights the difference between variable declarations in Informix and SQL Server stored procedures.

Example 4-25 Variable declarations in SQL Server and Informix

--SQL Server Stored Procedure Example:

```
CREATE PROCEDURE myproc(@fname CHAR(10)) AS
    SELECT emp_id,
    FROM employee
    WHERE firstname = @fname
```

--Informix Stored Procedure Example:

```
CREATE PROCEDURE myproc(i_fname CHAR(10)) RETURNING INT;
    DEFINE o_emp_id INT;
    SELECT emp_id
    INTO o_emp_id
    FROM employee
    WHERE firstname = i_fname;
    RETURN o_emp_id;

END PROCEDURE;
```

Additionally, you need to port assignment operations. SQL Server uses the SET keyword, and Informix utilizes the LET keyword. Example 4-26 highlights the difference in variable assignments between Informix and SQL Server.

Example 4-26 Variable assignments in SQL Server and Informix

```
--SQL Server assignment statement:  
SET @var = 1;  
  
--Informix assignment statement:  
LET var = 1;
```

Invocation

In Informix, you can use the following statements to invoke a stored procedure or function:

```
EXECUTE FUNCTION function_name (parameters ...)  
EXECUTE PROCEDURE procedure_name (parameters ...)  
CALL procedure_name (parameters ...)
```

SQL Server uses only one statement, and its abbreviation, to invoke stored procedures:

```
EXECUTE procedure_name parameters ...  
EXEC procedure_name parameters ...
```

The key difference in the previous statements is that Informix requires that the stored procedure (or function) parameters are contained within parentheses () whereas SQL Server does not. Example 4-27 highlights the difference between SQL Server and Informix when invoking a procedure called `add_brand()`, which requires three input parameters.

Example 4-27 Stored procedure invocation in SQL Server and Informix

```
-- SQL Server example:  
EXECUTE add_brand 101, 'My Brand', 'Info on My Brand';  
  
-- Informix Example:  
  
EXECUTE PROCEDURE add_brand(101, 'My Brand', 'Info on My Brand');  
  
CALL add_brand(101, 'My Brand', 'Info on My Brand');
```

SQL Server can access procedures from within procedures using the EXECUTE statement. In the following example, the return_status is the status of the procedure execution, parameter1 and parameter2 are the input parameters, and parameter3 is the value returned by the procedure execution:

```
EXECUTE return_status = procedure_name parameter1, parameter2, parameter3  
OUTPUT
```

Informix supports two methods for calling a procedure from within another procedure:

```
CALL  
EXECUTE PROCEDURE
```

Informix uses the following method for calling a procedure:

```
CALL procedure_name([args...]) [RETURNING variables...]  
EXECUTE PROCEDURE procedure_name([args...]) [ INTO variables...]
```

Note that the parentheses () after the procedure name are always required, even though the arguments can be optional.

Return values

SQL Server supports two types of functions: scalar functions (functions that return a single scalar value such as INT or CHAR) as well as table functions (functions that return a table). Table functions can be further classified as in-line functions or multi statement table-valued functions.

If the RETURNS clause specifies TABLE with no additional table definition information, the function is an in-line function and there must be a single SELECT statement as the body of the function. If the RETURNS clause uses the TABLE statement while specifying a table variable name, the function is a multi statement table-valued function. The RETURNS clause also lists the columns and data types for the table.

The biggest difference between the SQL Server and Informix stored procedures is with returning rows. SQL Server implicitly returns whatever you select, including the user of TABLE functions, whereas Informix is more structured about declaring what you want to return, and knowing when you want to return multiple rows.

Returning result sets

Whereas SQL Server stored procedures return a result set for each select statement (unless it selects into local variables), Informix stored procedures are more restrictive on what can be returned. In particular, Informix requires that you declare the structure of the result set using the RETURNING statement.

As Example 4-28 shows, SQL Server can have a stored procedure return multiple result sets of separate data types and structure.

Example 4-28 Stored procedure in SQL Server returning multiple sets

```
CREATE PROCEDURE showall AS
    SELECT count(*) FROM booktab
    SELECT brand_id, brand_name FROM brands
    SELECT job_id FROM jobs
```

When executed, the previous stored procedure returns three separate result sets.

Informix does not support multiple result sets of separate data types and structure. To work around this situation, Example 4-29 shows how you can split the previous SQL Server procedure into multiple procedures in Informix.

Example 4-29 Stored procedure in Informix returning multiple sets

```
CREATE PROCEDURE showall_1() RETURNING INT;
    DEFINE o_books_count INT;
    SELECT count(*)
        INTO o_books_count
        FROM booktab;
    RETURN o_books_count;
END PROCEDURE;

CREATE PROCEDURE showall_2() RETURNING INT, VARCHAR(20);
    DEFINE o_brand_id INT;
    DEFINE o_brand_name VARCHAR(20);
    SELECT brand_id, brand_name
        INTO o_brand_id, o_brand_name
        FROM brands;
    RETURN o_brand_id, o_brand_name;
END PROCEDURE;

CREATE PROCEDURE showall_3() RETURNING INT;
    DEFINE o_job_id INT;
    SELECT job_id
        INTO o_job_id
        FROM jobs;
    RETURN o_job_id;
END PROCEDURE;
```

You can also accomplish the previous example in a single, more efficient stored procedure, as shown in Example 4-30.

Example 4-30 Efficient method: Stored procedure in Informix returning multiple sets

```
CREATE PROCEDURE showall
    RETURNING INT, INT, VARCHAR(20), INT;
    DEFINE o_books_count INT;
    DEFINE o_brand_id INT;
    DEFINE o_brand_name VARCHAR(20);
    DEFINE o_job_id INT;
    SELECT count(*)
        INTO o_books_count
        FROM booktab;
    SELECT brand_id, brand_name
        INTO o_brand_id, o_brand_name
        FROM brands;
    SELECT job_id
        INTO o_job_id
        FROM jobs;
    RETURN o_books_count, o_brand_id, o_brand_name, o_job_id;
END PROCEDURE;
```

Although multiple results sets can be returned in Informix, they must all be explicitly typed and returned in an expected order.

Returning a row

SQL Server procedures return a result set for each select statement, without any modification to the syntax. Any legal SQL statement will return data, as shown in Example 4-31.

Example 4-31 Stored procedure in SQL Server returning a row

```
CREATE PROCEDURE myproc() AS
    SELECT book_id, book_no, title
    FROM booktab
    WHERE book_id = 100
```

Informix, however, requires that the data is explicitly selected into local variables and then is returned through a separate RETURNING statement, which is demonstrated in Example 4-32 on page 113.

Example 4-32 Stored procedure in Informix returning a row

```
CREATE PROCEDURE myproc() RETURNING INT, CHAR(10), VARCHAR(80);

    -- Declare internal variables used to return data.
    DEFINE o_book_id INT;
    DEFINE o_book_no CHAR(10);
    DEFINE o_title VARCHAR(80);

    -- Explicitly select into variables.
    SELECT book_id, book_no, title
        INTO o_book_id, o_book_no, o_title
        FROM booktab
        WHERE book_id = 100;

    -- Explicitly return values.
    RETURN o_book_id, o_book_no, o_title;

END PROCEDURE;
```

Returning multiple rows

When returning multiple rows, Informix stored procedures require the use of a FOREACH...END FOREACH loop with a RETURN WITH RESUME statement. Example 4-33 illustrates the use of the FOREACH clause in Informix.

Example 4-33 Stored procedure in Informix returning multiple rows

```
CREATE PROCEDURE myproc() RETURNING INT, INT, CHAR(10);

    -- Declare internal variables used to return data.
    DEFINE o_book_id INT;
    DEFINE o_book_no CHAR(10);
    DEFINE o_title VARCHAR(80);

    -- Explicitly select into variables.
    FOREACH
        SELECT book_id, book_no, title
            INTO o_book_id, o_book_no, o_title
            FROM booktab
            WHERE book_id > 100;
        RETURN o_book_id, o_book_no, o_title WITH RESUME;
    END FOREACH;

END PROCEDURE;
```

Example 4-34 displays the equivalent SQL Server procedure.

Example 4-34 Stored procedure in SQL Server returning multiple rows

```
CREATE PROCEDURE myproc() AS
  SELECT book_id, book_no, title
  FROM booktab
  WHERE cust_num > 100
```

Although multiple row sets can be returned by Informix, they must all have the same data types and structure.

An alternate solution in Informix, when the data types and structures differ, is to declare all variables at the beginning of the procedure. Then, all results can be returned by each select (including empty variables), as depicted in Example 4-35.

Example 4-35 Stored procedure in Informix returning multiple rows: Declare all variables

```
CREATE PROCEDURE myproc() RETURNING INT, VARCHAR(40), VARCHAR(80);
  DEFINE o_book_id INT;
  DEFINE o_author_name VARCHAR(40);
  DEFINE o_title VARCHAR(80);

  FOREACH
    SELECT book_id
      INTO o_book_id -- Explicitly select into variables.
    FROM booktab
    WHERE book_id = 100;

    -- Explicitly return values.
    LET o_author_name = "xxxxx";
    LET o_name = "xxxxx";
    RETURN o_book_id, o_author_name, o_title WITH RESUME;
  END FOREACH;

  FOREACH
    SELECT name
      INTO o_author_name -- Explicitly select into variables.
    FROM booktab, authors
    WHERE book_id = 100
    AND booktab.author_id = authors.author_id;

    -- Explicitly return values.
    LET o_book_id = "99999";
    LET o_name = "xxxxx";
    RETURN o_book_id, o_author_name, o_title WITH RESUME;
  END FOREACH;
```

```

FOREACH
  SELECT title
    INTO o_title -- Explicitly select into variables.
  FROM booktab, book_history
  WHERE booktab.book_id = 100
  AND booktab.book_id = book_history.book_id;

  -- Explicitly return values.
  LET o_book_id = "99999";
  LET o_author_name = "xxxxx";
  RETURN o_book_id, o_author_name, o_title WITH RESUME;
END FOREACH;

END PROCEDURE;

```

In Example 4-35 on page 114, the application needs to know how to interpret the returning data, probably through finding a set value ("99999") in one of the value slots.

Returning status

SQL Server procedures return an integer status value:

- ▶ 0 if successful
- ▶ -1 through -99 for SQL Server errors

You also can return a user-defined error outside the 0-99 range by using the **RETURN** statement, for example, a line of code, such as this example:

```
EXEC @status = myproc
```

In SQL Server, this line returns the **STATUS** of the `myproc` procedure into the `@status` variable. This returning status differs from Informix in that Informix procedures return data, for example:

```
EXECUTE PROCEDURE myproc() INTO status
```

The previous line returns the first data value that is specified to be returned in the `myproc` procedure. It is, therefore, difficult within Informix to return both data and status from a procedure.

SQL Server also has a wider range of user-defined error numbers. The **RAISERROR** function has this syntax:

```
RAISERROR error_number { "error_string" }
```

The following example shows the SQL Server **RAISERROR** function:

```
RAISERROR 25001 "Book not in booktab table"
```

SQL Server reserves the RAISERROR range of values 0 - 50,000. You can use any number outside this range. SQL Server can also return the names of the tables and columns that are causing an error, along with the error number and text.

Informix uses a single error number when returning a user-defined error text, for example:

```
RAISE EXCEPTION -746, 0, "Book not in booktab table"
```

The value -746 is the only number that can be used to return a user message.

Error handling

SQL Server provides the global variables, @@error and @@rowcount, that are immediately set after each SQL statement. The @@ERROR function is used to check for a success or failure status after the evaluation of a statement. A RAISERROR statement is equivalent to a PRINT statement, followed by an assignment to @@ERROR.

Similar information can be retrieved from an Informix stored procedure in the following fashion. For example, to get rowcount information, define a variable, and then make a call to the DBINFO function after the SQL statement, as shown in Example 4-36.

Example 4-36 Use of DBINFO in Informix to get rowcounts

```
DEFINE nrow INT;  
{ SQL statements }  
LET nrow = DBINFO('sqlca.sqlerrd2');
```

Error information is captured using the ON EXCEPTION statement in Informix procedures. For example, Example 4-37 performs an update in the ON EXCEPTION statement block after an insert is attempted and a duplicate row error is returned.

Example 4-37 Use of the ON EXCEPTION clause in Informix stored procedures

```
CREATE PROCEDURE sksp2 (i_col1 INT, i_col2 CHAR(20))  
  DEFINE errsql INT;  
  ON EXCEPTION  
    SET errsql  
    IF (errsql = -239)  
      THEN  
        UPDATE sktab  
        SET col2 = col2 + 10  
        WHERE col1 = i_col1;  
      END IF;
```

```
END EXCEPTION WITH RESUME;  
INSERT INTO sktab VALUES (i_col1, i_col2);  
END PROCEDURE;
```

When SQL Server completes the execution of a Transact-SQL statement, @@ERROR is set to the value 0 if the statement executed successfully. If an error occurs, an error message is returned. @@ERROR returns the number of the error message until another Transact-SQL statement is executed.

The equivalent SQLCA structure is not fully available in Informix stored procedures. It is not possible to check the SQLCODE variable in the stored procedure body. Instead, the function DBINFO can be used to extract the value of two members of the SQLCA structure:

- ▶ sqlca.sqlerrd1
- ▶ sqlca.sqlerrd2

Use this function inside the FOREACH construct while the cursor is open. After the cursor is closed, the DBINFO function cannot return proper values.

The sqlca.sqlerrd1 option returns separate values depending on the type of SQL statement. For INSERT statements, if the table contains a serial column, the sqlca.sqlerrd1 option will return the serial value of the last row inserted into the table. For SELECT, DELETE, and UPDATE statements, the sqlca.sqlerrd1 option will return the number of rows processed by the query. In these cases, on each pass through the FOREACH loop, sqlca.sqlerrd1 is the same value, which is the number of rows the FOREACH loop will process. The sqlca.sqlerrd1 value can still be interrogated during each pass through the FOREACH loop.

The sqlca.sqlerrd2 option returns the number of rows processed by a SELECT, INSERT, UPDATE, DELETE, or EXECUTE PROCEDURE statement. It must not be interrogated until after the cursor has finished processing. Within the FOREACH loop, the sqlca.sqlerrd2 value can be moved to a variable that is then interrogated outside of the FOREACH loop.

Cursors

Cursors are explicitly declared, opened, and fetched in SQL Server stored procedures. In Informix, cursors do not need to be explicitly declared, opened, and fetched in stored procedures. Instead, SQL Server stored procedure cursors can be replaced with Informix's FOREACH construct. The SQL Server cursor name must be used in the FOREACH statement, for example:

```
FOREACH cursor_name  
    SELECT ...  
END FOREACH
```

In order to update a cursor row in SQL Server, the cursor must be declared with the FOR UPDATE clause. In Informix, if the SELECT statement in the FOREACH construct is not a multi-table join, each fetched row can be updated using the statement:

```
UPDATE <table_name> WHERE CURRENT OF <cursor_name>
```

Informix does not support the SELECT FOR UPDATE statement in a stored procedure. Therefore, the SELECT FOR UPDATE clause cannot be used in the FOREACH construct.

In Informix-SPL, the DECLARE, OPEN, FETCH, CLOSE, and FREE statements are all combined into a single looping statement called FOREACH.

In Example 4-38, the book_id, book_no, and title columns are being selected from the book_history table. Each tuple that is returned is placed into the three variables p_book_id, p_book_no, and p_title. These three values then are returned to the calling program.

The WITH RESUME causes Informix-SPL to continue from this point the next time that the stored procedure is called. The FOREACH will continue to loop through the tuples in this fashion until all rows are processed at which point the cursor is closed and the FOREACH loop is exited.

Example 4-38 Use of the FOREACH statement in Informix stored procedures

```
FOREACH
  SELECT book_id, book_no, title
  INTO p_book_id, p_book_no, p_title
  FROM book_history

  RETURN p_book_id, p_book_no, p_title
  WITH RESUME;
END FOREACH;
```

In Informix-ESQL/C, cursors operate in the same fashion as in Transact-SQL. The only difference is the DEALLOCATE CURSOR statement, which is converted to the FREE statement for Informix-SQL.

Comments

Both SQL Server and Informix support the ANSI standard use of a double hyphen “--” for single line comments. SQL Server uses the “/*...*/” syntax for multiple line comments, while Informix uses “{ ...}” for multiple line comments.

Example 4-39 illustrates the usage of comments in SQL Server and Informix stored procedures.

Example 4-39 Comments in SQL Server and Informix stored procedures

SQL Server:

```
-- single line comment
/* multiple
   line comment */
```

Informix:

```
-- single line comment
{ multiple
  line comment }
```

System stored procedures

SQL Server provides an extensive and rich set of built-in stored procedures called *system procedures*. These system procedures have the following characteristics:

- ▶ Shortcuts for retrieving information from system tables
- ▶ Mechanisms for accomplishing database administration and other tasks involving updating system tables
- ▶ Prefixed with “sp_”

Because system procedures pertain to querying and maintaining system tables, including performing administration activities, we do not discuss the Informix equivalent here.

Dynamic SQL

Starting with Informix Version 11.50, Informix also supports dynamic SQL in SPL. Now, you can dynamically construct and execute SQL statements.

The following example shows the syntax of a dynamic SQL statement in Informix:

```
EXECUTE IMMEDIATE { SQL_quoted_string | Str_variable }
```

We explain this example:

- ▶ `SQL_quoted_string`: A string containing a single SQL statement
- ▶ `Str_variable`: A character variable containing the SQL statement

Example 4-40 shows sample code for using dynamic SQL in Informix SPL.

Example 4-40 Dynamic SQL in Informix SPL

```
CREATE PROCEDURE MYPROC()
RETURNING INT;
DEFINE A0 VARCHAR(30);
DEFINE A1 VARCHAR(5);
DEFINE A2 INT;
DEFINE A3 VARCHAR(60);
DEFINE A4 INT;
LET A0 = "INSERT INTO DYN_TAB VALUES (";
LET A1 = ")";
FOR A2 = 1 TO 100
LET A3 = A0 || A2 || A1;
EXECUTE IMMEDIATE A3 ;
END FOR;
SELECT COUNT(DISTINCT C1) INTO A4 FROM T1;
RETURN A4;
END PROCEDURE;
```

Refer to the *IBM Informix Guide to SQL: Syntax*, SC23-7751, for more information about dynamic SQL.

SPL availability

With SQL Server, the Transact-SQL language is available from anywhere within the environment. For example, you can create a file containing any of the T-SQL commands and then pipe it into the engine. With Informix, however, the SPL language is only available within a stored procedure. You can pipe any SQL statements from a file, such as CREATE TABLE, CREATE PROCEDURE, SELECT, INSERT, and UPDATE, successfully into the engine using the dbaccess utility. However, access to SPL extensions, such as FOR loops and IF statements, is not allowed outside a procedure.

Exceptions

The methods for handling exceptions are similar between Informix and SQL Server. SQL Server uses the TRY ... CATCH blocks whereas Informix uses the ON EXCEPTION statement. Both SQL Server and Informix also have the capability to RAISE exceptions.

Informix supports predefined and user-defined exceptions, both of which are represented numerically. All the exceptions checked in the stored procedure control blocks, delimited with BEGIN and END statements, must be declared explicitly at the top of each control block with the Informix ON EXCEPTION construct. The scope of an exception mechanism is always restricted to the block

in which it is located. This restriction leads to the addition of multiple Informix exception handling mechanisms, as well as a redesign of the overall stored procedure to a more blocked format.

Compiler

The Informix compiler was developed with the concept that developers can create stored procedures independently of the database. Therefore, Informix's stored procedure compiler has a limited set of errors that it checks.

For example, the Informix stored procedure compiler does not validate database object names, such as TABLE, COLUMN, and VIEW names. Therefore, database object naming errors will occur at run time, not compile time. Additionally, the error messages generated from the compiler are not specific. Identifying a problem in a stored procedure might require breaking it apart and recompiling the pieces until the problem is found. This approach can affect the amount of time that is necessary to unit test the stored procedures.

4.1.9 Triggers

Triggers are server-side database objects that are commonly used to maintain complex data integrity rules, to implement referential actions such as cascading DELETES, and to maintain an audit record of changes. Both SQL Server and Informix support triggers, although differently.

You code SQL Server triggers using T-SQL and store them directly in the database. SQL Server provides three types of triggers:

- ▶ A User Logging trigger is executed when a client connects to Informix.
- ▶ An AFTER trigger is executed one time for each INSERT, UPDATE, or DELETE statement, regardless of the number of rows that it affects.
- ▶ INSTEAD OF triggers replace the triggering SQL statement with the logic provided in the trigger body.

SQL Server supports multiple AFTER triggers per table, but only one INSTEAD OF trigger is permitted per table. You can use the system stored procedure `sp_settriggerorder()` to set the first and last AFTER trigger to be executed; any other AFTER triggers are executed in random order.

SQL Server supports INSERT, UPDATE, and DELETE triggers. A trigger can be associated with more than one type of INSERT, UPDATE, and DELETE event. Triggers can be nested up to 32 levels. Direct or indirect recursion is available as a database configuration option.

You code Informix triggers in-line with the CREATE TRIGGER statement, store them in the database, and compile them at run time with the SQL statements that are associated with the trigger. Informix supports up to three action clauses in a trigger: BEFORE, AFTER, and FOR EACH ROW:

- ▶ The BEFORE actions are executed one time for each triggering event, before Informix performs the triggering DML operation. This trigger therefore fires one time for each statement, no matter how many rows are affected by the triggering statement.
- ▶ The AFTER actions are also executed one time for each triggering DML event, after the operation on the table is complete, in the context of the triggering statement. This action is similar to that in SQL Server. This trigger therefore fires one time for each statement, no matter how many rows are affected by the triggering statement.
- ▶ The FOR EACH ROW actions are executed for each row that is inserted, updated, deleted, or selected in the DML operation, after the DML operation is executed on each row, but before Informix writes the values into the log and into the table. This trigger therefore fires every time that a row in the target table is affected by the triggering SQL statement.

The MERGE statement can also be the triggering event for an UPDATE, DELETE, or INSERT trigger. The event definition must specify the table or view on which the trigger is defined. (SELECT or UPDATE events for triggers on tables can also specify one or more columns.)

You can define multiple triggers for the same event. When a single triggering event executes multiple triggers, the order of execution is not guaranteed, but all of the BEFORE triggered actions execute before any of the FOR EACH ROW triggered actions, and all of the AFTER triggered actions execute after all of the FOR EACH ROW triggered actions.

A trigger action is composed of one or more SQL procedural statements, which can contain a dynamic compound statement or any of the SQL control statements.

User login triggers

Informix does not allow you to create a trigger for user/client logins. However, Informix includes a feature whereby a specific stored procedure called sysdbopen is executed every time that a user/client connects to the Informix instance.

The sysdbopen procedure is executed whenever users successfully issue the DATABASE or CONNECT statement to explicitly connect to a database where the procedures are installed.

To set the initial environment for one or more user sessions, create and install the `sysdbopen` SPL procedure. The typical effect of this procedure is to initialize properties of a session without requiring the properties to be explicitly defined within the session. Such procedures are useful if users access databases through client applications that cannot modify application code or set environment options or environment variables.

The `sysdbopen` procedure is an exception to the scope rule for stored procedures. In ordinary UDR procedures, the scope of variables and statements is local. `SET PDQPRIORITY` and `SET ENVIRONMENT` statement settings do not persist when these SPL procedures exit. In `sysdbopen`, however, statements that set the session environment remain in effect until another statement resets the options, or until the session ends.

For example, Example 4-41 sets the transaction isolation level to `REPEATABLE READ`, and sets the `OPTCOMPIND` environment variable in procedure `sysdbopen()` to instruct the query optimizer to prefer nested-loop joins. When a user who owns no other `user.sysdbopen` procedure connects to the database, this routine will be executed.

Example 4-41 Use of `sysdbopen()` procedure

```
CREATE PROCEDURE public.sysdbopen()  
    SET ISOLATION TO REPEATABLE READ;  
    SET ENVIRONMENT OPTCOMPIND '1';  
END PROCEDURE;
```

Accessing transition values in triggers

SQL Server maintains special pseudo tables called `deleted` and `inserted` that have the same set of columns as the underlying table being changed. A trigger has access to the before image and after image of the data through the special pseudo tables. The before and after values of specific columns can be checked and action taken depending on the values encountered. The `inserted` table can be referenced in a `DELETE` trigger and the `deleted` table can be referenced in an `INSERT` trigger. These tables are empty when the corresponding triggers are activated.

The Informix `CREATE TRIGGER` statement allows transition variables to be referenced using correlation names. Correlation names identify a specific row in the set of rows affected by the triggering SQL statement. Each row or set of rows affected by the triggering SQL statement is available to the triggered action by qualifying the desired columns with correlation names.

Informix uses the `REFERENCING` clause to declare correlation names (or for Update triggers, two correlation names). These names enable `FOR EACH ROW`

actions to reference the new or old column values in the result of trigger events. For example, Example 4-42 references the original value of the `basic_salary` column with the prefix `pre` and the updated value with the prefix `post`.

Example 4-42 Use of the REFERENCING clause in Informix stored procedures

```
CREATE TRIGGER emp_sal
  UPDATE OF basic_salary ON employee
  REFERENCING OLD AS pre NEW AS post
  FOR EACH ROW WHEN(post.basic_salary > pre.basic_salary * 2)
  (INSERT INTO warn_tab
    VALUES(pre.emp_id, pre.name, pre.job_id,
      post.basic_salary, CURRENT));
```

Accessing transition values in stored procedures

You can define specialized SPL routines, called *trigger routines*, that can be invoked only from the FOR EACH ROW section of the triggered action. Unlike ordinary UDRs that EXECUTE FUNCTION or EXECUTE PROCEDURE routines can call from the triggered action list, trigger routines include their own REFERENCING clause that defines correlation names for the old and new column values in rows that the triggered action modifies. You can reference these correlation names in SPL statements within the trigger routine, providing greater flexibility in how the triggered action can modify data in the table or view.

Trigger routines can also use trigger-type Boolean operators, called DELETING, INSERTING, SELECTING, and UPDATING, to identify what type of trigger has called the trigger routine. Trigger routines can also invoke the `mi_trigger*` routines, which are sometimes called *trigger introspection routines*, to obtain information about the context in which the trigger routine has been called.

Trigger routines are invoked by EXECUTE FUNCTION or EXECUTE PROCEDURE statements that include the WITH TRIGGER REFERENCES keywords. These statements must call the trigger routine from the FOR EACH ROW section of the triggered action, rather than from the BEFORE or AFTER sections.

Example 4-43 on page 125 creates an UPDATE trigger and the associated trigger routine. The trigger routine also includes boolean operators. The trigger `tr_employee` fires when the salary of any employee is updated. The trigger routine `trig_employee()` creates an audit of the change in salary by inserting a record in the table `log_records`. This same routine can be called by other INSERT, SELECT, and DELETE triggers, also.

Example 4-43 Informix trigger routine

```
CREATE TRIGGER tr_employee
UPDATE OF basic_salary ON employee
FOR EACH ROW (
EXECUTE PROCEDURE trig_employee() WITH TRIGGER REFERENCES);

CREATE PROCEDURE trig_employee()
    REFERENCING OLD as pre NEW as post for items;

    if (INSERTING)
    then
        insert into log_records
        values('I', pre.emp_id, post.basic_salary,post.basic_salary);
    end if
    if (UPDATING) then
    then
        insert into log_records
        values('U', pre.emp_id, post.basic_salary,post.basic_salary);
    end if
    if (SELECTING) then
    then
        insert into log_records
        values('S', pre.emp_id, post.basic_salary,post.basic_salary);
    end if
    if (DELETING) then
    then
        insert into log_records
        values('D', pre.emp_id, post.basic_salary,post.basic_salary);
    end if
END PROCEDURE;
```

Simple conversion example

Example 4-44 on page 126 shows a straightforward SQL Server AFTER trigger. This trigger fires after an INSERT or UPDATE operation occurs on the `booktab` table. This trigger enforces the business rule that if the author does not exist, it must be created. Whenever a new record is inserted into the `booktab` table, or a row in the `booktab` table is updated with a new `author_id` value, if the `author_id` does not exist in the `authors` table, the trigger will insert a new record with the `author_id` value, and default values for the rest of the columns, into the `authors` table. You can also use a FOREIGN KEY constraint on the `author_id` column instead of the trigger.

Example 4-44 SQL Server AFTER trigger

```
CREATE TRIGGER tr_ins_upd_brand
    ON booktab
    FOR INSERT, UPDATE
AS
-- Does the author exist?
IF NOT EXISTS
    (SELECT * FROM inserted WHERE inserted.author_id IN
      (SELECT author_id FROM authors) )
BEGIN
    DECLARE @VAL CHAR(3)
    SET @VAL = "IBM"
    INSERT INTO AUTHORS VALUES
        (inserted.author_id, @VAL, @VAL, @VAL, @VAL, @VAL, @VAL, @VAL)
END
GO
```

Example 4-45 shows the equivalent triggers in Informix.

Example 4-45 Equivalent Informix AFTER triggers

```
CREATE TRIGGER tr_upd_auth
    UPDATE OF author_id ON booktab
    REFERENCING OLD AS deleted NEW AS inserted
    FOR EACH ROW
    WHEN (NOT EXISTS
        (SELECT * FROM authors WHERE author_id = inserted.author_id))
    (INSERT INTO AUTHORS VALUES
        (inserted.author_id, "", "", "", "", "", "", ""))

CREATE TRIGGER tr_ins_brand
    INSERT ON booktab
    REFERENCING NEW AS inserted
    FOR EACH ROW
    WHEN (NOT EXISTS
        (SELECT * FROM authors WHERE author_id = inserted.author_id))
    (INSERT INTO AUTHORS VALUES
        (inserted.author_id, "", "", "", "", "", "", ""))
```

Two Informix triggers are created: a trigger for the INSERT operation and another trigger for the UPDATE operation. You must define separate triggers for each type of triggering SQL statement in Informix.

It is also important to note that, in Informix, a DELETE trigger cannot reference the NEW transition table and an INSERT trigger cannot reference the OLD transition table. To directly convert such functionality, references to the NEW table in a

DELETE trigger and the OLD table in an INSERT trigger can be translated into a query that evaluates to an empty table (with the correct set of columns). In many cases, the statement containing such references is redundant, and the trigger can be simplified by deleting the statement.

INSTEAD OF triggers

The SQL Server INSTEAD OF triggers override the triggering action with the logic specified in the trigger body. You can define INSTEAD OF triggers on views, but you cannot define AFTER triggers on views. You can also define INSTEAD OF triggers on tables. You can only define one INSTEAD OF trigger for each triggering action (INSERT, UPDATE, and DELETE) on a table or view.

In Informix, you can only define an INSTEAD OF trigger on a view to perform an INSERT, UPDATE, or DELETE request on behalf of the view. When fired, the triggering SQL statement against the view gets replaced by the trigger logic, which performs the operation on behalf of the view. An Informix INSTEAD OF trigger is activated after the triggering statement is issued to the base view. A view can have any number of INSTEAD OF triggers defined for each type of event (INSERT, DELETE, or UPDATE).

Example 4-46 shows an SQL Server INSTEAD OF trigger defined on a view. This trigger fires when an INSERT statement is attempted against the view `v_myView`. Instead of inserting into the view, the logic contained in the trigger body is executed, namely inserting the first two columns of the rows that were attempted to be inserted into `myView` into the table `myTable`. The following view and table definitions are assumed by this example:

```
CREATE TABLE myTable (col1 INTEGER, col2 INTEGER)
CREATE VIEW myView AS SELECT col1, col2 FROM myTable
```

Example 4-46 SQL Server INSTEAD OF trigger defined on a sample view

```
-- The following definitions are assumed for this example
-- CREATE TABLE myTable (col1 INTEGER, col2 INTEGER)
-- CREATE VIEW myView AS SELECT col1, col2 FROM myTable
```

```
CREATE TRIGGER tr1 on v_myView
INSTEAD OF INSERT
AS
BEGIN
    INSERT INTO myTable
        SELECT col1, col2 FROM inserted
END
GO
```

Example 4-47 on page 128 shows the equivalent Informix INSTEAD OF trigger.

Example 4-47 Equivalent Informix *INSTEAD OF* trigger

```
-- The following definitions are assumed for this example
-- CREATE TABLE myTable (col1 INTEGER, col2 INTEGER)
-- CREATE VIEW myView AS SELECT col1, col2 FROM myTable

CREATE TRIGGER tr1
  INSTEAD OF INSERT ON myView
  REFERENCING OLD AS o NEW AS n
  FOR EACH ROW
  (INSERT INTO myTable
   VALUES(n.col1, n.col2))
```

You can convert SQL Server *INSTEAD OF* triggers that are defined on views as Informix *INSTEAD OF* triggers. SQL Server *INSTEAD OF* triggers that are defined on normal tables differ significantly in behavior from Informix *BEFORE* and *AFTER* triggers. A direct translation to Informix *INSTEAD OF* triggers is not valid, because the SQL Server *INSTEAD OF* trigger action is executed instead of the triggering statement. In Informix *BEFORE* and *AFTER* triggers, both the triggering action *and* the triggering statements are executed eventually. Trigger redesign effort is required to convert this type of functionality to Informix if the same behavior is desired.

One method of performing this conversion involves renaming the original table, then creating a view with the same name as the original table. All the original statements in the application code referencing the table will now be referencing the view. You can then convert the *INSTEAD OF* trigger to an Informix *INSTEAD OF* trigger, because it will operate on the newly created view.

IF UPDATE(column)

You can use the SQL Server *IF UPDATE(column)* condition within a trigger to test if a particular column's value has been modified by the triggering SQL statement.

Informix supports similar functionality although the syntax differs slightly. To convert this condition to Informix, specify the column in the trigger's activation condition with the *UPDATE OF <column>* clause. Example 4-48 on page 129 shows an SQL Server trigger where the *IF UPDATE(column)* function is used. This trigger fires after an update operation occurs on the *authors* table, and the trigger logic only executes if the *contract* column is modified.

Example 4-48 SQL Server trigger using the IF UPDATE(column) function

```
CREATE TRIGGER tr_authors
    ON authors
    FOR UPDATE
AS
    IF UPDATE(contract)
        -- executable-statements
GO
```

Example 4-49 shows another equivalent Informix trigger using the UPDATE OF <column> syntax to simulate the IF UPDATE(column) function.

Example 4-49 Equivalent Informix trigger using the UPDATE OF <column> clause

```
CREATE TRIGGER tr_authors
    UPDATE OF contract ON authors
    REFERENCING OLD AS o NEW AS n
    FOR EACH ROW
    -- executable-statements
```

Overcoming in-line SPL limitations in triggers

The Informix procedural language for triggers is actually a limited subset of Informix's SPL. Certain SQL elements that are supported in stored procedures are not supported in triggers. Specifically, the following commonly used statements are not supported in Informix triggers:

- ▶ Transactional statements (COMMIT and ROLLBACK)
- ▶ Dynamic SQL statements (PREPARE, EXECUTE, and EXECUTE IMMEDIATE)
- ▶ Cursor declarations (DECLARE <cursor>, and ALLOCATE CURSOR)
- ▶ Complex exception handling (DECLARE ... HANDLER, and RESIGNAL)
- ▶ SQL constructs (SELECT INTO, LOOP, and REPEAT)

Therefore, to convert an SQL Server trigger using any of the previous features, you need to move the unsupported statements into a new stored procedure and invoke the stored procedure from the trigger.

Important: If a procedure is invoked by a trigger, a ROLLBACK or COMMIT statement is not allowed in the body of the procedure unless it is rolled back to a savepoint that is defined in the same procedure.

It is important to be aware of read and write conflict errors that can occur when an SQL stored procedure is invoked by a trigger in Informix. A set of data integrity rules is enforced when an SQL stored procedure is invoked by a trigger. Essentially, Informix does not allow conflicting operations to occur when a stored

procedure invoked by a trigger concurrently accesses the same table. If such a conflict occurs, an error is returned to the statement that caused the conflict at run time. It is therefore important to test all conditions to ensure that conflict errors do not occur at run time.

Trigger responding to DDL activity

SQL Server allows you to create triggers to respond to DDL activity, including auditing this activity. For example, you can create a trigger to prevent a user from creating or dropping a table.

Informix does not support DDL triggers. However, you can implement similar functionality using the Informix security features. You can set up database-level and table-level privileges so that only authorized users are allowed to perform specific database functions. For example, you can restrict the *Resource* privilege to specific individuals so that only those individuals can create new, permanent tables, indexes, and SPL routines.

Informix also has an auditing feature to monitor one or more database activities for all or specific users. Auditing creates a record of selected activities that users perform. An audit administrator who analyzes the audit trail can use these records to detect unusual, unauthorized, or suspicious user actions and identify the specific users who performed those actions.

4.2 DML

Most relational database management systems (RDBMSs) are either fully compliant or conform to one of the SQL standards. Thus, you can convert many SQL statements and queries that were written for SQL Server to Informix without modification. However, certain SQL syntax and semantic differences exist across DBMS types, depending on what standard is implemented and the level of conformance.

This section highlights several SQL language syntactical and semantic differences between SQL Server and Informix. Note that proprietary or nonstandard SQL Server syntax and features usually involve extra effort to convert to Informix.

4.2.1 SELECT statements

Typically, SQL Server SELECT statements do not require many modifications when converting to Informix. However, you need to be aware of a few differences during a conversion effort.

Unqualified columns

Both SQL Server and Informix permit the use of an unqualified column wildcard (*) alongside other elements in the SELECT clause list. Therefore, the following query is valid in both DBMSs:

```
SELECT *, e.* FROM booktab e, brands j WHERE e.brand_id = j.brand_id
```

Unspecified FROM clause

In SQL Server, the FROM clause of a SELECT statement is required except when the SELECT list contains only constants, variables, and arithmetic expressions (no column names), for example:

```
SELECT (4 * 5)
SELECT getdate()
```

In Informix, you must always specify the FROM clause as part of a SELECT statement. Use the SYSTABLES catalog table when a table or view name is not applicable. This view contains a single row. The previous statements can be translated to these statements:

```
SELECT (4 * 5) FROM systables WHERE tabid = 1
SELECT CURRENT TIMESTAMP FROM systables WHERE tabid = 1
```

Note that tabid = 1 is the systables table definition entry of itself.

Variable assignments

SQL Server syntax for queries involving assignment to variables differs from the Informix syntax. Consider, for example, this SQL Server statement:

```
SELECT @v_max=MAX(basic_salary) FROM employee
```

That statement corresponds to this Informix SELECT INTO statement:

```
SELECT MAX(basic_salary) INTO v_max FROM employee
```

However, the SELECT INTO clause in Informix is only supported in a Stored Procedure Language (SPL) routine or an ESQL/C program.

You must give special attention to SELECT INTO statements in Informix, because if the number of rows returned by the query is greater than one, Informix will raise an exception. The default behavior of SQL Server is to use the last row of the result set. When converting such statements to Informix, you must use the FETCH FIRST 1 clause, for example:

```
SELECT FIRST 1 book_id as v_c1 FROM booktab
```

COMPUTE

By including the COMPUTE clause in a SELECT statement, SQL Server generates totals that appear as additional summary columns at the end of the result set. When used in conjunction with the BY clause, control-breaks and subtotals are generated in the result set. For instance, the SQL Server query in Example 4-50 generates an additional summary column for each department, consisting of the sum of employee salaries for that department.

Example 4-50 Query generates additional summary column in SQL Server

```
SELECT emp_id, basic_salary  
FROM employee order by emp_id COMPUTE sum(basic_salary) BY emp_id
```

To convert this functionality to Informix, you can use the UNION ALL of multiple SELECT statements. Example 4-51 shows how you can convert the previous SQL Server query to Informix.

Example 4-51 Equivalent Informix summary generation using UNION ALL

```
SELECT TO_CHAR(emp_id) emp_id, basic_salary FROM employee  
      UNION ALL  
SELECT TO_CHAR(emp_id)||'Sum' emp_id, SUM(basic_salary) basic_salary  
      FROM employee GROUP BY emp_id  
      UNION ALL  
SELECT 'Sum' emp_id, SUM(basic_salary) basic_salary  
      FROM employee ORDER BY emp_id
```

Column and table aliases

In SQL Server and Informix, you can rename column and table names in a SELECT statement locally for that statement, although with separate syntax.

Example 4-52 shows an SQL Server SELECT statement that uses column alias names and table correlation names.

Example 4-52 Column aliases in SQL Server

```
SELECT DepartmentNo = d.dept_id,  
      EmployeeNo = e.emp_id,  
      EmployeeName = e.firstname + ' ' + e.middlename  
FROM departments d INNER JOIN employee e ON d.dept_id = e.dept_id
```

Informix syntax uses the AS keyword to define column aliases and table correlation names, as shown in Example 4-53 on page 133.

Example 4-53 Column aliases with the AS clause in Informix

```
SELECT d.dept_id AS DepartmentNo,  
       e.emp_id AS EmployeeNo,  
       e.firstname || ' ' || e.middlename AS EmployeeName  
FROM   departments AS d  
INNER JOIN employee AS e ON d.dept_id = e.dept_id
```

Alternatively, you can also omit the AS keyword, as shown in Example 4-54.

Example 4-54 Column aliases without the AS clause in Informix

```
SELECT d.dept_id DepartmentNo,  
       e.emp_id EmployeeNo,  
       e.firstname || ' ' || e.middlename EmployeeName  
FROM   departments d INNER JOIN employee e ON d.dept_id = e.dept_id
```

SELECT INTO

SQL Server's SELECT INTO statement differs completely from Informix's SELECT INTO statement. SQL Server's SELECT INTO statement creates a new table and is equivalent to a CREATE TABLE statement followed by an INSERT statement in Informix. Thus, the following query in SQL Server is a typical example:

```
SELECT * INTO t2 FROM t1
```

The previous example is equivalent to the Informix SQL statements that are shown in Example 4-55.

Example 4-55 Example of SELECT ... INTO in Informix

```
-- define all columns in t1 for t2 between ()  
CREATE TABLE t2 (col1 int, ...);  
INSERT INTO t2 SELECT t1.* FROM t1
```

However, in Informix, you can define a TEMP table using the SELECT statement:

```
SELECT * FROM t1 INTO TEMP t2
```

Case sensitivity

Both SQL Server and Informix support case-sensitive environments, where object identifiers and character strings can contain mixed upper and lower case characters; however, both products achieve this functionality differently.

In SQL Server, case sensitivity is determined by the environment settings. The environment can be either case sensitive or case insensitive. In most cases, operations that work in a case-sensitive environment will also work in a case-insensitive environment, but not the converse. For example, the object identifier (table name) in the SQL statement `SELECT * FROM mytable` is equivalent to the SQL statement `SELECT * FROM MYTABLE` in a case-insensitive environment, but not in a case-sensitive environment. The result of comparing character strings in table columns or variables is also determined by the environment settings.

The following query can help determine whether you are operating in a case-sensitive or case-insensitive environment in SQL Server:

```
SELECT CASE WHEN ('A' = 'a') THEN -1 ELSE 1 END
```

In a case-insensitive environment, this query returns the value -1; otherwise, it returns the value 1. In SQL Server, case sensitivity is controlled at multiple levels. Each SQL Server instance has a default collation setting, which determines the character set and sort order of characters. Databases can override the instance-level collation settings.

In Informix, all database object identifiers (tables, views, columns, and so on) are stored in the catalog tables in lowercase characters, unless they are explicitly delimited upon object creation. If a delimited identifier is used to create the object, the exact case of the identifier is stored in the catalog tables. An identifier, such as a column name or table name, is treated as case insensitive when used in an SQL statement unless it is explicitly delimited. For example, assume that the following statements are issued in Informix:

```
CREATE TABLE MyTable (id INTEGER)
CREATE TABLE "YourTable" (id INTEGER)
```

The following two statements are therefore equivalent and execute without any errors both in Informix and in case-insensitive SQL Server environments:

```
SELECT * FROM MyTable
SELECT * FROM MYTABLE
```

However, both of the following statements will succeed in a case-insensitive SQL Server environment whereas both statements will fail with an error in Informix:

```
SELECT * FROM "YourTable"          -- fails with syntax error 201
SELECT * FROM YourTable             -- error 206, table not found
```

Because of the table definition, the second `SELECT` statement will always fail in Informix. To overcome the first error shown, you must set the environment variable `DELIMIDENT` (assume UNIX korn shell):

```
export DELIMIDENT=Y
```

The DELIMIDENT environment variable specifies that strings enclosed between double quotation (") marks are delimited database identifiers. DELIMIDENT is also supported on client systems, where it can be set to Y, to N, or to no setting.

- ▶ *Y* specifies that client applications must use a single quotation mark (') to delimit character strings, and they must use double quotation marks (") only around delimited SQL identifiers, which can support a larger character set than is valid in undelimited identifiers. Letters within delimited strings or delimited identifiers are case-sensitive, which is the default value for an Object Linking and Embedding (OLE) DB and .NET.
- ▶ *N* specifies that client applications can use double quotation marks (") or a single quotation mark (') to delimit character strings, but not to delimit SQL identifiers. If Informix encounters a string delimited by double or single quotation marks in a context where an SQL identifier is required, it issues an error. An owner name that qualifies an SQL identifier can be delimited by single quotation marks ('). You must use a pair of the same quotation marks to delimit a character string.

Therefore, if you set the DELIMIDENT variable to Y, only the following statement will succeed in Informix:

```
SELECT * FROM "YourTable"
```

Informix strings are also case sensitive. For example, the string "daTAbase" is not the same as the string "DATABASE". Case sensitivity must be taken into account when comparing strings to ensure consistent results. One strategy for comparing strings involves always performing string comparisons using the uppercase or lowercase representation of the string, using the UPPER() and LOWER() Informix functions. Additionally, views constructed over base tables can present the data in uppercase or lowercase. Application users then perform all operations against the views so that case does not become an issue.

Joins

SQL Server supports two join syntaxes. The first join syntax is a proprietary syntax, which is not supported by Informix. This proprietary syntax uses the *= and =* operators in the WHERE clause to specify right and left outer joins, and this join syntax is now only supported for backward compatibility. The second join syntax is the ANSI-style syntax.

The Informix syntax for joins is ANSI-style, with the operators:

- ▶ INNER
- ▶ LEFT [OUTER]
- ▶ RIGHT [OUTER]
- ▶ FULL [OUTER]

The ANSI join operators also follow the ANSI definitions for join behavior. The ANSI-style join syntax helps to avoid ambiguous interpretation when other conditions are specified in the WHERE clause.

Inner joins

The SQL Server syntax for inner joins is similar to the Informix syntax. Example 4-56 shows what the syntax for an inner join looks like in both SQL Server and Informix.

Example 4-56 Example of INNER JOINS in Informix and SQL Server

```
SELECT r.title, a.name
FROM authors a
INNER JOIN booktab r
ON a.author_id = r.author_id
```

Outer joins

SQL Server supports the use of proprietary outer join operators (*= and =*) for backward compatibility. Table 4-4 shows how this proprietary syntax can be mapped to Informix supported syntax.

Table 4-4 Mapping proprietary outer join syntax to Informix

SQL Server	Informix
SELECT R.book_no, R.title, A.name FROM booktab R, authors A WHERE R.author_id *= A.author_id	SELECT R.book_no, R.title, A.name FROM booktab R LEFT OUTER JOIN authors A ON A.author_id = R.author_id
SELECT COUNT(R.book_id) amount, A.name FROM booktab R, authors A WHERE R.author_id =* A.author_id GROUP BY A.name ORDER BY amount desc	SELECT COUNT(R.book_id) AS amount, A.name FROM booktab R RIGHT OUTER JOIN authors A ON R.author_id = A.author_id GROUP BY A.name ORDER BY amount DESC

If the ANSI style outer join syntax is already used, it is unlikely that you will need to make changes during a conversion to Informix.

Cross join

A *cross join* (also known as a *cartesian product*) produces every possible join permutation of the selected columns. The example in Table 4-5 translates to “every job is available in all departments”. In Informix, cross joins are specified by not including any join conditions in the `WHERE` clause of the `SELECT` statement.

Table 4-5 Mapping of cross join syntax to Informix

SQL Server	Informix
SELECT job_desc, dept_code FROM jobs CROSS JOIN departments	SELECT job_desc, dept_code FROM jobs, departments

Set operators (UNION, EXCEPT, and INTERSECT)

The `UNION` and `UNION ALL` operators in both SQL Server and Informix have similar functionality and conditions of use. However, Informix does not support the `EXCEPT` and `INTERSECT` operators.

Example 4-57 shows how an SQL statement containing the `EXCEPT` operator in SQL Server can be modified to provide similar functionality in Informix.

Example 4-57 Example of `EXCEPT` operator in SQL Server and Informix

```
-- SQL Server Example:  
SELECT book_id  
FROM booktab  
EXCEPT  
SELECT book_id  
FROM book_history  
  
-- Informix Example:  
SELECT DISTINCT book_id  
FROM booktab where book_id NOT IN  
(SELECT book_id FROM book_history);
```

Example 4-58 shows how an SQL statement containing the `INTERSECT` operator in SQL Server can be modified to provide similar functionality in Informix.

Example 4-58 Example of `INTERSECT` operator in SQL Server and Informix

```
-- SQL Server Example:  
SELECT book_id  
FROM booktab  
INTERSECT  
SELECT book_id  
FROM book_history
```

```
-- Informix Example:  
SELECT DISTINCT book_id  
FROM booktab where book_id IN  
(SELECT book_id FROM book_history);
```

NULL handling

SQL Server and Informix both support the use of NULL values; however, differences exist in the ways that they are evaluated.

In both SQL Server and Informix, NULL is interpreted as a synonym for “unknown”. Therefore, you cannot write expressions that test for nullability using the equals (=) operator. For instance, the following expression is not provided for in the ANSI SQL standard and will return an error:

```
SELECT ... WHERE empno = NULL
```

Expressions that test for nullability must use the IS [NOT] NULL clause. You can rewrite the previous expression correctly:

```
SELECT ... WHERE empno IS NULL
```

In SQL Server, the ANSI NULL behavior is set by default. However, if the ANSI NULL option is not set, SQL Server interprets equality and inequality comparisons involving a variable, such as the expression `someColumn = @someVariable`, in a nonstandard way. In particular, when both `someColumn` and `@someVariable` are NULL, the expression will evaluate to true in SQL Server, but will evaluate to NULL (as per the SQL standard) in Informix. Additionally, the result of concatenation and arithmetic operations (for example, `NULL + 3`, `'MAR' concat NULL`) differs between SQL Server and Informix.

Ordering and grouping

Both SQL Server and Informix use similar syntax for the ORDER BY and GROUP BY clauses. Further, the functionality of the available options is similar. For example, both products display all NULL values at the beginning of the result set when the ORDER BY or GROUP BY clause is included in a SELECT statement. Both products also allow ordering by integers that represent column positions rather than column names.

Although both Informix and SQL Server support expressions in their ORDER BY clause, SQL Server does not allow expressions in its GROUP BY clause. Informix allows you to order by a substring instead of by the entire length of a character, BYTE, or TEXT column, or instead of an expression returning a character string. For example, Example 4-59 on page 139 shows two SELECT statements, which are valid in Informix but not in SQL Server.

Example 4-59 Valid SELECT statements in Informix only

```
SELECT basic_salary - 100 FROM employee  
GROUP BY 1;
```

```
SELECT * FROM brands ORDER BY brand_name[6,9];
```

One major difference between SQL Server and Informix is in the aggregate functions that SQL Server provides for the GROUP BY clause. SQL Server allows you to GROUP BY the business intelligence functions ROLLUP, CUBE, and GROUPING SETS. Informix has no equivalents. You can convert SQL statements using the GROUP BY GROUPING SETS clause by using the GROUP BY clause in Informix.

Example 4-60 provides an example of converting an SQL statement in SQL Server that uses the GROUPING SETS option of the GROUP BY clause to the equivalent statement in Informix.

Example 4-60 Informix equivalent of the GROUPING SETS option in the GROUP BY clause

```
-- SQL Server:  
SELECT emp_id, hire_date, SUM(basic_salary)  
FROM employee  
GROUP BY GROUPING SETS ((emp_id), (hire_date))  
  
-- Informix Example:  
SELECT emp_id, today - 365 hire_date, SUM(basic_salary)  
FROM employee  
GROUP BY emp_id  
UNION ALL  
SELECT 0 as emp_id, hire_date, SUM(basic_salary)  
FROM employee  
GROUP BY hire_date
```

You must rewrite SQL statements using the ROLLUP and CUBE aggregate functions or make extensive use of SPL routines to handle the aggregation.

Hierarchical

In SQL Server, you can define tables to hold hierarchical data by using one of the following two methods:

- Define a column that references itself, as shown in Example 4-61.

Example 4-61 Create a hierarchical table using the REFERENCES clause

```
CREATE TABLE emp1
  (empid  INTEGER NOT NULL PRIMARY KEY,
   name   VARCHAR(10),
   salary DECIMAL(9, 2),
   mgrid  INTEGER
   REFERENCES emp1 (empid));
```

- Define a column with the built-in data type `hierarchyid`, as shown in Example 4-62.

Example 4-62 Create a hierarchical table using the hierarchyid data type

```
CREATE TABLE emp1
  (empid  hierarchyid,
   name   VARCHAR(10),
   salary DECIMAL(9, 2),
   mgrid  INTEGER);
```

In SQL Server, you can select rows in tables that contain hierarchical data in a hierarchical order in the following ways:

- Creating temporary result sets using recursive Common Table Expressions (CTEs) and selecting from the CTE. Example 4-63 creates a CTE named `org` and recursively retrieves all managers and their employees by repeatedly calling itself.

Example 4-63 Create a CTE in SQL Server that retrieves hierarchical data

```
WITH org(mgrid, empid) AS
(
  SELECT mgrid, empid
  FROM emp1 e
  WHERE mgrid IS NULL
  UNION ALL
  SELECT e.mgrid, e.mgrid
  FROM emp1 e
  INNER JOIN org d
  ON e.mgrid = d.empid
```



```
)  
SELECT mgrid, empid  
FROM org;
```

- For a column of a table defined with the `hierarchyid` data type, SQL Server provides supporting functions to use with this data type in SQL statements. Functions include finding ancestors or descendants, and also getting root and level information.

Informix also provides two methods to work with hierarchical data: using the `CONNECT BY` clause and using the `NODE DataBlade` module.

CONNECT BY

The `SELECT` statement of Informix supports `START WITH ... CONNECT BY` syntax for recursively querying a table in which a hierarchy of parent-child relationships exists.

The `CONNECT BY` clause specifies conditions for performing recursive operations in hierarchical queries. If you include the `START WITH` clause, the search condition that it specifies is applied in producing the first intermediate result set for the hierarchical query. This intermediate result set consists of the rows of the table specified in the `FROM` clause for which the `START WITH` condition is true.

If the `START WITH` clause is omitted, no `START WITH` condition is available as a filter, and the first intermediate result set is the entire set of rows in the table that the `FROM` clause specifies.

The `CONNECT BY` clause produces successive intermediate result sets by applying the `CONNECT BY` search condition until this recursive process terminates when an iteration yields an empty result set. Continuing our previous example, in Example 4-64, the row that satisfies the condition `name = 'Goyal'` becomes the root for beginning the recursive operations of the `CONNECT BY` clause.

Example 4-64 Using the `CONNECT BY` clause in Informix to retrieve hierarchical data

```
SELECT name, empid, mgrid  
FROM emp  
START WITH name = 'Goyal'  
CONNECT BY PRIOR empid = mgrid
```

The query returns all employees, including managers, who work under “Goyal”.

NODE DataBlade module

Informix provides a node data type to model hierarchical relationships. The data type and the supported functions, ancestor, depth, getparent, getmember, length, and so forth, are packaged as the Node DataBlade Module, Version 2.0 in Informix 11. You need to register this DataBlade module in your database to use the node data type.

The node data type is an OPAQUE data type that models the tree structure instead of flattening hierarchies to relations. Each value represents the edge of the hierarchy, not simply a number or string. Therefore, when you increment node 1.9, you get 1.10, and not the numerical increment value of 1.91.

See *Informix Dynamic Server 11: Advanced Functionality for Modern Business*, SG24-7465, for more information about how to use the Node DataBlade module to migrate hierarchical data to node type data in Informix.

TOP n

SQL Server's TOP *n* clause in a SELECT statement can be translated to Informix using the FIRST *n* ROWS clause, as shown in Example 4-65.

Example 4-65 Retrieving the first five records of a query in Informix

```
SELECT FIRST 5 *  
FROM booktab  
ORDER BY price ASC
```

There is no equivalent in Informix for the TOP *n* PERCENT clause. Converting this functionality requires customized cursor logic.

Hints

Informix offers *optimizer directives*, which are similar to SQL Server optimizer hints. This feature provides the query developer the flexibility to direct the optimizer to follow specific paths, rather than choosing a plan through its analysis.

A query is processed in two steps. First, it is optimized and compiled to produce a query plan. Second, it is executed to produce results. Optimizer directives are a method for influencing the choice of the optimizer in the creation of the plan.

Optimizer directives address the following key considerations and issues:

- ▶ Reduced time for correcting performance problems. Rewriting queries can be time consuming, and optimizer directives provide a quick way to alter a plan.
- ▶ Competitive pressure. Users accustomed to this function in SQL Server were looking for it in Informix.

- Product development tuning. Optimizer directives allow product development to alter plans dynamically rather than through code changes to evaluate the effect of various optimization techniques.

The behavior of the Informix implementation is compatible with SQL Server optimizer hints; however, the syntax differs. In Informix, all directives are written as a comment whose first character is a plus sign (+). However, the syntax for hints in SQL Server differs significantly. Furthermore, you can provide hints in SQL Server in several formats; for example, you can provide JOIN hints using the OPTION clause or you can specify a hint within the JOIN clause.

Example 4-66 provides an example of two equivalent SQL statements in SQL Server that tell the optimizer to use a HASH JOIN between the tables book_history and brands.

Example 4-66 Hint to use HASH JOIN on SELECT statements in SQL Server

```
SELECT * from book_history r, brands b
where r.brand_id= b.brand_id
OPTION (HASH JOIN);

SELECT * from book_history r
LEFT HASH JOIN brands b ON r.brand_id= b.brand_id
```

Example 4-67 depicts the Informix equivalent of the hint in SQL Server.

Example 4-67 Directives to use HASH JOIN on SELECT statement in Informix

```
SELECT --+ USE_HASH (book_history)
*
FROM book_history r, brands b
WHERE r.brand_id = b.brand_id
```

Informix supports the notion of negative directives, whereas SQL Server only supports direct hints. A directive that tells the optimizer what to avoid, rather than what to choose, is unique to Informix. The query result can be realized by writing a directive to avoid certain actions known to cause performance issues, but to allow any new indexes or table attributes to be explored by the optimizer as they are added over the life of the table and query. This design allows a DBA to continue to add indexes to tables and not have to rewrite directives.

Additionally, Informix supports full recognition of directives with SET EXPLAIN output. Informix will highlight semantic and syntactic errors. Optimizer directives support control in the following areas of the optimization process:

- Access methods: Index versus scans
- Join methods: Forcing hash joins, nested loop joins

- Join order: Specify in which order the tables are joined
- Goal: Specify first rows or all rows (response time versus throughput)

For more information, see the *IBM Informix Guide to SQL: Syntax, v11.50*, SC23-7751. Or, see *IBM Informix Guide to SQL: Reference*, G229-6374, at this website:

<http://www.oninit.com/manual/informix/1150/GuidetoSQLReference.pdf>

External optimizer directives

External optimizer directives give the DBA the ability to specify query directives and save them in the database. These directives are applied automatically to subsequent instances of the same query.

The SAVE EXTERNAL DIRECTIVES statement associates one or more optimizer directives with a query, and stores a record of this association in the sysdirectives system catalog table, for subsequent use with queries that match the specified query string. The query string must be an exact match that includes case and white space positioning. This statement establishes an association between the list of optimizer directives and the text of a query, but it does not execute the specified query. Only the DBA or user Informix can execute SAVE EXTERNAL DIRECTIVES.

Example 4-68 depicts associating AVOID_INDEX and FULL directives with the specified query.

Example 4-68 External directives in Informix

```
SAVE EXTERNAL DIRECTIVES
  /*+ AVOID_INDEX (table1 index1)*/, /*+ FULL(table1) */
  ACTIVE FOR
    SELECT col1, col2 FROM table1, table2
    WHERE table1.col1 = table2.col1
```

These directives are applied automatically to subsequent instances of the same query. You must include one of the ACTIVE, INACTIVE, or TEST ONLY keyword options to enable, disable, or restrict the scope of external directives. When external directives are enabled and the sysdirectives system catalog table is not empty, Informix compares every query with the query text of every ACTIVE external directive, and for queries executed by the DBA or user Informix, with every TEST ONLY external directive.

External directives are ignored if the EXT_DIRECTIVES parameter is set to 0 in the ONCONFIG file. In addition, the client system can disable this feature for its current session by setting the IFX_EXTDIRECTIVES environment variable to 0. If an external directive has been applied to a query, output from the SET EXPLAIN statement indicates “EXTERNAL DIRECTIVES IN EFFECT” for that

query. Any in-line directive is ignored by the optimizer when the external directives are applied to a query that matches the SELECT statement.

Like in-line optimizer directives that are embedded within a query, external directives can improve performance in certain queries for which the default behavior of the query optimizer is not satisfactory. Unlike in-line directives, external directives can be applied without revising or recompiling existing applications.

Cursors

SQL Server supports all American National Standards Institute (ANSI)-style cursors: static, dynamic, forward only, and keyset-driven. SQL Server includes support for INSENSITIVE and SCROLL cursor behavior and for all fetch options (FIRST, LAST, NEXT, PRIOR, RELATIVE, and ABSOLUTE). Cursor support is available through the following interfaces: ActiveX Data Objects (ADO), Object Linking and Embedding (OLE) DB, Open Database Connectivity (ODBC), DB-Library, and T-SQL.

In Informix, you can declare a cursor as a sequential cursor (the default), a scroll cursor (by using the SCROLL keyword), or a hold cursor (by using the WITH HOLD keywords). The SCROLL and WITH HOLD keywords are not mutually exclusive.

If you use only the CURSOR keyword, you create a sequential cursor, which can fetch only the next row in sequence from the active set. The sequential cursor can read through the active set only one time each time that it is opened. If you are using a sequential cursor for a Select cursor, on each execution of the FETCH statement, Informix returns the contents of the current row and locates the next row in the active set.

You can use the SCROLL keyword to create a scroll cursor, which can fetch rows of the active set in any sequence. Informix retains the active set of the cursor as a temporary table until the cursor is closed. You can fetch the first, last, or any intermediate rows of the active set, as well as fetch rows repeatedly, without having to close and reopen the cursor.

Transaction behavior

The default behavior in SQL Server is to have cursors remain open after a COMMIT or ROLLBACK statement is issued. This behavior strays from the ANSI SQL standard, although it is possible to configure SQL Server to use the ANSI SQL standard behavior of closing open cursors after a COMMIT or ROLLBACK statement is issued.

Informix's default behavior follows the ANSI SQL standard of closing open cursors whenever a COMMIT or ROLLBACK statement is issued. However,

cursors that are declared with the WITH HOLD option remain open after a COMMIT statement is issued. In Informix, all open cursors are closed when a ROLLBACK statement is issued.

Important: In Informix, after a COMMIT statement has been issued, a cursor declared with the WITH HOLD statement remains open and is positioned before the next logical row of the results table. Additionally, all locks are released except for locks protecting the current cursor position.

DEALLOCATE

SQL Server provides a cursor cleanup command called DEALLOCATE, which is typically used as a performance benefit to eliminate the cost of declaring a new cursor after the last reference to the cursor is deallocated. The DEALLOCATE command allows a cursor to be closed and reopened without re-declaring it.

Informix does not have a similar DEALLOCATE command. Cursors must be explicitly closed when they are no longer needed, and re-declared if and when they are needed again.

Converting cursor attributes

SQL Server supports cursor attributes to obtain information about the current status of a cursor. Use SQLCODE/SLSTATE values to obtain the analogous information in Informix. This section shows how to match SQL Server cursor attribute functions with Informix SQLCODE/SQLSTATE values.

@@CURSOR_ROWS

In SQL Server, the @@CURSOR_ROWS function returns the number of qualifying rows from the previous cursor declaration. In Informix, you can use a local (counter) variable to store this information after each FETCH operation from the cursor.

Example 4-69 shows implementing this function in Informix. In the example, a local variable is declared to hold the number of fetched rows [1] and is incremented each time that a new row is fetched [2].

Example 4-69 Informix cursor that counts the number of fetched rows

```
CREATE PROCEDURE tt(val int)
    DEFINE v_cursor_rows INTEGER;          -- [1]
    DEFINE my_ename INTEGER;
    DEFINE my_deptno VARCHAR(40);

    LET v_cursor_rows = 0;
    prepare p1 from 'SELECT author_id, name FROM authors';
    DECLARE c1 CURSOR FOR p1;
```

```

        LOOP
            FETCH c1 INTO my_ename, my_deptno;
            LET v_cursor_rows = v_cursor_rows + 1;           -- [2]
        END LOOP;

END PROCEDURE;

```

4.2.2 INSERT statements

You cannot translate certain INSERT statements directly from SQL Server to Informix because of the use of IDENTITY or ROWVERSION columns. For example, the following statements are valid in SQL Server, but not in Informix:

```

CREATE TABLE t1(c1 INT, c2 INT IDENTITY)
INSERT INTO t1 VALUES(1)

```

The second statement is not valid in Informix, because the number of columns in t1 does not match the number of columns that is specified in the VALUES clause. Informix requires that the columns for which the values are intended are explicitly stated, as shown in the following valid translation:

```

INSERT INTO t1(c1) VALUES(1)

```

DEFAULT VALUES clause

In SQL Server, you can specify the DEFAULT VALUES clause in an INSERT statement to force the new row to contain the default values defined for each column. For example, in SQL Server, assume the table definition that is shown Example 4-70.

Example 4-70 CREATE TABLE in SQL Server

```

CREATE TABLE defaulttab (
    c1 INT IDENTITY,
    c2 VARCHAR(30)
    CONSTRAINT default_name DEFAULT ('column default'),
    c3 INT NULL)

```

The following INSERT statement inserts the values (1, column default, NULL) into a new row, for columns c1, c2, and c3.

```

INSERT INTO defaulttab DEFAULT VALUES

```

Informix does not support the DEFAULT keyword when inserting data. Informix tries to use a column's default value, if one exists, to supply a value for column

not specified in an INSERT statement. Assume the equivalent table definition in Informix, as shown in Example 4-71.

Example 4-71 CREATE TABLE in Informix

```
CREATE TABLE defaulttab (  
    c1 serial,  
    c2 VARCHAR(30) DEFAULT 'column default',  
    c3 INT)
```

Translate the previous INSERT statement in this way in order to preserve the same functionality:

```
INSERT INTO defaulttab(c1) VALUES (0)
```

Column c2 has the default value column default, and the column c3 is NULL.

Important: Informix attempts to use NULL as a default, if no default value is defined for a column. If a column is defined as NOT NULL and does not have a default value specified, an INSERT statement using the default keyword to supply that column's value will fail.

In Informix, a default value can be a literal character string that you define or one of the following SQL constant expressions:

- ▶ CURRENT returns the current time and date from the system clock.
- ▶ CURRENT_ROLE returns the name of the role, if any, whose privileges are enabled for the current user.
- ▶ DEFAULT_ROLE returns the name of the role, if any, that is the default role for the current user.
- ▶ DBSERVERNAME returns the name of the current Informix instance.
- ▶ SITENAME is a synonym for DBSERVERNAME.
- ▶ SYSDATE reads the DATETIME value from the system clock like the CURRENT operator, but has a separate default precision.
- ▶ TODAY returns the current calendar date from the system clock.
- ▶ USER returns the login name (also called the *authorization identifier*) of the current user.

INSERT INTO <table_name> EXECUTE <procedure_name>

In SQL Server, you can combine an INSERT statement with an EXECUTE statement to invoke stored procedures on the same server or a remote server. The procedure must return data with the SELECT statement. The procedure on the remote server is executed, and the result sets are returned to the local server

and loaded into the table on the local server. For example, Example 4-72 creates a stored procedure in SQL Server and then invokes it as part of an INSERT statement. The duplicate behavior in Informix, also shown, uses an INSERT statement with a sub-SELECT statement.

Example 4-72 INSERT INTO equivalents in SQL Server and Informix

```
-- SQL Server Example:
CREATE PROCEDURE proc1
AS
SELECT * FROM table1
GO

INSERT INTO table1 EXECUTE proc1
GO

--Informix Example:
INSERT INTO table1 SELECT * FROM t1
```

Because no single technique exists to convert this feature to Informix, the actual technique that is used will depend on how and why this functionality is being used in your application.

4.2.3 UPDATE statements

SQL Server allows you to specify more than one table in the FROM clause of UPDATE statements. However, non-determinism might make the translation complex. Non-determinism arises when multiple rows match the join condition of the tables that are specified in the UPDATE statement. For example, refer to the following SQL Server UPDATE statement:

```
UPDATE t1 SET c1=t2.c1 FROM t1,t2 WHERE t1.c2=t2.c2
```

In this example, non-determinism arises when there are multiple matching rows in t2, meaning that any of the c1 values of these rows can be selected to do the update. If there is, at most, a single matching row in t2, the following statement is the equivalent Informix statement:

```
UPDATE t1 SET c1 = (SELECT DISTINCT t2.c1 FROM t2 WHERE t1.c2 = t2.c2)
WHERE EXISTS(SELECT * FROM t2 WHERE t1.c2 = t2.c2)
```

Important: The only way to correctly simulate this behavior is by using a cursor-based implementation. Usually, such non-determinism is unintentional and needs to be corrected during a conversion to Informix.

UPDATE with a JOIN

SQL Server T-SQL provides the ability to perform an UPDATE or DELETE based on join conditions on another table or tables. Specifically, SQL Server T-SQL allows the use of the FROM clause within the UPDATE and DELETE statements, as shown in Example 4-73.

Example 4-73 UPDATE with JOIN on another table in SQL Server

```
UPDATE titles
  SET total_sales = total_sales + qty
  FROM titles t,
       sales s,
       salesdetail d
  WHERE t.title_id = d.title_id
        AND d.stor_id = s.stor_id
        AND s.date IN (SELECT MAX(sales.date) FROM sales)
```

Informix does not allow the use of the FROM clause in an UPDATE or DELETE. You have two possible work-arounds for these situations:

- ▶ Perform a sub-select to update the columns.
- ▶ Create a stored procedure and use a cursor to perform the update.

Example 4-74 illustrates the work-arounds.

Example 4-74 UPDATE with FROM clause comparison in Informix

```
-- SQL Server Example:
UPDATE t1
  SET t1.col1 = t2.col1,
      t1.col2 = t2.col2
  FROM t1,
       t2
  WHERE t1.col3 = t2.col4

--Informix Example:
UPDATE t1
SET t1.col1, t1.col2 =
  (SELECT DISTINCT t2.col1, t2.col2
   FROM t2
   WHERE t2.col3 = t1.col4)
WHERE t1.col3 IN (SELECT col4 FROM t2);
```

In certain cases, the performance suffers by using sub-selects in this manner. (Note that Informix 7.30 and later versions include significant subquery performance enhancements so that this situation might be less of an issue. You need to test and confirm using sub-selects in this manner.)

Currently, the preferred alternative is to rewrite the statement using a cursor within a stored procedure, as seen in Example 4-75.

Example 4-75 UPDATE with FROM clause using a stored procedure in Informix

```
FOREACH
  SELECT t2.col1,
         t2.col2,
         t2.col4
  INTO var1,
        var2,
        var3
  FROM t1,
        t2
  WHERE t1.col3 = t2.col4;

  UPDATE t1
  SET col1 = var1,
      col2 = var2
  WHERE col3 = var3
END FOREACH;
```

4.2.4 DELETE statements

SQL Server allows more than one table to be specified in the FROM clause of DELETE statements. For example, the following DELETE statement is valid in SQL Server, but not in Informix:

```
DELETE t1 FROM t1, t2 WHERE t1.c1=t2.c1
```

Example 4-76 shows how you can translate the previous statement to Informix by using a combination of a DELETE statement and SELECT statement.

Example 4-76 DELETE FROM clause that references an outside table in Informix

```
DELETE FROM t1
WHERE EXISTS (SELECT * FROM t2 WHERE t1.c1 = t2.c1)
```

4.2.5 TRUNCATE statement

SQL Server's TRUNCATE TABLE statement provides a quick way to delete rows in a table, for example, the following statement deletes all the rows from the booktab table:

```
TRUNCATE TABLE booktab
```

Informix also provides an equivalent command to perform the same operation. However, by default, all of the partition extents that had been allocated to the specified table and to its indexes are released after TRUNCATE successfully executes.

Alternatively, if it is your intention to keep the same storage space allocated to the same table for subsequently loaded data, you can specify the REUSE STORAGE keywords to prevent the space from being deallocated. The REUSE STORAGE option of TRUNCATE can make storage management more efficient in applications where the same table is periodically emptied and reloaded with new rows.

The following example truncates the employee table and free all extents except the first extent:

```
TRUNCATE TABLE employee DROP STORAGE;
```

The following example truncates the same table but only removes the actual data. All extents stay the same.

```
TRUNCATE TABLE state REUSE STORAGE;
```

Whether you specify DROP STORAGE or REUSE STORAGE, any out-of-row data values are released for all rows of the table when the TRUNCATE transaction is committed. Storage, which is occupied by any BLOB or CLOB values that become unreferenced in the TRUNCATE transaction, is also released.

Important: There is a minor syntactical difference between SQL Server and Informix: Informix does not require the keyword TABLE. The permissions needed for TRUNCATE TABLE depend on the existence of DELETE triggers on the table.

For tables with DELETE triggers, ALTER permission on the table is required and RESOURCE or DBA privileges are required, because, otherwise, the DELETE triggers will be bypassed. For tables without DELETE triggers, you need DELETE permission on the table and CONNECT privilege to the database.

Informix permits a TRUNCATE TABLE in a transaction, but it must be the last statement before COMMIT or ROLLBACK. Any other statement will generate an error. In an Informix non-ANSI database, TRUNCATE is run as a singleton statement. Outside any explicit transaction in MODE ANSI databases, COMMIT immediately.

4.2.6 MERGE statement

Both SQL Server and Informix use the MERGE statement to transfer data from a source table into a target table by combining UPDATE or DELETE operations with INSERT operations in a single SQL statement. Both DBMSs also use this statement to join the source and target tables, and then, they perform only UPDATE operations, only DELETE operations, or only INSERT operations on the target table.

As with regular SQL statements, SQL Server also supports the TOP *n* PERCENT clause in the MERGE statement. There is no equivalent in Informix for the clause. Converting this functionality requires customized cursor logic.

The default behavior for the WHEN NOT MATCHED [BY TARGET] option in both SQL Server and Informix is to insert records in the target table using records from the source table. However, SQL Server also provides the option WHEN NOT MATCHED BY SOURCE. This option can update or delete records in the target table, which do not exist in the source table.

Example 4-77 provides an example of using the WHEN NOT MATCHED BY SOURCE option of the MERGE statement. Specifically, any records in the target table (table *t*), which do not exist in the source table (table *s*), are deleted. The following table definitions are assumed in the example:

```
CREATE TABLE s(c1 INT);  
CREATE TABLE t(c1 INT, c2 INT, c3 INT);
```

Example 4-77 SQL Server WHEN NOT MATCHED BY SOURCE clause of MERGE

```
MERGE  
  t AS TARGET  
  USING s AS SOURCE  
  ON (TARGET.c1 = SOURCE.c1)  
  WHEN MATCHED THEN  
    UPDATE SET TARGET.c1 = SOURCE.c1+5  
  WHEN NOT MATCHED BY TARGET  
    THEN INSERT (c1, c2, c3)  
    VALUES (SOURCE.c1, 70, 70)  
  WHEN NOT MATCHED BY SOURCE THEN  
    DELETE;
```

The Informix equivalent of the above MERGE statement consists of two SQL statements, a MERGE statement that is followed by a DELETE statement. Example 4-78 on page 154 shows the two SQL statements.

```
MERGE
    INTO t
    USING s
    ON t.c1=s.c1
    WHEN MATCHED THEN
        UPDATE SET t.c1 = t.c1+5
    WHEN NOT MATCHED
        THEN INSERT (c1, c2, c3) values (s.c1, 70, 70);

DELETE FROM t
WHERE t.c1 NOT IN (SELECT s.c1 from s);
```

4.2.7 SQL in system catalog tables

In SQL Server, metadata is stored in the master database and is accessible using a special schema called INFORMATION_SCHEMA.

In Informix, each database contains its own metadata in a set of base tables and views called the *catalog*. The catalog contains information about the logical and physical structure of the database objects, object privileges, integrity information, and so on. All tables and views in the catalog begin with the *sys* prefix, for example, systables.

The Informix database catalog is automatically created when a database is created. The tables and views in the catalog belong to the root dbspace. The catalog is always created with the schema informix.

The Informix system catalog views cannot be modified using traditional SQL statements. They are automatically updated each time that an SQL data definition statement is executed.

Normal user access to the system catalog is read-only. Users with Connect or Resource privileges cannot alter the catalog, but they can access data in the system catalog tables on a read-only basis by using standard SELECT statements. Although user *informix* can modify most system catalog tables, you must not update, delete, or insert any rows in them. Modifying the content of system catalog tables can destroy the integrity of the database.

The following example shows accessing the Informix catalog table called systables:

```
SELECT tabname, tabid FROM systables WHERE tabid > 99
```

To validate system catalog tables, you can use the **oncheck -cc** command.

Due to the separate product architectures, querying the system for information differs in SQL Server and Informix.

4.3 Transact-SQL statements

Transact SQL (T-SQL) is SQL Server's extension to the ANSI SQL language. T-SQL is a dynamic database programming language. All SQL Server stored procedures, user-defined functions, triggers, and SQL statement batches are coded using T-SQL. The major client application focus for T-SQL is to provide a developer with an SQL-based 4GL programming environment rather than supporting only SQL batches.

The best mapping for T-SQL is the Stored Procedure Language (SPL) that is provided by the Informix. The SPL is invented as a development environment for implementing your SQL-based functionality, enriched with variables, flow and status control, cursor support, and dynamic statements. SPL provides an easy to use programming interface for implementing SQL-based applications. This code is precompiled and stored in the database server. Informix maintains the execution of the SP code, and it guarantees parallel and remote execution control. You can apply SQL-based access control in the database server.

You must rewrite T-SQL-based applications in SPL. Because both T-SQL and SPL are extensions of the same standard, many of the same features are available in both languages, although, sometimes with differing syntax. In this section, we describe the differences and required replacements of your T-SQL scripts based on examples. Table 4-6 shows, as a start, the equivalence between T-SQL and SPL elements and statements.

Table 4-6 Mapping common T-SQL programming constructs to Informix SPL

T-SQL	Informix SPL
DECLARE @varname datatype = defaultvalue	Define varname datatype DEFAULT defaultvalue;
SELECT @var1=value	LET var1 = value;
SELECT @var1=colname FROM table WHERE...	SELECT colname INTO var1 FROM table WHERE...;
SELECT @v1=col1,@v2=col2,@v3=col3 FROM table...	SELECT col1,col2,col3 INTO v1,v2,v3 FROM table...
WHILE expression BEGIN ... END	WHILE expression ... END WHILE;

T-SQL	Informix SPL
CONTINUE	CONTINUE
BREAK	EXIT
IF (...) BEGIN ... END ELSE ...	IF (...) THEN ... ELSE ... END IF;
EXECUTE ('INSERT INTO t1 VALUES(2)')	--dynamic SQL Let var='INSERT INTO t1 VALUES(2)' EXECUTE IMMEDIATE (var);
EXECUTE procname(parm1,parm2,...)	CALL procname(parm1,parm2,...);
EXECUTE @retval=procname(parm1,parm2,...)	CALL procname(parm1,parm2,...) returning retval;
RETURN <int_value>	RETURN <int_expr>;
GOTO <label>	GOTO <<label>>
PRINT	TRACE
@@FETCHSTATUS	FOREACH See the following sections for more details.
RAISERROR	RAISE EXCEPTION See the following sections for more details.
@@ERROR	ON EXCEPTION See the following sections for more details.
@@ROWCOUNT	DBINFO('sqlca.sqlerrd2'); See the following sections for more details.
@@TRANCOUNT	N/A See the following sections for more details.
@@VERSION	DBINFO('VERSION') See the following sections for more details.

4.3.1 Variables

Variables describe the containers that maintain values associated to a certain data type. Their current values depend on the code logic, the return values in SQL statements, and the executed code tree. Variables are visible in the current

object context, such as an SP, a trigger, or a T-SQL batch. In Informix SPL, variables can be declared as global and retain their values after execution of the current SP.

Variables must be declared before values can be assigned. There are simple value assignments, expression-based assignments, and assignments from SQL statements, such as SELECTS and nested SP calls. Table 4-7 shows examples of handling variables in T-SQL and Informix SP.

Table 4-7 Handling variables in T-SQL and Informix SPL

	T-SQL	Informix SPL
Variable definition	<pre>DEFINE @var Integer DEFINE @var1 varchar= 'A'</pre>	<pre>DECLARE var Integer; DECLARE GLOBAL VAR1 Integer Default 1; DECLARE name LIKE Authors.name;</pre>
Simple value assignment	<pre>SET @var = 1000 SET @var1= ' informix'</pre>	<pre>LET var = 1 ; LET var1 = 1000; LET name = 'Smith';</pre>
Simple value assignment using expressions	<pre>SET @var = 1000 + 1000 SET @var1= 'Informix' + 'Informix' SET @type = replace (data_type,'bit','integer')</pre>	<pre>LET var = 1 + 2000; LET cvar = 'Informix' ' Informix'; LET status= DBINFO ('sqlca.sqlerrd2');</pre>
Value assignment using SQL	<pre>SELECT @var1=@@version FETCH next from cursorname INTO @var EXECUTE @pr=avg_price 1</pre>	<pre>SELECT today INTO dtvar FROM systables WHERE tabid=1; SELECT author_id INTO name FROM dbo.authors CALL avg_price(1) RETURNING pr;</pre>

4.3.2 Conditional statements

Conditional statements allow the definition of alternative code trees. Depending on a result of a boolean expression, only a specific branch of the current statement tree is executed. T-SQL currently supports the IF and the CASE statements. Example 4-79 shows examples for both statement types.

Example 4-79 Conditional statements in T-SQL

```
--Case expression
SET @job_desc = CASE @job_id
    WHEN 1 THEN 'Secretary'
    WHEN 2 THEN 'Developer'
    WHEN 4 THEN 'Teamleader'
    ELSE 'CIO'
```

```

END

--if clause
IF @job_id = 1
BEGIN
SET @job_desc='Secretary'
END
ELSE
BEGIN
SET @job_desc='CIO'
END
PRINT @job_desc

```

The IF statement in comparison with T-SQL looks quite similar in the Informix SPL. The only difference is that in Informix SPL, the statement block marker, which starts with BEGIN and ends with END, is not required. In addition, you can expand the IF statement by using an ELIF clause. You can benefit from that clause to rewrite the CASE expression that is not supported in SPL. See Example 4-80 for further details about how to implement IF THEN ELSE logic and how to rewrite a case expression in Informix SPL.

Example 4-80 Conditional statements in Informix SPL

```

-- Implementing a simple CASE expression in Informix SPL
IF job_id=1 THEN
    LET job_desc='Secretary';
ELIF job_id=2 THEN
    LET job_desc='Developer';
ELIF job_id=4 THEN
    LET job_desc='Teamlead';
ELSE
    LET job_desc='CIO';
END IF;

--IF clause in Informix SPL
IF job_id = 1
THEN
    LET job_desc='Secretary';
ELSE
    LET job_desc='CIO';
END IF;

```

4.3.3 GOTO statements

Using GOTO statements is handy when you want to break the current code flow and branch to a specific code section for handling a certain condition. For example, if there is a code failure, branch to the code section that performs final cleanup of cursors, adjusts variable or return value settings, and raises the defined errors. The jump to place is marked by a label. A GOTO statement works with labels that define the points to where the GOTO refers. Example 4-81 demonstrates the references between a label and the GOTO in T-SQL code pieces.

Example 4-81 GOTO statement and labels in T-SQL

```
DECLARE @status int;
DECLARE @id int = 100;
DECLARE @cnt int = 100;

SELECT @cnt=count(*) FROM jobs WHERE job_id=@id
if ( @cnt = 0 ) BEGIN
    GOTO fail
END
ELSE BEGIN
    GOTO success
END

success:
SET @status = 1
PRINT @status
fail:
SET @status = 0
PRINT @status
```

Informix SPL also implements the GOTO clause and supports the definition of the labels for specifying where to jump. Be aware that the syntax of the label definition differs in these two languages. Compare the Informix SPL code in Example 4-82 with T-SQL code in Example 4-81.

Example 4-82 Equivalent implementation of the code piece in Informix SPL

```
CREATE PROCEDURE using_gotos ( ) RETURNING int

DEFINE status int;
DEFINE id int;
DEFINE cnt int;

LET id=100;
LET cnt=0;
```

```

SELECT count(*) INTO cnt FROM jobs WHERE job_id=id;

IF ( cnt = 0 )
THEN
    GOTO fail;
ELSE
    GOTO success;
END IF

<<success>>
LET status = 1;
RETURN status;

<<fail>>
LET status = 0;
RETURN status;

END PROCEDURE

```

4.3.4 Statements maintaining loops

The definition of a loop condition is required in SQL-based programming environments, especially for handling an open cursor. Depending on the error conditions or a certain end criteria, the loop continues, exits, or ends. The loop exit can be defined when the cursor has processed all rows from the retrieved result set for the query. T-SQL provides the **WHILE** statement for maintaining a loop. You can either loop until a certain boolean condition is met or until a conditional break is issued. See Example 4-83 for sample **WHILE** loop implementations in T-SQL.

Example 4-83 The WHILE statement in T-SQL

```

-- For N to M implementation
WHILE (@count < 100)
BEGIN
    SET @count= @count+1
END

-- Cursor handling in a while loop
DECLARE @id char(29)
DECLARE authors CURSOR FOR
SELECT author_id FROM dbo.authors
OPEN authors;
FETCH NEXT FROM authors into @id;
WHILE @@FETCH_STATUS = 0
    BEGIN
        FETCH NEXT FROM authors into @id;
    END

```

```
END;  
CLOSE authors;  
DEALLOCATE authors;
```

Informix SPL provides similar functionality in maintaining loops:

- ▶ **FOR statement:** Informix supports the FOR loop statement. You can define a start and end value for a variable as a range and a certain increment value. The current value of the variable is incremented by the specified value after every loop in the FOR statement block. After that, the new value is checked if it still meets the defined loop range. Conditional exits for the loop are supported.
- ▶ **WHILE statement:** Similar to T-SQL, Informix supports the WHILE statement in the SPL. The statement block that is enveloped by the WHILE clause and the END WHILE clause is executed again when a certain boolean expression is true. The WHILE loop terminates after the boolean expression becomes false or when a conditional exit is detected.
- ▶ **FOREACH statement:** Informix SQL has a special FOREACH loop statement that was introduced specifically for cursor handling. It automatically fetches all the rows returned by a certain query specified for FOREACH for further evaluation. Similar to T-SQL, you can implement conditional breaks in the FOREACH loops.

Example 4-84 provides Informix SPL loop statement examples.

Example 4-84 Loop implementations in Informix SPL

```
-- FOR clause  
FOR variable IN (1 TO 100 STEP 2)  
SELECT count (*) INTO cnt FROM jobs WHERE job=variable;  
IF cnt = 0  
THEN  
    EXIT FOR;  
END IF  
  
END FOR  
  
-- While clause  
SELECT count (*) INTO cnt FROM jobs WHERE job=variable;  
WHILE cnt > 0  
SELECT count (*) INTO cnt FROM jobs WHERE job=variable;  
END WHILE  
  
variable=1  
WHILE variable< 10  
LET variable = variable + 1 ;  
IF variable = 9
```

```

THEN
    EXIT WHILE;
END IF
END WHILE

#simple FOREACH LOOP
FOREACH SELECT author_id,contract FROM authors
IF contract > 0 THEN
    CONTINUE FOREACH;
END IF
INSERT INTO tracking VALUES ( author_id );
END FOREACH

#FOREACH LOOP with update cursor
FOREACH c_author FOR SELECT author_id,contract FROM authors
IF contract > 0 THEN
    CONTINUE FOREACH;
END IF
UPDATE authors SET contract = 1 WHERE CURRENT OF c_author;
END FOREACH

```

4.3.5 Dynamic SQL using the EXECUTE statement

An SQL application programming environment must support dynamic SQL. Statements are not always known at the implementation time of the code and need to be generated and executed individually during run time. The SQL Server EXECUTE statement, which is followed by a string argument passing a statement for execution on the database server, can be converted differently, depending on its context. Example 4-85 shows a typical use case for the EXECUTE statement in T-SQL.

Example 4-85 T-SQL batch using an EXECUTE statement

```

DECLARE @var varchar(100)

SET @var = ' INSERT INTO dbo.jobs ( job_desc, min_level, max_level)  VALUES ('
+ ''Secretary'' + ',44,139);'
execute (@var)
execute (' INSERT INTO dbo.jobs ( job_desc, min_level, max_level)  VALUES
(''Secretary'' ,44,139)')

```

In case a static string is passed as the input parameter, you can use the statement as a static SQL statement in the Informix SPL code. If a variable is used, representing a dynamically built statement, you have to replace the EXECUTE statement with an EXECUTE IMMEDIATE statement in Informix SPL. You can use the EXECUTE IMMEDIATE SPL statement either for executing

non-cursor statements, such as Data Definition Language (DDL) statements, or cursor statements, such as selects. Example 4-86 shows the equivalent Informix SPL procedure declaration.

Example 4-86 Informix stored procedure using an EXECUTE IMMEDIATE

```
CREATE PROCEDURE dynamic()
DEFINE var varchar(100);
LET var = ' INSERT INTO dbo.jobs ( job_desc, min_level, max_level)  VALUES ('
|| ' "Secretary" ' || ',44,139)';

EXECUTE IMMEDIATE (var);

END PROCEDURE;
```

4.3.6 Error handling and status handling

A good programming style includes these components:

- ▶ Verification of success for the implemented operations
- ▶ Notification of the status back to the application
- ▶ Implementation of corrective actions in case unexpected errors occur

Now, we compare ways to determine program status and possible errors in T-SQL and Informix SPL.

@@ROWCOUNT

The T-SQL system function @@ROWCOUNT indicates the number of rows selected or affected by the previous statement.

The equivalent implementation in Informix SPL is the built-in function called dbinfo(). Use this statement to retrieve the number of rows affected by any INSERT, UPDATE, or DELETE statement and to estimate the number of rows expected as a result of a PREPARE statement. Example 4-87 illustrates the use of the dbinfo function and appropriate parameter settings.

Example 4-87 Investigation of the processed rows in the last executed SQL statement

```
CREATE PROCEDURE ins_brand () RETURNING integer;
DEFINE nrows INTEGER;
LET nrows= 0;

INSERT INTO brands (brand_id, brand_name, brand_desc)
VALUES (1, 'Informix', 'Informix');
-- Investigate the #rows processed by the previous insert, expect one
LET nrows= DBINFO('sqlca.sqlerrd2');
```

```
RETURN nrow;  
END PROCEDURE;
```

@@IDENTITY

An T-SQL application can use the @@IDENTITY system variable to obtain the last internally assigned value for any IDENTITY columns in a table during an INSERT. You can map IDENTITY data types in Informix to SERIAL or BIGSERIAL types during schema migration. In addition, there is a certain interface to obtain the assignment of the values, for columns defined on these data types in Informix SPL. The following example shows the use of the dbinfo built-in function:

```
LET last_serial_value = DBINFO('sqlca.sqlerrd1');
```

@@TRANCOUNT

Transactions can be nested in T-SQL. You start a transaction with BEGIN TRANSACTION and before you commit this transaction, you can split another transaction with a subsequent BEGIN TRANSACTION by using certain levels of nesting. The next COMMIT TRANSACTION transaction statement commits the statements that are submitted on the server on the current transaction level in the session. You can obtain the value of the current level of transaction nesting using the @@TRANCOUNT. Without an open transaction, the value is 0, the next level is 1, and so on.

Informix has a similar transaction nesting implementation. Implement nested transactions by using *savepoints*. You start a transaction first. If you want to split another transaction, you create a savepoint at this place. Later, you can decide if you want to commit or roll back the activities up until this savepoint or any other existing savepoints. This action can spawn certain levels of nesting.

@@ERROR

In T-SQL, the execution of every statement, as well as the evaluation of any IF condition, has the effect of setting the @@ERROR system function to a success or failure value. Execution continues to the next statement. It is the programmer's responsibility to check if the execution of any statement failed and to take appropriate action. Only fatal errors cause the transaction to be rolled back.

In Informix SPL, an error encountered during the processing of any SQL statement will result in an exception being raised. If the exception is not caught by a specific exception handler, the stored procedure is terminated and returns the error of the failed SQL statement back to the application. You can retrieve the error status codes for the specific error either from the SQLCODE or SQLSTATE system variables.

Informix SPL provides an interface for overwriting the default behavior. You can catch certain error conditions and add additional code to ensure the successful final execution of the stored procedure. The most direct way of converting @@ERROR into Informix SPL logic is to create a default exception handler in every stored procedure, as shown in Example 4-88.

Example 4-88 Informix SQL using a stored procedure with a default exception handler

```
CREATE procedure add_job(desc CHAR(20), minlvl int, maxlvl int)
RETURNING INT;
DEFINE x INT;
DEFINE sqlcode int;
ON EXCEPTION IN (-206) set sqlcode
-- If no table was found, create one
CREATE TABLE jobs
(
    job_id          serial,
    job_desc        varchar(50),
    min_level int CHECK (min_level >= 10),
    max_level int CHECK (max_level <= 250)
);
INSERT INTO jobs VALUES (0,desc,maxlvl,minlvl);
END EXCEPTION WITH RESUME;
INSERT INTO jobs VALUES (0,desc,maxlvl,minlvl);
SELECT count(*) INTO x FROM emp_list;
RETURN x;
END PROCEDURE;
```

This SET clause in the exception handler triggers Informix to copy the SQLCODE and the appropriate indexed sequential access method (ISAM) error value into a local variable, which can be used for tracking reasons or returned to the calling application. The exception handler must be declared as a RESUME handler so that execution continues with the statement following the statement that raised the exception.

Exceptions during a condition

In T-SQL, if an error occurs during the evaluation of the IF branch of an IF-THEN-ELSE condition, the condition evaluates to false and the ELSE branch of the condition is executed. In Informix SPL, an exception is raised, and if an exception handler has been defined, Informix SPL will catch the exception. After the error processing logic is complete, statement execution continues with the statement following the statement that raised the error.

There is no direct way to convert this behavior to Informix SPL. Rather than simulate this type of logic in Informix SPL, you can redesign the error-handling

code where this type of behavior occurs (errors in IF condition evaluation) and adopt an Informix SPL style of error-handling behavior and logic.

RAISERROR

T-SQL provides the capability to create user-defined errors that are associated with their appropriate error messages. A RAISERROR statement is equivalent to a PRINT statement, followed by an assignment to @@ERROR.

Informix SPL allows you to raise user-defined or system-defined errors, depending on investigated status information in the program logic. User-defined errors can use their own SQL error number -746. You can add your own error message to this particular error.

The message is returned together with the error to the caller of the SP. You are also able to raise any other system-defined error to force an certain exception handler in your current stored procedure. Example 4-89 describes a common scenario showing how to raise a system SQL error in order to force the execution of the appropriate handler in the stored procedure.

Example 4-89 Using raise exception to force error handling code in Informix SPL

```
CREATE procedure add_job(desc CHAR(20), minlvl int, maxlvl int)
RETURNING INT;
DEFINE x INT;
DEFINE sqlcode int;
ON EXCEPTION IN (-206) set sqlcode
-- If no table was found, create one
CREATE TABLE jobs
(
    job_id          serial,
    job_desc        varchar(50),
    min_level int CHECK (min_level >= 10),
    max_level int CHECK (max_level <= 250)
);
INSERT INTO jobs VALUES (0,desc,maxlvl,minlvl);
END EXCEPTION WITH RESUME;
SELECT count(*) INTO x FROM systables WHERE tablename = "jobs";
IF ( x = 0 )
THEN
    raise exception -206, 0 ;
ELSE
INSERT INTO jobs VALUES (0,desc,maxlvl,minlvl);
END IF
SELECT count(*) INTO x FROM emp_list;
RETURN x;
END PROCEDURE;
```

The conversion of T-SQL's RAISERROR to Informix's RAISE EXCEPTION depends on the context in which RAISERROR originally was used:

- Convert a RAISERROR statement that is immediately followed by a RETURN statement using the Informix SPL construct:

```
FOREACH SELECT author_id INTO id FROM authors
IF id < 0 THEN
RAISE EXCEPTION -746, 0, 'No author found'
END IF
END FOREACH
```

- Convert the commonly used sequence RAISERROR; ROLLBACK; RETURN as though it were the sequence ROLLBACK; RAISERROR; RETURN, because the @@ERROR value is reset to zero by the ROLLBACK statement in the former sequence.

4.3.7 Try and catch

T-SQL implements the constructs of a BEGIN TRY, END TRY followed by a BEGIN CATCH, END CATCH block. The statements in the TRY block are executed. In case an error occurred that is specified by a certain severity range that is defined in T-SQL, the statements in the associated CATCH block are executed next. See a T-SQL example where we violated the value range of a variable:

```
declare @aa smallint
begin try
    set @aa =40000 +1
end try
begin catch
    print @@error
end catch
```

Informix SPL does not implement this logic. You have to replace your TRY and CATCH implementation. The best match is an implementation of consistently checking on the errors or the status of variables after certain required statements in your program. If an error or an exception occurred, use a GOTO label statement, triggering the execution of the associated error handling routines. An alternative implementation in your Informix SPL can be the raise of exceptions. Look at Example 4-90 for a possible simple replacement. In this example, a simple range violation is caught and an insert into a tracking table is triggered.

Example 4-90 Catch a data type range violation

```
DROP PROCEDURE range_violation;

CREATE PROCEDURE range_violation()
    DEFINE x SMALLINT;
```

```

DEFINE cnt SMALLINT;
DEFINE sqlcode int;

-- take this as the Begin catch
ON EXCEPTION IN (-1214) set sqlcode
-- If no table was found, create one
SELECT count(*) INTO cnt FROM systables;
IF ( cnt = 0 )
THEN
CREATE TABLE track
(
    job_id          serial,
    sqlcode integer
);
END IF
INSERT INTO track VALUES (0,sqlcode);
END EXCEPTION WITH RESUME;
-- take this as the end catch

-- take this as begin try -- assignment will cause range violation
LET x=40000 +1;
-- take this as end try
END PROCEDURE;

EXECUTE PROCEDURE range_violation();

```

4.3.8 Debugging

We compare the facilities of printing status variables and indicating reached code points by using tracing.

PRINT

In SQL Server, use a **PRINT** statement to display a character string to the user. You can use a **PRINT** statement during development to print debugging statements, but you can also use the **PRINT** statement for logging and as a quick way to notify the user. The **PRINT** statement differs from the **SELECT** statement. The **PRINT** statement returns a message of severity 0; the **SELECT** statement returns a set of data.

No direct equivalent to the **PRINT** statement exists in Informix SPL. But, Informix SPL provides a complete mechanism for tracing stored procedure code. You can specify a trace file location based on the database server. You can optionally switch on debugging, switch off debugging, and include your own debug messages in your stored procedure code.

In addition, SQL statements and expressions are traced automatically, which differs from SQL Server, where the generated output is not sent back to the application. For Informix, all messages are written to a local file on the database server machine, instead of being displayed on the window. Example 4-91 depicts a sample stored procedure and the generated trace output that was taken from the target trace file.

Example 4-91 Adding debugging code to your Informix stored procedures

```
DROP PROCEDURE testproc();
CREATE PROCEDURE testproc()
    DEFINE i INT;
    DEFINE id char(40);
    DEFINE tabname char(40);
    LET i=0;
    LET tabname="";
    SET DEBUG FILE TO 'C:\debugging\testproc.trace';
    TRACE ON;
    TRACE 'Entering SP';
    FOREACH SELECT author_id INTO id FROM authors
        LET i = i + 1;
    END FOREACH;
    TRACE 'i+1 = ' || i+1;
    TRACE 'Exiting testproc';
    TRACE OFF;
END PROCEDURE;
EXECUTE PROCEDURE testproc();
```

-- generated output in the trace file
trace on

trace expression :Entering SP

start select cursor.
select author_id
from authors
select cursor iteration.
select cursor returns
expression:(+ i, 1)
evaluates to 1

....

trace expression :i+1 = 84
trace expression :Exiting testproc
trace off

4.3.9 More function mapping

In this section, we list the mapping between functionality provided by T-SQL and its corresponding implementation in Informix SPL. Our major focus is user and session information retrieval. Refer to Table 4-8 for details.

Table 4-8 User and session management in T-SQL and Informix SPL

T-SQL	Informix SPL
KILL <sessionid>	As Informix user: EXECUTE FUNCTION sysadmin:task ("onmode","z",<sessionid>"); Session id taken from onstat -g sql
KILL <indoubt XA transaction>	As Informix user: EXECUTE FUNCTION sysadmin:task ("onmode","Z",<Address of tx>"); Address taken from onstat -G
SELECT USER_ID('dbo')	Check <database>:sysusers for user details, such as permissions and roles
SELECT SYSTEM_USER	SELECT user FROM systables WHERE tabid=1;
@@SERVERNAME	SELECT DBINFO('dbhostname') FROM systables WHERE tabid=1;
@@VERSION	SELECT DBINFO('version') FROM systables WHERE tabid=1;
@@SPID	SELECT DBINFO('sessionid') FROM systables WHERE tabid=1;

4.4 SQL-based security considerations

The overall database server instance includes the SQL-based security, auditing, client/server communication, server-server communication, and user authentication. We refer to the tasks, which maintain the access control on the databases, as well as database objects and their content, as the *SQL-based security*, because the security is managed through the SQL statements, such as GARNT. The access control includes permission management and also column data encryption.

In this section, we focus our discussion on SQL-based security. For the other security topics, refer to *Security and Compliance Solutions for IBM Informix Dynamic Server*, SG24-7556-00.

4.4.1 Database user management

SQL Server defines two types of database user management. You can allow only the Microsoft Windows user accounts, or you can allow a combination of Windows accounts and database server-managed accounts. Create database server-managed accounts by using the following SQL statements:

```
CREATE LOGIN author WITH PASSWORD = 'author_passwrd';  
CREATE USER author FOR LOGIN author;
```

Informix authenticates the database user for new incoming session requests based on the underlying operating system. Currently, no additional database user management in the database server exists in Informix.

4.4.2 Database server and database permissions

SQL Server provides a fine granularity of permissions on the database level. The permissions are maintained with the SQL GRANT and REVOKE statements. Permissions on the database server level can be granted to a user. The database-level control includes the ability to create, access, alter, and drop any kind of database objects, such as databases, tables, views, and so on. You can grant additional permissions on the instance level, for example, the shutdown permissions and the user login management.

Informix maintains permissions on the database and instance levels differently from SQL Server. Several of the permissions cannot be applied by SQL statements to a particular user; instead, the instance administrator has to change certain configuration files.

Comparing the SQL Server's permission management for the instance and database with Informix's, five major methods exists by which you can accomplish similar user restrictions:

- ▶ SQL GRANT and REVOKE statements
- ▶ Grant access for sysadmin database and execution permission for the task function
- ▶ ONCONFIG file
- ▶ Role separation
- ▶ Instance access control

Granting and revoking permissions on the database level

In general, both SQL Server and Informix use SQL statements to maintain the database-level access permissions. Informix provides, on the database level, the following permissions that are managed by using the SQL GRANT and REVOKE statements:

- ▶ Connect database permission:
 - User can connect to a database.
 - Depending on the object permissions, the user can perform operations, such as select, update, and delete on database objects.
 - Depending on the object permissions, the user is able to create synonyms, views, temp tables, and indexes on temp tables.
- ▶ Resource database permission
Extend the connect permission by allowing the creation of new tables, indexes, stored procedures, and user-defined types.
- ▶ DBA permission
Extend the resource permission:
 - Grant permissions to users or roles in the name of another user
 - Create and drop any database object
 - Create database objects in the name of another user
 - Drop and rename the database

The following examples grant or revoke the database-level permissions:

```
GRANT RESOURCE TO UserA
GRANT DBA TO ProjectManager
REVOKE CONNECT FROM ExProjectUser
```

The user permissions on the database level are maintained in the sysusers system catalog table in each database on the server.

Sysadmin database

SQL Server provides GRANTs for a set of instance permissions for administration tasks. You must apply these permissions to an Informix instance in the migration. Informix uses a separate method to enable a user to perform a certain administration task.

The Informix database server instance contains several internal databases. One of them is the *sysadmin* database that is created at the instance initialization time. The general purpose of sysadmin is for task scheduling using an internal system function task() that provides the remote database administration and SQL-based database administration support, including log management, system shutdown, and space management.

Initially, the access on the sysadmin database is only granted to the *informix* user. If you want to give another user, other than the *informix* user, the ability to perform administration tasks, you have to grant the user the connect permission to the sysadmin database and the execution permission to the task() function. Note that doing so grants the user all the permissions, not just the selected ones.

ONCONFIG file

In Informix, the permission that is similar to the SQL Server's CREATE ANY DATABASE permission is maintained within the instance configuration file that is specified in the ONCONFIG environment variable. You can specify a parameter DBCREATE_PERMISSION followed by a list of users in the file, which allows a user or a certain user group to create databases in the current database instance.

Also, accessing sensitive session data from monitoring utilities, such as the onstat for a certain database user, is maintained in the ONCONFIG file by a separate parameter.

Role separation

In a typical Informix database server installation, the user *informix* is the administrator ID for the database server. You can perform the maintenance tasks, such as backup and restore, shutdown and startup, auditing, and monitoring with the *informix* user. Migrating from SQL Server's SQL, you do not need to grant the *informix* user the authorities to manage the database. The authorities have been granted to the *informix* user, by default.

Informix provides a role separation functionality that allows you to split the responsibilities (the database administration work, audit maintenance, and audit analysis) to separate user accounts. You can divide the responsibilities when you install the database product or as a subsequent activity after the installation. Using role separations is similar to using the SQL Server GRANT ALTER ANY SERVER AUDIT to grant authorities to a user other than the *informix* user.

For more detailed information about role separation and security in Informix, refer to *Security and Compliance Solutions for IBM Informix Dynamic Server*, SG24-7556-00.

Instance access control

In Informix, you can grant and revoke the access to a specific database from or to a user. You are also able to restrict a user from accessing the entire instance. Informix uses the \$INFORMIXDIR/dbssodir/seccfg file provided in the database server distribution to control the instance access.

In the default server setup, all users are allowed to access the instance with the following content in the `seccfg` file:

```
IXUSERS=*
```

To restrict the access to the database instance to a certain group of users, replace the asterisk (*) with the group name to which to grant the access to the instance:

```
IXUSERS=instance_user_group
```

Any user that does not belong to the group is rejected during a connection attempt.

4.4.3 Database object permissions

The EXECUTE, INSERT, DELETE, SELECT, and other permissions are commonly granted on the database object level. The objects can be tables, columns, indexes, views, built-in functions, stored procedures, and user-defined functions. SQL Server and Informix have the same implementation on the database object permission management:

- ▶ A grantor can only grant the permissions that the grantor has and revoke the permissions that the grantor grants.
- ▶ GRANT is a permission that can be granted.
- ▶ The DBA can grant permissions of an object that is owned by other users using the `AS <grantor>` clause of the GRANT statement.

The following SQL examples show granting and revoking database object permissions:

```
GRANT SELECT,UPDATE ON dbo.authors TO userA
      WITH GRANT OPTION AS informix;
REVOKE UPDATE ON dbo.authors(author_id) FROM userB;
GRANT EXECUTION ON dbo.add_brand TO userC
```

In Informix, the existing permissions for the database objects are maintained in the system catalog for each database separately. The following tables maintain the object permissions:

- ▶ `syscolauth` for the column permissions
- ▶ `systabauth` for the table permissions
- ▶ `sysprocauth` for the procedure permission
- ▶ `sysxtdtypeauth` for permission on extended types

4.4.4 Roles

Role management is similar in SQL Server and Informix. Roles logically combine database users performing similar types of work in the same department or hierarchy to a workgroup. Use the following two typical roles to simplify the access management:

- ▶ To implement security requirements, set up roles according to the job responsibility and security requirements, for example, manager and accountant. The data access permissions are granted to these roles. The users then are assigned to the role based on their job responsibilities. The user can only access the data required in that role. If the user's responsibility changes, the DBA removes the user but does not have to change the security settings for that role.
- ▶ To ease the user management, create a role for each project and grant the object access permissions to the role. The users, such as developers, then are assigned to the role. If a user leaves the project, the DBA removes this user from the role. This approach saves the DBAs time in tracking the permission in the individual level. The time savings is especially noticeable in large companies with a great number of databases and objects.

On both database servers, you can use this statement to create roles:

```
CREATE ROLE rolename
```

SQL Server maintains the users and role assignments with system stored procedures.

In Informix, you can grant roles to users with SQL statements, for example:

```
GRANT userA TO rolename
```

After the user is assigned to a role, the role and the inherited permissions are not automatically applied to the user at the connection time of the database. The database user or the application has to explicitly set the role. To assign the user and the role automatically, you set a role as the default role when a user is assigned to a role. Then, every time that the user connects to the database, the role is automatically set. You can grant the default role with the following statement:

```
EXEC SQL GRANT DEFAULT ROLE 'rolename' TO UserA.
```

Permissions are granted to the roles that are similar for a particular user:

```
GRANT SELECT ON dbo.author(author_id) TO rolename AS dba
```

Roles are maintained in trees. A role can be assigned to another role, depending on project hierarchies or company organizational structure. SQL Server provides

the `sp_addrolemember` and `sp_droprolemember` stored procedures for maintaining role trees. Informix creates the role tree with the following statements:

```
GRANT lowerrole TO upperrole
REVOKE lowerrole FROM upperrole
```

In Informix, the role trees are maintained in the `sysrolauth` system catalog table. The `sysusers` table contains roles and the default role definitions.

4.4.5 Column-level encryption

SQL Server's T-SQL provides the `DECRYPTBY` and `ENCRYPTBY` function sets for maintaining data encryptions for table columns. You can invoke separate encryption functions based on the type of key selected. To display the original content of an encrypted column, you must use the corresponding decryption routine and apply the key used for the encryption.

Informix also provides built-in functions for encrypting values in table columns. The names of the functions are `ENCRYPT_AES` and `ENCRYPT_TDES` for encryption and `DECRYPT_CHAR` and `DECRYPT_BINARY` for decryption. The encryption is based on Advanced Encryption Standard (AES) and Data Encryption Standard (DES) ciphers. The target data types in the table storing the encrypted values must be `CHAR`, `VARCHAR`, `LVARCHAR`, `BLOB`, and `CLOB` data types, because the encryption functions generate variable length output. Nevertheless, encryption can also be applied on values that are taken from numeric and date or "datetime" columns.

Encryption and decryption are password-based in Informix. You can set a general password for your session to be used for all column value encryption. You also can specify a password for each column-based or row-based key separately. For decryption, make sure that you apply the appropriate key used by the encryption.

Example 4-92 shows examples using a global and a local password for the encryption of a `CHAR`-based column in the `authors` table.

Example 4-92 Using encryption in Informix

```
# Use one encryption key for the row
dbaccess -e booktab << EOF
CREATE TABLE authors ( author_id SERIAL , name CHAR(50)) ;
INSERT INTO authors VALUES ( 0, ENCRYPT_AES("Smith","bookdb"));
SELECT * FROM authors
EOF
```

```
author_id    name
```

```

1 01Gv/AAAAEADz0i4qJ+PNR8b+hdYI+

dbaccess -e bookdb << EOF
select author_id, decrypt_char(name,"bookdb") FROM authors;
EOF

#Use one encryption key for the session within a general setting
dbaccess -e booktab << EOF
CREATE TABLE authors ( author_id SERIAL , name CHAR(50)) ;
SET ENCRYPTION PASSWORD "booktab";
INSERT INTO authors VALUES ( 0, ENCRYPT_AES("Smith"));
SELECT * FROM authors
EOF

author_id name

1 01Gv/AAAAEADz0i4qJ+PNR8b+hdYI+

dbaccess -e bookdb << EOF
SET ENCRYPTION PASSWORD "booktab";
SELECT author_id, decrypt_char(name) FROM authors;
EOF

author_id (expression)
1 Smith

```

4.5 Development considerations

To be successful in migrating SQL Server to Informix, one of the key considerations is application conversion. To achieve that, it is important to understand the various aspects that can influence the performance, scalability, and concurrency.

This section presents the development considerations of the transactional model, isolation level, locking control, and cursor differences between Informix and SQL Server.

4.5.1 Concurrency and transaction

Concurrency control is an important topic in the database management system (DBMS). *Concurrency control* makes sure that multiple users can read and write records in a consistent manner. Every DBMS has similar concurrency control mechanisms, but the implementations of these mechanisms typically differ.

Therefore, it is important to consider these differences in planning your migration and application conversion.

We expand on the description of the transaction model in 2.7, “Transactional model” on page 39 and explain the nested transaction model.

In SQL Server, transactions are in the nested model. The outermost pair creates and commits the transaction, and the inner pairs track the nesting level. When a nested transaction is encountered, the @@TRANCOUNT function is incremented. Usually, this apparent transaction nesting occurs as stored procedures or triggers with BEGIN...COMMIT pairs calling each other.

On Informix, there is a similar transaction nesting implementation. Nested transactions are achieved by savepoints. You start a transaction, and if you want to split another transaction, you create a savepoint at this place. Later, you can decide if you want to commit or roll back the activities to this savepoint or any other existing savepoints. Using savepoints also can spawn certain levels of nesting. Table 4-9 shows examples that illustrate the point.

Table 4-9 Nested transactions

SQL Server	Informix
<pre>CREATE PROCEDURE proc_a AS BEGIN TRANSACTION; INSERT INTO dbo.authors (author_id, name, firstname, contract) VALUES ("001-07-1234", 'John Smith', 'John', 1); COMMIT TRANSACTION; GO CREATE PROCEDURE proc_b AS BEGIN TRANSACTION; EXECUTE proc_a; UPDATE dbo.booktab SET author_id = "001-07-1234" WHERE book_id = 820318; COMMIT TRANSACTION; GO EXECUTE proc_b ; GO</pre>	<pre>CREATE PROCEDURE proc_a() SAVEPOINT new_author; INSERT INTO authors (author_id, name, firstname, contract) VALUES ("001-07-1234", "John Smith", "John", 1); END PROCEDURE; CREATE PROCEDURE proc_b() EXECUTE PROCEDURE proc_a(); SAVEPOINT change_author; UPDATE booktab SET author_id = "001-07-1234" WHERE book_id = 820318; END PROCEDURE; BEGIN WORK; EXECUTE PROCEDURE proc_b (); COMMIT WORK;</pre>

The scope of savepoints set inside a stored procedure is restricted to the individual execution of the stored procedure. Procedure calls can be nested or recursive, but the scope of a savepoint is restricted to the execution of the stored procedure in which it is defined. Therefore, a savepoint name needs to be unique only in its scope of execution. You can refer to a savepoint only in the same execution of a procedure, but not from another execution of the same or other procedures irrespective of the nesting level.

4.5.2 Isolation level

The *isolation level* controls when and what types of locks are obtained on a database object, such as a row, page, or table. Informix supports both the American National Standards Institute (ANSI) SQL-92 specification and non-ANSI compliant. See Table 4-10 for a comparison of SQL Server and Informix isolation levels.

Table 4-10 A comparison of SQL Server and Informix (both ANSI and non-ANSI)

SQL Server	Informix ANSI standard	Informix non-ANSI standard
read committed (default)	read committed	committed read
Serializable	Serializable	repeatable read
repeatable read	repeatable read	repeatable read
uncommitted read	read uncommitted	dirty read
Snapshot™	N/A	committed read with LAST COMMITTED option
Read committed snapshot	N/A	committed read with LAST COMMITTED option

Default isolation levels

The default isolation level for a particular database is established when you create the database according to the database type. Table 4-11 on page 180 lists the default isolation level for each database type.

Table 4-11 Informix default isolation level based on database type

Non-ANSI isolation level	ANSI isolation level	Conditions when this level is the default level of isolation
dirty read	read uncommitted	Database without transaction logging
committed read	read committed	Database with logging that is not ANSI-compliant
repeatable read	Serializable	ANSI-compliant database

The default level remains in effect until you issue a SET ISOLATION statement. After a SET ISOLATION statement executes, the new isolation level remains in effect until one of the following events occurs:

- ▶ You enter another SET ISOLATION statement.
- ▶ You open another database that has a default isolation level that differs from the level that your last SET ISOLATION statement specified.
- ▶ The program ends.

For an Informix database that is not ANSI-compliant, unless you explicitly set the USELASTCOMMITTED configuration parameter, the LAST COMMITTED feature is not in effect for the default isolation levels. The SET ENVIRONMENT statement or the SET ISOLATION statement can override this default and enable LAST COMMITTED for the current session.

Informix isolation levels

Informix implements various isolation levels to support read concurrency.

Informix dirty read isolation

The simplest isolation level, ANSI read committed or Informix dirty read, amounts to virtually no isolation. When a program fetches a row, it places no locks, and it respects none. It simply copies rows from the database without regard to other programs' activities. A program always receives complete rows of data. Even under ANSI read uncommitted or Informix dirty read isolation, a program never sees a row in which certain columns are updated and other columns are not. However, a program that uses ANSI read committed or Informix dirty read isolation sometimes reads updated rows before the updating program ends its transaction. If the updating program later rolls back its transaction, the reading program processes data that never really existed.

ANSI read committed or Informix dirty read is the most efficient isolation level. The reading program never waits and never makes another program wait. It is the preferred level in any of the following cases:

- ▶ All tables are static, so concurrent programs read but never modify data.
- ▶ The table is held in an exclusive lock.
- ▶ Only one program is using the table.

Informix committed read isolation

When a program requests the ANSI read committed or Informix committed read isolation level, the database server guarantees that it **never** returns a row that is not committed to the database. This action prevents reading data that is not committed and that is subsequently rolled back.

ANSI read committed or Informix committed read is implemented simply. Before it fetches a row, the database server tests to determine whether an updating process placed a lock on the row; if not, it returns the row. Because rows that have been updated (but that are not yet committed) have locks on them, this test ensures that the program does not read uncommitted data.

ANSI read committed or Informix committed read does not actually place a lock on the fetched row, so this isolation level is almost as efficient as ANSI read uncommitted or Informix dirty read. This isolation level is appropriate to use when each row of data is processed as an independent unit, without reference to other rows in the same or other tables.

Locking conflicts can occur in ANSI read committed or Informix committed read sessions. The attempts to place the test lock might be unsuccessful, because a concurrent session holds a shared lock on the row. To avoid waiting for concurrent transactions to release shared locks (by committing or rolling back), Informix supports the last committed option to the committed read isolation level. When this Last Committed option is in effect, a shared lock by another session causes the query to return the most recently committed version of the row.

You also can activate the last committed feature by setting the USELASTCOMMITTED configuration parameter to COMMITTED READ or to ALL, or by setting the USELASTCOMMITTED session environment option in the SET ENVIRONMENT statement in the sysdbopen() procedure when the user connects to the database. For more information about the Last Committed option to the ANSI read committed or Informix committed read isolation levels, see the description of the SET ISOLATION statement in *IBM Informix Guide to SQL, Syntax*, SC23-7751. For information about the USELASTCOMMITTED configuration parameter, see *IBM Informix Dynamic Server Administrator's Reference*, SC23-7749.

Informix cursor stability isolation

When cursor stability is in effect, Informix places a lock on the latest row fetched. It places a shared lock for an ordinary cursor or a promotable lock for an update cursor. Only one row is locked at a time. That is, each time that a row is fetched, the lock on the previous row is released (unless that row is updated, in which case the lock holds until the end of the transaction).

Because cursor stability locks only one row at a time, it restricts concurrency less than a table lock or database lock. Cursor stability ensures that a row does not change while the program examines it. Such row stability is important when the program updates another table based on the data it reads from the row. Because of cursor stability, the program is assured that the update is based on current information and prevents the use of stale data.

Informix repeatable read isolation

Where ANSI serializable or ANSI repeatable read is required, a single isolation level is provided, which is called *Informix repeatable read*. This isolation level is logically equivalent to ANSI serializable. Because ANSI serializable is more restrictive than ANSI repeatable read, Informix repeatable read can be used when ANSI repeatable read is desired (although Informix repeatable read is more restrictive than is necessary in such contexts).

The repeatable read isolation level asks the database server to put a lock on every row that the program examines and fetches. The locks that are placed are shareable for an ordinary cursor and promotable for an update cursor. The locks are placed individually as each row is examined. They are not released until the cursor closes or a transaction ends. Repeatable read allows a program that uses a scroll cursor to read selected rows more than one time and to be sure that they are not modified or deleted between readings. (Programming with SQL describes scroll cursors.) No lower isolation level guarantees that rows still exist and are unchanged the second time that they are read.

Repeatable read isolation places the largest number of locks and holds them the longest. Therefore, it is the level that reduces concurrency the most. If your program uses this level of isolation, think carefully about how many locks it places, how long they are held, and what the effect can be on other programs. In addition to the effect on concurrency, the large number of locks can cause issues. The database server records the number of locks by each program in a lock table. If the maximum number of locks is exceeded, the lock table fills up, and the database server cannot place a lock. In this case, an error code is returned. The person who administers an Informix database server system can monitor the lock table and tell you when it is heavily used.

Setting the isolation level

The isolation level is set by the SET ISOLATION statement, as shown in Example 4-93.

Example 4-93 Setting the isolation level

```
SET ISOLATION TO dirty read;  
SET ISOLATION TO repeatable read;
```

You can also set the isolation level using the SET TRANSACTION ISOLATION LEVEL statement, as shown in Example 4-94.

Example 4-94 Set the isolation level using set transaction statement

```
SET TRANSACTION ISOLATION LEVEL TO REPEATABLE READ
```

The major difference between the SET TRANSACTION and SET ISOLATION statements is the behavior of the isolation levels within transactions. The SET TRANSACTION statement can be issued only one time for a transaction. Any cursors opened during that transaction are guaranteed to have that isolation level. With the SET ISOLATION statement, after a transaction is started, you can change the isolation level more than one time within the transaction.

Choosing the appropriate isolation level

Choosing the appropriate isolation level to use for a transaction is important. The isolation level not only influences how well the database supports concurrency, but it also affects the overall performance of the application containing the transaction. That is because the resources needed to acquire and free locks vary with each isolation level.

If you have a requirement that readers do not block writers, use the Informix dirty read isolation level. The readers will not block the writers with dirty read processing; however, the query result will contain any updates performed while the query was running. If read consistency is a requirement, use the Informix repeatable read isolation level to lock all the resulting rows in a query from updates while the query is running. If both conditions are required, question why they are both required. It will most likely be due to performance, in which case, you must analyze the application to see where changes can be made in the database schema and access methods to eliminate the performance issue.

4.5.3 Locking

SQL Server allows a range of database objects to be locked, from the entire database to a range of key values. Locks are primarily obtained as the result of data read and update requests, but locks can also be obtained for administration reasons, such as during table reorganizations or index rebuilds.

Informix supports similar lock granularity as SQL Server:

- ▶ Database level
- ▶ Table level
- ▶ Page level
- ▶ Row level
- ▶ Key level

There are ways to explicitly request a certain lock level in both Informix and SQL Servers. We describe the Informix lock types, the use of lock mode, how deadlocks are handled, and also a comparison of the explicit lock requests between both databases.

Lock types

Table 4-12 shows the types of locks that Informix supports for various situations.

Table 4-12 *Types of locks in Informix*

Lock type	Use
Shared	A <i>shared lock</i> reserves its object for reading only. It prevents the object from changing while the lock remains. More than one program can place a shared lock on the same object. More than one object can read the record while it is locked in shared mode.
Exclusive	An <i>exclusive lock</i> reserves its object for the use of a single program. This lock is used when the program intends to change the object. You cannot place an exclusive lock where any other kind of lock exists. After you place an exclusive lock, you cannot place another lock on the same object.
Promotable (or Update)	A <i>promotable (or update) lock</i> establishes the intent to update. You can only place it where no other promotable or exclusive lock exists. You can place promotable locks on records that already have shared locks. When the program is about to change the locked object, you can promote the promotable lock to an exclusive lock, but only if no other locks, including shared locks, are on the record at the time that the lock will change from promotable to exclusive. If a shared lock was on the record when the promotable lock was set, you must drop the shared lock before the promotable lock can be promoted to an exclusive lock.

Lock mode option

Lock mode specifies whether to use row or page locks for the table.

The PAGE lock obtains and releases one lock on the entire page of rows. This type of locking is the default locking granularity. Page-level locking is especially useful when you know that the rows are grouped into pages in the same order that you use to process all the rows. For example, if you process the contents of a table in the same order as its cluster index, page locking is appropriate.

The ROW lock obtains and releases one lock per row. Row-level locking provides the highest level of concurrency. If you use many rows at one time, however, the lock-management overhead can become significant. You might also exceed the maximum number of locks available, depending on the configuration of your database server, but Informix can support up to 18 million locks on 32-bit platforms, or 600 million locks on 64-bit platforms. Only tables with row-level locking can support the LAST COMMITTED isolation level feature.

You determine whether the database server acquires page locks or row locks when you create a table or when it is altered:

```
CREATE TABLE orders(  
  order_num SERIAL NOT NULL,  
  customer_num INTEGER,  
  order_date DATE)  
LOCK MODE ROW;
```

```
ALTER TABLE orders LOCK MODE (PAGE);
```

When you create a table, you choose the lock mode used when any rows from that table are accessed. Page-level locking locks an entire data page whenever a single row located on that page needs to be locked. Row-level locking locks only the row in question. The default lock mode when you create a table is page level.

Page locks are useful when, in a transaction, you process rows in the same order as the cluster index of the table or process rows in physically sequential order.

Row locks are useful when, in a transaction, you process rows in an arbitrary order.

Important: When the number of locked rows becomes large, you run these risks:

- ▶ The number of available locks becomes exhausted.
- ▶ The overhead for lock management becomes significant.

Informix lock mode can also be changed by using one of the following methods: (resolved in the following order of precedence)

- ▶ LOCK MODE specified using an attribute of the CREATE TABLE or ALTER TABLE statement
- ▶ IFX_DEF_TABLE_LOCKMODE environment variable setting
- ▶ DEF_TABLE_LOCKMODE parameter setting in the ONCONFIG file

If the DEF_TABLE_LOCKMODE parameter cannot be found in the ONCONFIG file, you can add it to make the specification for every database within the instance. The Informix instance must be restarted for this parameter to take effect.

DEADLOCK

A *deadlock* is a situation in which a pair of programs blocks the progress of each other. Each program has a lock on an object that the other program wants to access. A deadlock arises only when all concerned programs set their lock modes to wait for locks.

Informix detects deadlocks immediately when they only involve data at a single network server. It prevents the deadlock from occurring by returning an error code (error -143 ISAM error: deadlock detected) to the second program to request a lock. This error code is the code that the program receives if it sets its lock mode to not wait for locks. If your program receives an error code related to locks even after it sets lock mode to wait, you know the cause is an impending deadlock.

Use the DEADLOCK_TIMEOUT configuration parameter in the onconfig file to specify the maximum number of seconds that a database server thread can wait to acquire a lock. It takes effect when the database server is shut down and restarted. The default value is 60 (seconds).

Explicit lock request comparison

To help developers switch from SQL Server to Informix, we provide several commonly used lock request comparisons between SQL Server and Informix; see Table 4-13.

Table 4-13 A comparison of SQL Server and Informix explicit lock requests

SQL Server	Informix
SELECT... (TABLOCK)	LOCK TABLE...IN SHARE MODE
SELECT... (TABLOCKX)	LOCK TABLE...IN EXCLUSIVE MODE
LOCK_TIMEOUT	SET LOCK MODE TO [WAIT seconds NOT WAIT]

In Informix, use the SET LOCK MODE statement to define how the database server handles a process that tries to access a locked row or table. Use the following syntax:

```
SET LOCK MODE TO [NO WAIT / WAIT seconds]
```

When NO WAIT is used, the database server ends the operation immediately and returns an error code. This condition is the default.

When WAIT is used, the database server suspends the process until the lock releases.

When the WAIT *seconds* parameter is used, the database server suspends the process until the lock releases or until the waiting period (which is specified in the number of seconds) ends. If the lock remains after the waiting period, the operation ends and an error code is returned.

4.5.4 Cursors

Other than the syntax differences, the cursor usage is similar between Informix and SQL Server. Table 4-14 on page 188 compares the common uses of cursors in both databases.

Table 4-14 Comparison of cursor usage between SQL Server and Informix

Usage	SQL Server	Informix
Declare cursor	DECLARE cursor_name CURSOR FOR select_statement	DECLARE cursor_name CURSOR FOR SELECT ... INTO :var1, :var2 FROM table_name;
Open a cursor	OPEN cursor_name	OPEN cursor_name;
Fetch from a cursor	FETCH [[NEXT PRIOR FIRST LAST] cursor_name [INTO @variables]	FETCH cursor_name;
Update fetched row	UPDATE table_name SET statements... WHERE CURRENT OF cursor_name	UPDATE table_name SET statements... WHERE CURRENT OF cursor_name;
Delete fetched row	DELETE FROM table_name WHERE CURRENT OF cursor_name	DELETE FROM table_name WHERE CURRENT OF cursor_name;
Close a cursor	CLOSE cursor_name	DELETE FROM table_name WHERE CURRENT OF cursor_name;
Free a cursor	CLOSE cursor_name	UPDATE table_name SET statements... WHERE CURRENT OF cursor_name;

SQL Server supports all ANSI-style cursors: static, dynamic, forward only, and keyset-driven. This support includes support for insensitive and scroll cursor behavior and for all fetch options (FIRST, LAST, NEXT, PRIOR, RELATIVE, and ABSOLUTE). Cursor support is available through the following interfaces: ADO, OLE DB, ODBC, DB-Library, and T-SQL.

In Informix, you can declare a cursor as a sequential cursor (the default), a scroll cursor (by using the SCROLL keyword), or a hold cursor (by using the WITH HOLD keywords). The SCROLL and WITH HOLD keywords are not mutually exclusive.

If you use only the CURSOR keyword, you create a sequential cursor, which can fetch only the next row in sequence from the active set. The sequential cursor can read through the active set only one time each time that it is opened. If you use a sequential cursor for a Select cursor, on each execution of the FETCH

statement, Informix returns the contents of the current row and locates the next row in the active set.

You can use the SCROLL keyword to create a scroll cursor, which can fetch rows of the active set in any sequence. Informix retains the active set of the cursor as a temporary table until the cursor is closed. You can fetch the first, last, or any intermediate rows of the active set, as well as fetch rows repeatedly without having to close and reopen the cursor.

Transaction behavior in cursors

The default behavior in SQL Server is to have cursors remain open after a COMMIT or ROLLBACK statement is issued. This behavior strays from the ANSI SQL standard, although it is possible to configure SQL Server to use the ANSI SQL standard behavior of closing open cursors after a COMMIT or ROLLBACK statement is issued.

Informix's default behavior follows the ANSI SQL standard of closing open cursors whenever a COMMIT or ROLLBACK statement is issued. However, cursors that are declared with the WITH HOLD option remain open after a COMMIT statement is issued. In Informix, all open cursors are closed when a ROLLBACK statement is issued.

Important: In Informix, after a COMMIT statement has been issued, a cursor declared with the WITH HOLD statement remains open and is positioned before the next logical row of the results table. Additionally, all locks are released except for locks protecting the current cursor position.

DEALLOCATE in SQL Server

SQL Server provides a cursor cleanup command called DEALLOCATE, which typically is used as a performance benefit to eliminate the cost of declaring a new cursor after the last reference to the cursor is de-allocated. The DEALLOCATE command allows a cursor to be closed and reopened without re-declaring it.

Informix does not have a similar DEALLOCATE command. Cursors must be explicitly closed when they are no longer needed, and they must be re-declared if and when they are needed again.

Converting cursor attributes

SQL Server supports cursor attributes to obtain information about the current status of a cursor. You can use SQLCODE/SLSTATE values to obtain the analogous information in Informix. This section shows how to match SQL Server cursor attribute functions with Informix SQLCODE/SQLSTATE values.

@@CURSOR_ROWS

In SQL Server, the @@CURSOR_ROWS function returns the number of qualifying rows from the previous cursor declaration. In Informix, you can use a local (counter) variable to store this information after each FETCH operation from the cursor. Example 4-69 on page 146 shows how to implement this function in Informix. In Example 4-95, a local variable is declared to hold the number of fetched rows [1], and it is incremented each time that a new row is fetched [2].

Example 4-95 Informix cursor that counts the number of fetched rows

```
CREATE PROCEDURE tt(val int)
  DEFINE v_cursor_rows INTEGER;          -- [1]
  DEFINE my_ename INTEGER;
  DEFINE my_deptno VARCHAR(40);

  LET v_cursor_rows = 0;
  prepare p1 from 'SELECT author_id, name FROM authors';
  DECLARE c1 CURSOR FOR p1;

  LOOP
    FETCH c1 INTO my_ename, my_deptno;
    LET v_cursor_rows = v_cursor_rows + 1;    -- [2]
  END LOOP;

END PROCEDURE;
```

FOREACH in Informix SPL

We described FOREACH in Informix SPL in 4.1.8, “Procedures and functions” on page 105. Now, we expand on the topic by focusing on the comparison of the SPL cursor between SQL Server and Informix.

In Informix stored procedures, cursors do not need to be explicitly declared, opened, and fetched. You can replace SQL Server stored procedure cursors with Informix’s FOREACH construct. See Table 4-15 on page 191.

Table 4-15 Common use comparisons of cursors between SQL Server and Informix

SQL Server	Informix
DECLARE cursor_name CURSOR FOR select_statement OPEN cursor_name FETCH NEXT FROM cursor_name WHILE @@FETCH_STATUS = 0 BEGIN FETCH NEXT FROM cursor_name END CLOSE cursor_name DEALLOCATE cursor_name	FOREACH cursor_name [WITH HOLD] FOR select_statement [INTO variables] END FOREACH;

To execute a FOREACH statement, the database server takes these actions:

1. It declares and implicitly opens a cursor.
2. It obtains the first row from the query contained within the FOREACH loop or else the first set of values from the called routine.
3. It assigns to each variable in the variable list the value of the corresponding value from the active set that the SELECT statement or the called routine creates.
4. It executes the statement block.
5. It fetches the next row from the SELECT statement or called routine on each iteration, and it repeats steps 3 and 4.
6. It terminates the loop when it finds no more rows that satisfy the SELECT statement or called routine. It closes the implicit cursor when the loop terminates.

Because the statement block can contain additional FOREACH statements, cursors can be nested. No limit exists on the number of nested cursors.

An SPL routine that returns more than one row, collection element, or set of values is called a *cursor function*. An SPL routine that returns only one row or value is a *noncursor function*.

The SPL procedure that is shown in Example 4-96 illustrates FOREACH statements with a SELECT ... INTO clause, with an explicitly named cursor, and with a procedure call.

Example 4-96 The use of FOREACH in SPL

```
CREATE PROCEDURE foreach_ex()  
  DEFINE i, j INT;  
  FOREACH SELECT c1 INTO i FROM tab ORDER BY 1
```

```

        INSERT INTO tab2 VALUES (i);
    END FOREACH
    FOREACH cur1 FOR SELECT c2, c3 INTO i, j FROM tab
        IF j > 100 THEN
            DELETE FROM tab WHERE CURRENT OF cur1;
            CONTINUE FOREACH;
        END IF
        UPDATE tab SET c2 = c2 + 10 WHERE CURRENT OF cur1;
    END FOREACH
    FOREACH EXECUTE PROCEDURE bar(10,20) INTO i
        INSERT INTO tab2 VALUES (i);
    END FOREACH
END PROCEDURE; -- foreach_ex

```

A SELECT cursor is closed when any of the following situations occur:

- ▶ The cursor returns no further rows.
- ▶ The cursor is a SELECT cursor without a HOLD specification, and a transaction completes using COMMIT or ROLLBACK statements.
- ▶ An EXIT statement executes, which transfers control out of the FOREACH statement.

Database schema and data migration

A successful database migration depends highly on accurate planning in schedules, DBA resources, and the hardware infrastructure availability, such as disk space and CPU resources. Additional factors that can affect the success of a migration project include knowledge of the existing database objects, their sizes, relationships, and how the database and its objects are maintained on the source database server. A level of complexity is added if you take the migration as an opportunity to improve the existing design of the database, such as data fragmentation and storage layout. A good understanding of the various data movement methods for unloading, transforming, and loading the data consistently and efficiently is also a key factor to meet a company's requirement for a database migration.

In this chapter, we provide details about the database schemas and data migration, and we include practical examples using a demonstration database:

- ▶ Database schema generation on the SQL Server
- ▶ Database object implementation and SQL syntax differences
- ▶ Data extraction and data conversion
- ▶ Informix data loading facilities

5.1 Migration methodology

In this section, we focus our discussion on the technical requirements to successfully migrate the database objects and data from an SQL Server source database server to an Informix target database server. Figure 5-1 depicts an overview of the database and object migration process.

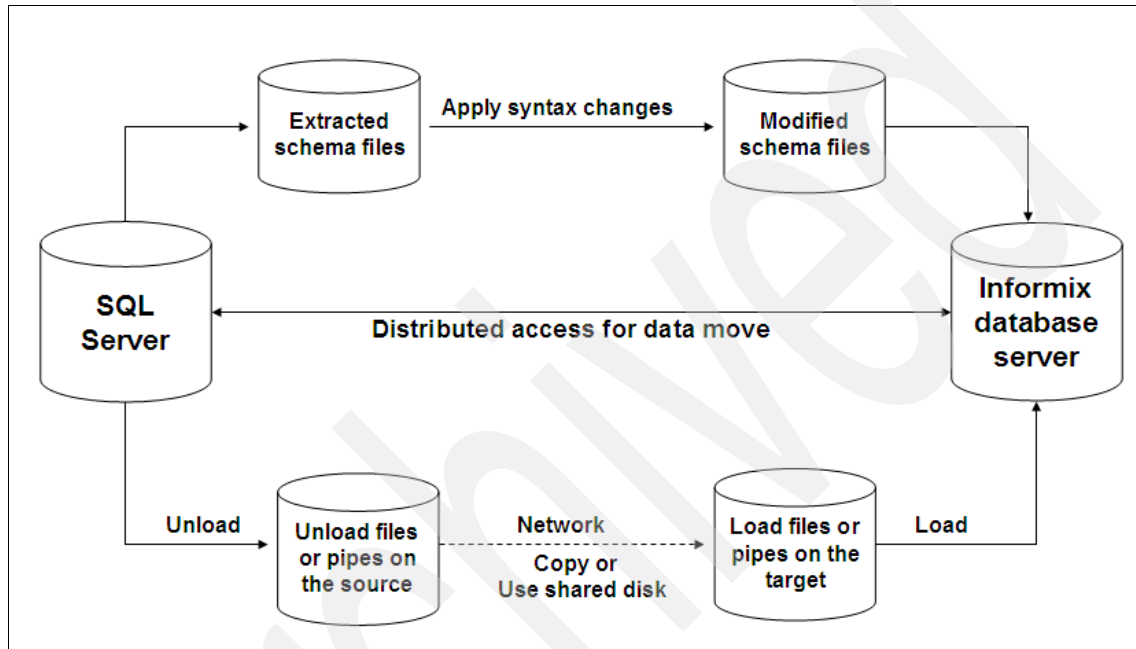


Figure 5-1 Database object and data movement process overview

To migrate the databases and applications from SQL Server to Informix, complete the following steps:

1. On the SQL Server database server:
 - a. Determine the databases to migrate.
 - b. Determine the database objects to migrate. The database objects include constraints, indexes, views, stored procedures, triggers, and security objects.
 - c. Extract database objects using SQL Server Management Studio, SQL Server Transact-SQL (T-SQL) scripts, or a migration tool, such as IBM Migration Toolkit.
 - d. Estimate the size of the database objects. Determine, based on the data size and the movement strategy, the required disk space for unloading

data. The disk space requirement differs if you plan to unload tables in parallel.

2. On the Informix database server:
 - a. Determine the target system hardware requirements, memory, storage, and CPUs.
 - b. Set up and initialize the target database system.
 - c. Define and set up the storage objects, such as dbspaces and chunks.
 - d. Define the transaction space, such as physical logs and logical logs.
 - e. Create the target databases.
3. On the SQL Server database server, extract the schema of the database objects to be moved.
4. On the Informix database server:
 - a. Modify the Data Definition Language (DDL) statements for Informix if there are syntax differences.
 - b. Add the appropriate target storage objects to the storage clause of the table and index DDL statements.
 - c. Consider applying data partitioning (also known as *fragmentation*) to your tables. Make the appropriate changes before processing the load. Depending on the load utility or strategy chosen, you can run load jobs in parallel.
 - d. Consider the following performance improvement actions in the new production environment if tables are fragmented:
 - Combine the fragmented tables with parallel database query (PDQ) queries for workload parallelization.
 - Choose a smart expression-based fragmentation to distinguish between historical data from the daily data or frequently used data and sporadically used data. This distinction allows the optimizer to apply fragment elimination to queries by skipping the access to the table data fragments that are not required.
 - e. Merge all the modified SQL statements into one batch file to create all the database objects in one step on the target database server.
 - f. Set the DELIMIDENT=y environment variable in the client environment if the object name or owner name is case sensitive.
 - g. Create the database objects in the empty database. Use a script file that contains all the SQL DDL statements, and run the SQL batch with the dbaccess utility.
 - h. Verify the success of the database object creation.

5. Consider utilizing a distributed database server access product, such as Informix Enterprise Gateway Manager, to transfer the data of smaller tables by inserting the data from the SQL Server database directly into the Informix database.
6. On the SQL Server side, unload the data with an appropriate unload strategy based on the table sizes, available disk space, and other system resources. Do not forget the following considerations:
 - A parallel data unload method is usually faster than a serial unload method.
 - If you prefer using the pipes instead of the flat unload files, set up an appropriate data copy strategy to transfer the data from the source to the target system.
 - Be aware of data type differences.
7. On the Informix database server:
 - a. Prepare the load jobs based on the selected load utility.
 - b. Automate and parallelize the jobs with the scripts.
 - c. Apply optimizer statistics for the data.
 - d. Consider data compression or encryption.
 - e. Set up a backup strategy, and incorporate the database backup into the company's existing storage management solution.
 - f. Schedule the common database maintenance tasks within OpenAdmin Tool (OAT) and the sysadmin database.
 - g. Migrate the database scripts and the client applications, and change the communication interfaces, such as Open Database Connectivity (ODBC) and Java Database Connectivity (JDBC).
 - h. Run functional tests for the applications.
 - i. Check the general application performance, such as the throughput.
 - j. Consider Informix business continuation solutions, such as shared disk secondary, remote stand-alone secondary, or high-availability data replication for your database infrastructure, to provide data reliability and workload partitioning.

In addition to these steps, consider the following factors that can affect a migration plan:

- Be aware that the unloading the data will affect the production environment. Schedule the unload jobs in the off-peak hours. Do not unload too many database objects in parallel with the production workload.

- ▶ Define a delta data extraction process if you cannot unload and load the data in one step. Most of the time, the source and target systems run in parallel for a certain time frame. Schedule multiple delta extraction and load processes, depending on your need.
- ▶ Select sample database objects and migrate them to a test environment:
 - Check if you are able to migrate the sample schema and the data successfully. Look for the correct definition of constraints and target data types specifically.
 - Check if data unloading and loading meet the required time constraints.
 - Calculate the required time for the complete migration based on the sample objects' migration.

These steps give a high-level overview of the process that is required for a successful migration. You need to estimate the time and the resource that are required for each step and define a fallback plan before you begin the migration project.

In the following sections, we explain how to move the database objects and data.

5.2 Database schema movement

The first stage in the database schema migration process is to identify the database objects that you need to move, such as tables, indexes, and object relationships (such as constraints, views, and triggers). This *schema movement* includes extracting the SQL DDL definition of each object from the entire database. After the extraction, you have to identify whether any SQL statement requires modification due to differences in syntax and semantics between two database systems. After all these changes, you can enable Informix to create the objects on the target server successfully. In this section, we focus on the migration process of table, index, view, sequence, roles, and privilege. For converting procedures, functions, and triggers, refer to Chapter 4, “SQL considerations” on page 67.

For our discussion, we assume the following environment:

- ▶ An empty Informix database server instance has been created and initialized.
- ▶ The required storage objects, dbspaces and chunks, are created.
- ▶ The physical and logical log sizes are defined. Note that you can adjust the log size at the end of the load based on the data distribution.

5.2.1 Creating a database

When creating a database in Informix, you need to define only the logging mode and the storage. The definition and initialization of the dbspaces and chunks are independent from the database creation. They can be shared across multiple databases. The storage clause in the database statement defines only initially where the system catalog resides. Each new table or index object can be relocated to the database catalog according to special storage needs.

When creating the database in Informix, make sure that you understand if and how your data is to be localized. Prepare the database with the appropriate code set to enable Informix to store all the character-based data in the expected internal ASCII representation. Use the DB_LOCALE and SERVER_LOCALE environment variables for the appropriate setting.

Example 5-1 shows the SQL statement for creating a database. The database is created in a dbspace named book_dbs.

Example 5-1 Create a database in Informix

```
export DB_LOCALE=en_us.8859-1
#export DB_LOCALE=de_DE.8859-15
dbaccess << EOF
CREATE DATABASE bookdb IN book_dbs;
-- in case you want to create a logging database, use the following statements
--CREATE DATABASE bookdb IN book_dbs WITH LOG;
--CREATE DATABASE bookdb IN book_dbs WITH LOG MODE ANSI;
EOF
```

For performance considerations in data migration, create the database without logging initially. This setting can save time in loading data into the database. After the data is loaded, you can then enable logging, as shown in Example 5-2.

Example 5-2 Enable the logging in an Informix database

```
#Change the ONCONFIG parameter TAPEDEV to dev/null and run
ontape -s -L 0 -B bookdb
```

5.2.2 Extracting table statements using SQL Server scripting facilities

The easiest method to extract the schema of all the table objects from the SQL Server is to use the SQL Server Management Studio. With this tool, you can generate the SQL DDL statements of particular tables or the entire database.

To extract the DDL for a particular table, use the following procedure:

1. Go to the ObjectExplorer, and expand your database and the tables folder.
2. Highlight your table, and right-click to open the context menu.
3. Select **Script table as**, and select **Create to**.
4. Redirect the output to a file.

Perform these tasks to extract the DDL of all the table objects of a particular database in one step:

1. Go to ObjectExplorer, and highlight your database.
2. Go to the Tasks tab, and select **Generate Script** to start the wizard.
3. Select your database from the list.
4. In the script options window, make sure that all the required objects are selected. Do not forget the indexes; they are not selected by default.
5. In the next window, select all the tables, schema, and view objects to be extracted.
6. Specify the file output according to your environment, and start the scripting.

After you have extracted the DDL, the file that you specified will contain the CREATE TABLE statements. If there are syntax or semantic differences between the two databases, you need to modify the statements so that the CREATE TABLE SQL statements can run successfully on Informix. Table 5-1 shows the CREATE TABLE statement syntax for SQL Server and Informix.

Table 5-1 Comparison of the CREATE TABLE statement syntax

SQL Server	Informix
<pre>CREATE TABLE [dbo].[authors]([author_id] [dbo].[t_id] NOT NULL, [name] [varchar](40) NOT NULL, [firstname] [varchar](20) NOT NULL, [phone] [char](12) NOT NULL, [address] [varchar](40) NULL, [city] [varchar](20) NULL, [state] [char](2) NULL, [zip] [char](5) NULL, [contract] [bit] NOT NULL)</pre>	<pre>CREATE TABLE "dbo".authors (author_id VARCHAR(11) NOT NULL , name VARCHAR(40) NOT NULL , firstname VARCHAR(20) NOT NULL , phone CHAR(12) DEFAULT 'n/a' NOT NULL , address VARCHAR(40), city VARCHAR(20), state CHAR(2), zip CHAR(5), contract INTEGER NOT NULL);</pre>

The following initial syntax changes are required:

- ▶ Remove the brackets on the column name and the data types.
- ▶ Change the brackets on the owner to parentheses.
- ▶ Remove the NULL clause for columns that allow the NULL value.
- ▶ If the object name or the owner name is case sensitive, set the name in parentheses. Set the DELIMIDENT=y environment variable at the object creation time on the Informix target.

Other possible table definition element changes that might be required for the table definition migration are data types, default values, the storage clause, and constraints. In the next sections, we discuss these particular table definition elements in more detail.

Data types

There are only a few data types available in SQL Server that do not have an equivalent type in Informix. In certain cases, you must change only the name of the data type; however, in other cases, you must find a matching value range.

Built-in data types

Table 5-2 lists all the built-in data types that require change. The data types that are not listed do not need to be changed.

Table 5-2 Data types that need to be replaced by another type on the target

Data type	SQL Server	Informix
Numeric data types	TINYINT BIT TIMESTAMP ROWVERSION IDENTITY	SMALLINT SMALLINT/BOOLEAN SERIAL/BIGSERIAL SERIAL/BIGSERIAL SERIAL/BIGSERIAL
Decimal	SMALLMONEY	MONEY
Date and datetime	DATETIME2 SMALLDATETIME DATETIMEOFFSET TIME	All as DATETIME YEAR TO FRACTION(5)
Binary and unstructured data	BINARY VARBINARY IMAGE TEXT	All as binary large object or BLOB (BYTE/TEXT) or character large object or CBLOB

User-defined data types

If the table definitions contain user-defined types, you must create the user-defined data types and change the statements. Example 5-3 depicts the differences in the definition of a simple user-defined type between SQL Server and Informix. You replace the `sp_addtype` statement with the `CREATE DISTINCT TYPE` statement.

Example 5-3 User-defined type definitions

```
-- Creating a new user-defined simple type in SQL Server
EXECUTE SP_ADDTYPE t_id , 'VARCHAR(11)' , 'NOT NULL'

-- Create a new simple user-defined data type in Informix
CREATE DISTINCT TYPE 'informix'.t_id AS      VARCHAR(11);
```

Example 5-4 shows how to create a row type that contains a set of columns in Informix.

Example 5-4 Creating a user-defined data type

```
-- Create a row type containing a set of columns
CREATE ROW TYPE 'informix'.location_t
(
    location_id SERIAL NOT NULL,
    loc_type CHAR(2),
    company VARCHAR(20),
    street_addr LIST(varchar(25) NOT NULL),
    city VARCHAR(25),
    country VARCHAR(25)
);
CREATE ROW TYPE 'informix'.loc_us_t
(
    state_code CHAR(2),
    zip ROW(code INTEGER, suffix SMALLINT),
    phone CHAR(18)
) UNDER location_t;
```

IDENTITY data type

There is no `IDENTITY` data type in Informix. However, Informix provides the `SERIAL` and `BIGSERIAL` data types, which implement similar functionality. You also can use the `SEQUENCE` data type to generate unique identity values for data. Unlike the `IDENTITY` and `SERIAL` types, the `SEQUENCE` type does not belong exclusively to the table. It can be used in general in the database server.

In addition to the direct data type mapping of both database servers, you can check the rich variety of additional data types in Informix, such as `SET`, `LIST`, and `MULTISET`. Informix provides a simple SQL interface for creating other new

types. Check the databases for additional available data types and defined functionality that extend the Informix functionality.

For a detailed table-based data type overview of SQL Server and corresponding types in Informix, refer to Appendix B, “Data types” on page 409. We also provide an in-depth discussion about the representation for certain data types and the required transformations on the source during the unload in 5.3.4, “Changing the data representation” on page 233.

Computed column

The computed column in a table definition is not implemented in Informix. You can implement this function easily by defining a new view or creating an insert and update trigger on the top of the table. Example 5-5 depicts the necessary definitions on Informix to apply a computed column.

Example 5-5 Implementing computed columns in Informix

```
/* Computed column in SQL Server
CREATE TABLE employee
(
    basic_salary    DECIMAL(8,2) NULL,
    commission      AS (basic_salary * 0.5)
)

-- Informix implementation
CREATE VIEW v_employee (basic_salary, commission)
AS SELECT commission, basic_salary * 0.5 AS commission FROM employee;

-- Update trigger
CREATE TRIGGER up_employee
UPDATE OF basic_salary ON employee
REFERENCING OLD AS pre NEW AS post
FOR EACH ROW
    (UPDATE employee SET comm=post.basic_salary *0.5 WHERE
     emp_id=post.emp_id);

--Insert trigger (make sure that you only update the target row
CREATE TRIGGER "informix".ins_emp INSERT ON "informix".employee
REFERENCING NEW AS newemployee
FOR EACH ROW
    (
        UPDATE employee
        SET employee.comm = (newemployee.basic_salary * 0.5 )
        WHERE (emp_id = newemployee.emp_id
    ) );
INSERT INTO employee ( basic_salary ) VALUES ( 1000.00 ) ;
```

Default values

In SQL Server, you define default values in two ways:

- ▶ Specify the value at the column level definition.
- ▶ Use an ALTER TABLE statement.

When you generate the table DDL script with SQL Server Management Studio, the default values are assigned to columns as ALTER TABLE statements immediately after all the table definitions. The ALTER TABLE statement clause for defining the default value differs between SQL Server and Informix. Therefore, you must either change the syntax of the ALTER TABLE statement or move the default value definition to the column definition in the CREATE TABLE statement.

Important: If the column also has a NOT NULL constraint, make sure that you apply both definitions in the expected order.

Example 5-6 shows the generated statement for the authors table from our SQL Server bookdb database and the corresponding Informix statements.

Example 5-6 Possible default value definitions in SQL Server and Informix

```
----- SQL Server
CREATE TABLE authors
(
    phone          CHAR(12)          NOT NULL DEFAULT ('n/a')
)

ALTER TABLE [dbo].[authors] ADD  DEFAULT ('n/a') FOR [phone]

----- Informix
-- Create table, specify the Default value on column level
CREATE TABLE "dbo".authors
(
    phone CHAR(12) DEFAULT 'n/a' NOT NULL
);

-- Alter table
ALTER TABLE "dbo".authors MODIFY ( phone CHAR(12) DEFAULT 'n/a' NOT NULL )
```

Constraints

You can have unique, not null, check, primary key, and foreign key constraints on a table. In the generated DDL script, depending on the constraint type, a constraint can be defined in the table on a column or a table level, or it can be applied with an ALTER TABLE statement.

In this section, we explain the syntax differences of the constraint definitions. When converting the SQL Server statements to Informix, perform the following changes:

- ▶ Remove the brackets in the generated SQL, and set the constraint owner name in parentheses for the target database server.
- ▶ Remove the CLUSTERED and NONCLUSTERED keywords (for unique and primary key constraints).
- ▶ Change the order of the constraint name and the constraint definition. SQL Server defines the constraint name first and then the definition. Informix defines the constraint in the reverse order.
- ▶ Change the ALTER TABLE statement. Informix requires only one statement for adding a constraint to a table. Replace the WITH CHECK ADD clause with the ADD CONSTRAINT clause.

Primary key

A *primary key* in a table defines one column or a set of columns that provide uniqueness and a not null guarantee on the table data, which assures that each row can be identified uniquely by specifying the key. You can define the primary key on a column level or on a table level. You also can apply the primary key with an ALTER TABLE statement.

Example 5-7 depicts the possible ways to create a primary key in SQL Server. The output from the SQL Server scripting facility generally shows the primary key definitions as a constraint clause on the table level. Syntactically, this clause is after the column list in the CREATE TABLE statement.

Example 5-7 Primary key constraint definitions in SQL Server

```
--Constraints in the create table statement
CREATE TABLE authors
(
    author_id t_id CONSTRAINT PK_author_id PRIMARY KEY CLUSTERED
)

--Constraints by the script generator from the Management Studio
CREATE TABLE [dbo].[authors](
    author_id    t_id
/* all other column definitions removed */
CONSTRAINT [PK_author_id] PRIMARY KEY CLUSTERED
(
    [author_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```



```
--Constraint created by an attached alter table statement  
ALTER TABLE [dbo].[authors] WITH CHECK ADD PRIMARY KEY([author_id])
```

Informix supports all of the primary key constraint definitions. You can name a constraint. If no name for a constraint is specified, an internally generated name is assigned. In contrast to the SQL Server, in Informix, the base columns of a primary key constraint do not require an explicit NOT NULL definition. Similar to the SQL Server, the primary key in Informix also is maintained as an index. Consider the primary key index as clustered at the creation time. Over the course of production activities, the key can lose the clustering with subsequent index operations, such as inserts, updates, and deletes. However, you can reestablish the cluster manually with an SQL statement. To migrate the CREATE TABLE statement, remove the CLUSTERED clause.

Example 5-8 shows the various examples of the primary key constraint definitions in the Informix server.

Example 5-8 Constraint definitions in Informix

```
-- Table level primary key definitions, for example, key contains more than one  
-- column  
CREATE TABLE "dbo".authors  
(  
    author_id VARCHAR(11) NOT NULL ,  
    PRIMARY KEY (author_id) CONSTRAINT "dbo".pk_author_id  
);  
-- Column level primary key definitions  
CREATE TABLE "dbo".authors  
(  
    author_id VARCHAR(11) PRIMARY KEY  
);  
  
-- PK Constraint definition with Alter table  
ALTER TABLE dbo.author ADD CONSTRAINT PRIMARY KEY (author_id)  
    CONSTRAINT "dbo".pk_author_id
```

Unique constraint

In general, you can define the unique constraints using the methods for defining the primary keys that we described in the previous section. Similar to primary keys, the unique constraint is maintained internally as an index. The syntax that the scripting table generates looks similar to that shown in Example 5-9 on page 206 for both database servers.

Example 5-9 Unique constraint definitions in comparison

```
/* SQL Server examples for unique constraints

CREATE TABLE [dbo].[authors](
  author_id      t_id,
  UNIQUE NONCLUSTERED
  (
    [author_id]
  ))

ALTER TABLE [dbo].[authors] WITH CHECK ADD UNIQUE([author_id])

-- Unique constraint definition in Informix
-- Table level
CREATE TABLE "dbo".authors(
  author_id t_id,
  UNIQUE ( author_id )
);

-- Column level
CREATE TABLE "dbo".authors(
  author_id t_id UNIQUE
);

-- Alter table
ALTER TABLE dbo.authors ADD CONSTRAINT UNIQUE ( author_id ) CONSTRAINT
uniq_author;
```

To migrate the unique constraints definition, change the syntax. Remove the **NONCLUSTERED** and **CLUSTERED** clauses from the constraint definition in the **CREATE TABLE** statement. If you use the **ALTER TABLE SQL** statement, replace the **WITH CHECK ADD** clause with an **ADD CONSTRAINT** clause.

Check constraint

The check constraint statements from the SQL Server scripting utility are created as a set of two **ALTER TABLE** statements. The check specification follows the column definition in the **CREATE TABLE** statement. Example 5-10 shows the check constraint DDLs for both SQL Server and Informix.

Example 5-10 Check constraint comparison

```
/* SQL Server */
/* default in scripting */
ALTER TABLE [dbo].[authors] WITH CHECK ADD CONSTRAINT [CK_author_id] CHECK
(((author_id like ' [0-9] [0-9] [0-9] - [0-9] [0-9] - [0-9] [0-9] ')))
GO
ALTER TABLE [dbo].[authors] CHECK CONSTRAINT [CK_author_id]
```

GO

```
/* create check constraint in the table itself */
CREATE TABLE authors
(
    author_id      t_id CONSTRAINT CK_author_id CHECK (
        author_id LIKE '[0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9]' )
)

-- ON Informix
-- Inside the table
CREATE TABLE "dbo".authors
(
    author_id VARCHAR(11) not null ,
    check (author_id LIKE ' 0-9 0-9 0-9 - 0-9 0-9 - 0-9 0-9 ' )
);

-- Using an alter table
ALTER TABLE dbo.authors ADD CONSTRAINT
CHECK (zip LIKE (' 0-9 0-9 0-9 0-9 0-9 ' )) CONSTRAINT check_zip
```

NOT NULL constraint and NULL constraint

The NOT NULL constraint is a special form of check constraint. The constraint definition clause for NOT NULL follows the column definition in the CREATE TABLE statement on both database servers. Handling and syntax are the same in SQL Server and Informix.

However, the NULL specification for a column differs in the two database servers. If you have columns defined to allow nulls, the SQL Server scripting generates the NULL keyword in the column description. Informix does not accept the NULL keyword. All columns that are not defined as NOT NULL are defined, by default, as allowing NULL values. No further definition is required on Informix. You must remove the NULL keyword from the CREATE TABLE statement.

Referential constraint

A *referential constraint* defines a relationship on one or more columns between two tables. The table that has the columns to where the reference is defined is the *master*. On the master, the reference columns have a primary key definition. The other table behaves as a *subordinate* and has a foreign key. Under a foreign key, the database server defines an index on the table. The primary and foreign key relationship ensures that all the data in the subordinate foreign key columns always has a corresponding value in the parent. During an insert and update, you are not allowed to insert or change values that do not exist on the primary table.

SQL Server provides four options for the DELETE referential constraints:

- ▶ CASCADE
- ▶ NO ACTION (default)
- ▶ SET NULL
- ▶ SET DEFAULT

Informix supports the CASCADE and NO ACTION options. The NO ACTION option is the default. Thus, if an application attempts to delete a row on the master table but there is still a row with the same value in the subordinate table, an error is returned to the application. In the case of the CASCADE option, the row on the subordinate is deleted also.

Example 5-11 shows referential constraint definitions for SQL Server and Informix. You can define the constraint on the column level, the table level, or in an ALTER TABLE statement. You are required to change the ALTER TABLE statement. Replace the WITH CHECK ADD clause with the ADD CONSTRAINT clause on the Informix side. Enclose the foreign constraint definition clause with parentheses. For the NO ACTION and CASCADE options, the column-level and table-level constraint definitions are the same on both sides.

Example 5-11 Create a foreign key constraint

```
/* SQL SERVER */
ALTER TABLE [dbo].[booktab] WITH CHECK ADD FOREIGN KEY([author_id])
REFERENCES [dbo].[authors] ([author_id])
CREATE TABLE [dbo].[booktab]
(
  author_id int NOT NULL REFERENCES authors(author_id) ON DELETE CASCADE
)

-- Informix
-- Column level constraint definition
CREATE TABLE "dbo".booktab (
  author_id INTEGER
  REFERENCES "dbo".authors(author_id)
)

--Table level constraint definition
CREATE TABLE "dbo".booktab (
  author_id INTEGER NOT NULL,
  FOREIGN KEY(author_id)
  REFERENCES "dbo".authors(author_id) ON DELETE CASCADE
)

-- Define a referential constraint with an Alter Table statement
ALTER TABLE "dbo".booktab ADD CONSTRAINT (FOREIGN KEY(author_id)
REFERENCES "dbo".authors(author_id))
```

Storage options

When you move the table definition's SQL from one database server to another database server, you also can consider the definition of the final table layout. SQL Server refers to this final table layout as a *table partition*, and Informix refers to it as *table fragmentation*. The Informix database server supports the table fragmentation by the round-robin method. Incoming rows are applied to the specified fragments in a balanced order but not by a certain fragmentation criteria based on the table data.

However, Informix also provides the fragmentation by expression strategy where you can distribute incoming data across various storage objects based on a certain expression. An often used example is the time-based expression. The newest data is clustered in one fragment, and historical data is kept in other fragments because of legal requirements.

Having the appropriate definitions of the fragmentation schema is beneficial in the following areas:

- ▶ You might benefit from a parallelized load through external tables and IBM Informix High-Performance Loader jobs.
- ▶ Applications can use the parallel database query (PDQ) parallelization facilities to have multiple readers scanning the table fragments in parallel. Answer times can be increased significantly if the underlying storage objects are defined on separate disks and if the I/O can be applied in parallel.
- ▶ You might benefit from a database server optimizer-based fragment elimination if your fragmentation schema defines an elimination strategy. For example, eliminate historical data from a table so the highly requested recent data can be accessed faster.

Example 5-12 shows the storage options in Informix.

Example 5-12 Possible fragmentation definitions in a CREATE TABLE clause in Informix

```
-- Time based fragment schema for fragment elimination and parallel I/O
CREATE TABLE "dbo".bookdb (
  release_date DATE )
FRAGMENT BY EXPRESSION
release_date < '12/31/2000' IN dbs_old,
release_date < '12/31/2001' IN dbs_2001,
release_date < '12/31/2002' IN dbs_2002,
release_date < '12/31/2003' IN dbs_2003,
release_date >='01/01/2004' IN dbs_new ;

-- Round robin based fragment schema for parallel I/O in data read
CREATE TABLE "dbo".authors
  ( author_id VARCHAR(11) NOT NULL ,
)
```

```
FRAGMENT BY ROUND ROBIN IN author_dbs1,author_dbs2,author_dbs3;
```

You have to remove the storage options for the table, such as volume groups and partition definitions, that are defined on the SQL Server. These options are not compatible with Informix. If you want to apply a one-to-one mapping on the storage options, map the volume group to a target dbspace in Informix, and then apply the dbspace in the storage clause. Example 5-13 shows how to map the storage for tables.

Example 5-13 Storage mapping for tables

```
/* SQL Server */
CREATE TABLE [dbo].[brands](
    [brand_id] [int] IDENTITY(1,1) NOT NULL,
    [brand_name] [varchar](20) NOT NULL,
    [brand_desc] [varchar](80) NOT NULL,
) ON [PRIMARY]

C:\temp>onstat -d
IBM Informix Dynamic Server Version 11.50.TC6      -- On-Line -- Up 22:27:29 --

Dbspaces
number  flags      fchunk  nchunks  pgsz  flags  owner  name
1       0x60001    1       1        4096  N B    informix rootdbs
2       0x60001    2       1        4096  N B    informix ol_ids_1150_1
2 active, 2047 maximum

Chunks
chunk/dbs  offset  size    free  flags  pathname
1      1      0      51200  16648  PO-BD  C:\ol_ids_1150_1\rootdbs_dat
2      2      0      25600  25547  PO-BD  C:\ol_ids_1150_1\dbs_data
2active, 32766 maximum

-- Informix with a simple storage clause without fragmentation
-- Data for the table are finally stored in the "ol_ids_1150_1" dbspace
CREATE TABLE "dbo".brands(
    brand_id SERIAL NOT NULL,
    brand_name VARCHAR(20) NOT NULL,
    brand_desc VARCHAR(80) NOT NULL,
) IN ol_ids_1150_1
```

5.2.3 Generating table statements with a T-SQL script

There are benefits in using the SQL Server scripting facility to extract the object schema for a certain table or the entire database. You can use SQL Server Management Studio to generate the output to a file. However, creating the SQL

Server schema is only the first step in schema movement. After you create the schema, you must apply all your changes manually, table by table, object by object, as we explained in the previous sections.

It is more efficient to have a script that can create output in the format that the Informix database server expects it. Example 5-14 provides a sample T-SQL script that creates a CREATE TABLE statement in the Informix SQL language. You can use this sample code as a starting point to develop a migration script with full conversion functions.

Example 5-14 T-SQL sample script for creating an Informix-based CREATE TABLE DDL

```
USE bookdb;
GO

DECLARE @informix_table VARCHAR(128)
SET @informix_table = 'authors'
DECLARE @schema table ( line VARCHAR (500) , count INT IDENTITY)
DECLARE @owner VARCHAR(128)
DECLARE @maxcol INTEGER

SELECT @owner =table_schema FROM information_schema.columns WHERE
    table_name= @informix_table
SELECT @maxcol= MAX(ordinal_position) FROM information_schema.columns WHERE
    table_name= @informix_table
INSERT INTO @schema (line) VALUES
    ( 'CREATE TABLE ' +''''+@owner+'\".' + @informix_table +' (')

INSERT INTO @schema
SELECT column_name + ' ' + REPLACE(data_type,'bit','integer') + ' ' +
    CASE WHEN character_maximum_length>0 THEN '('+
    CAST(character_maximum_length AS VARCHAR) +')'
    ELSE ' ' END + ' ' +
    CASE WHEN column_default IS NOT NULL THEN 'DEFAULT ' +
        REPLACE(REPLACE(column_default,',' , ' '),')', ' ') + ' '
        ELSE ' ' END +
    CASE WHEN is_nullable = 'No' THEN ' NOT NULL ' ELSE ' ' END +
    CASE WHEN ordinal_position = @maxcol THEN ' ' ELSE ',' END
FROM information_schema.columns
WHERE table_name=@informix_table

DECLARE @pk VARCHAR(128)
SELECT @pk = constraint_name FROM information_schema.table_constraints
    WHERE table_name= @informix_table AND constraint_type='PRIMARY KEY'

IF ( @pk IS NOT NULL )
BEGIN
```

```

SELECT @maxcol= MAX(ordinal_position) FROM information_schema.key_column_usage
WHERE
    table_name= @informix_table

INSERT INTO @schema VALUES (', PRIMARY KEY ( ' )

INSERT INTO @schema
SELECT column_name +
    CASE WHEN ordinal_position= @maxcol THEN ' ' ELSE ' ,' END
    FROM information_schema.key_column_usage
    WHERE table_name=@informix_table

INSERT INTO @schema (line) VALUES ( ' ) CONSTRAINT ' +@pk)
END

INSERT INTO @schema
SELECT ', CHECK ' + REPLACE(REPLACE( b.check_clause,'[',' '),']',' ')

    FROM information_schema.table_constraints AS a,
        information_schema.check_constraints AS b
    WHERE table_name= @informix_table AND constraint_type='CHECK'
        AND a.constraint_name=b.constraint_name

INSERT INTO @schema (line) VALUES (');')
SELECT line FROM @schema ORDER BY count

```

The script first examines the data types for all the columns from the system catalog. As an example, we substitute the bit data type with integer. Default values are examined, and the NOT NULL constraints are added to the table definition for the required columns. The primary key constraints and the check constraints follow the column definition. You can refine the script to move the T-SQL code into a stored procedure and add the unique constraint and referential key support.

Example 5-15 shows the output for the Informix-compatible CREATE TABLE statement.

Example 5-15 Output of a converted CREATE TABLE statement in Informix SQL

```

CREATE TABLE "dbo".authors(
    author_id varchar(11) NOT NULL,
    name varchar(40) NOT NULL,
    firstname varchar(20) NOT NULL,
    phone char(12) default 'n/a' NOT NULL,
    address varchar(40),
    city varchar(20),
    state char(2),
    zip char(5),

```



```

        contract integer NOT NULL
    ,PRIMARY KEY(
        author_id
    ) CONSTRAINT PK_author_id ,
    CHECK ( zip like ' 0-9 0-9 0-9 0-9 0-9 ' ) ,
    CHECK(author_idlike'0-90-90-9-0-90-9-0-90-90-9')
);

```

5.2.4 Extracting index statements using SQL Server scripting facility

Indexes logically belong to the table objects. They also consume disk space, and for that reason, you need to apply the same attention in deciding where to store the index data. You can create an index explicitly using the `CREATE INDEX` statement, or you can create an index implicitly using constraints, such as a primary key, unique key, or foreign key.

To generate the `CREATE INDEX` statements on a certain table using SQL Server Management Studio, you must select the index that you want manually. If the table contains many indexes, it is easier to extract the DDLs for the entire database rather than extract each index individually. To extract the indexes with the database DDLs, you must enable the scripting for the index statements in the options window. Index definitions are not generated by default.

Clustered and non-clustered indexes

Both database servers support clustered and non-clustered indexes. In SQL Server, you must define the cluster type of an index within the `CREATE INDEX` statement when the index is created.

Informix does not force the index to be clustered all the time. The index is clustered initially at creation. However, the index loses clustering gradually after new rows are inserted or deleted or after indexed columns are updated. You can use the `ALTER INDEX TO CLUSTER` statement to reorganize the data rows to be stored in the way that the index is maintained.

For a better illustration of the differences of index definition between SQL Server and Informix, Example 5-16 shows the `CREATE INDEX` statements of these two database servers.

Example 5-16 CREATE INDEX statement comparison

```

/* SQL Server */
CREATE NONCLUSTERED INDEX [authors_idx] ON [dbo].[authors]
(
    [author_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]

```

```
-- Informix
CREATE INDEX authors_idx ON authors ( author_id ASC );
CREATE UNIQUE INDEX authors_idx_unq ON authors ( author_id );

ALTER INDEX authors_idx TO CLUSTER ;
```

Ascending and descending sort order

Both database servers allow an ascending or a descending sorting order of the index columns. The specification of the sorting order for particular columns is similar. No syntax changes are required for this option.

Unique and duplicate indexes

Unique and duplicate indexes are defined in the same way and in the same syntax on SQL Server and Informix. Both database servers define the duplicate index as the default. The unique index is specified by an additional keyword in the CREATE INDEX statement.

Additional options

SQL Server includes many additional options within the CREATE INDEX statement, which you can remove. In Informix, creating an index includes sort operations. If the memory that the sort requires is insufficient, rows are written to the defined temporary dbspaces. Indexes can be enabled or disabled using an SQL statement. Index locks depend on the index structure and the isolation level set. In Informix, indexes can be built online and offline. Thus, building an index online does not block parallel work on the table.

In general, indexes are built in parallel for performance reasons. In addition, building an index forces a renewal of the indexed column statistics that are used by the SQL optimizer. Because these operations are tied together closely, they do not require extra time.

Storage options

You can apply the migration consideration for a table's storage options to the index. In general, the index follows the storage definition of the table. However, a DBA or any database user with the authority can detach the index from the table and define separate storage fragments for the index.

The storage options for the indexes that are defined on the SQL Server are incompatible with Informix and must be removed from the statement. Example 5-17 on page 215 shows an SQL Server index statement and the detailed storage specifications for Informix.

Example 5-17 Index storage determination and usage in Informix

```
/* SQL Server */
CREATE NONCLUSTERED INDEX [authors_idx] ON [dbo].[authors]
(
    [author_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]

# Storage mapping in Informix
C:\temp>onstat -d
IBM Informix Dynamic Server Version 11.50.TC6      -- On-Line -- Up 22:27:29 --

Dbspaces
number  flags      fchunk  nchunks  pgsz   flags  owner  name
1       0x60001    1       1        4096   N B    informix rootdbs
2       0x60001    2       1        4096   N B    informix ol_ids_1150_1
2 active, 2047 maximum

Chunks
chunk/dbs  offset  size    free  flags  pathname
1          1      0      51200 16648 PO-BD C:\ol_ids_1150_1\rootdbs_dat
2          2      0      25600 25547 PO-BD C:\ol_ids_1150_1\dbs_data
2 active, 32766 maximum

-- Informix index creation not fragmented
CREATE INDEX authors_idx ON authors ( author_id asc ) IN ol_ids_1150_1;
-- Informix index creation fragmented
CREATE INDEX zip_idx ON authors ( zip )
PARTITION area1 (zip< 10000) IN ol_ids_1150_1,
PARTITION area2 (zip< 20000) IN ol_ids_1150_1,
PARTITION area3 (zip< 30000) IN ol_ids_1150_1,
PARTITION area4 (zip< 40000) IN ol_ids_1150_1,
PARTITION area5 (zip<50000) IN ol_ids_1150_1,
PARTITION rest REMAINDER IN ol_ids_1150_1,
;
```

5.2.5 Generating index statements with a T-SQL script

Similar to generating the CREATE TABLE statements in Informix syntax format, here we provide a script for generating the index schema for a given table using T-SQL and the system tables in the SQL Server. Example 5-18 shows the complete T-SQL sample code.

Example 5-18 Generating the CREATE INDEX statements for Informix

```
USE bookdb
```

```

GO

DECLARE @informix_table VARCHAR(128)
SET @informix_table = 'authors'
DECLARE @idx_schema table ( line VARCHAR (500) , count INT IDENTITY)

DECLARE @idxname VARCHAR(128),
        @indexid INTEGER, @isuniq INTEGER,
        @object_id INTEGER, @descr INTEGER, @card INTEGER,
        @colname VARCHAR(128),@message VARCHAR(1000)

DECLARE idx CURSOR FOR
SELECT a.name, a.object_id,a.index_id,a.is_unique
FROM
sys.indexes AS a ,sys.tables AS b
WHERE b.name= @informix_table
AND is_primary_key=0
AND is_unique_constraint= 0
AND a.object_id=b.object_id
AND a.name IS NOT NULL

OPEN idx

FETCH NEXT FROM idx INTO @idxname, @object_id, @indexid,@isuniq

WHILE @@FETCH_STATUS = 0
BEGIN

    SET @message=' CREATE '

    IF ( @isuniq >0)
    BEGIN
        SET @message = @message + ' UNIQUE '
    END

    SET @message=@message + ' INDEX ' + @idxname + ' on ' + @informix_table + '
( '

    INSERT INTO @idx_schema values (@message)

    DECLARE col_name CURSOR FOR

    SELECT b.name, a.is_descending_key, a.key_ordinal
    FROM sys.index_columns AS a , sys.columns AS b
    WHERE
    b.object_id=a.object_id
    AND a.object_id=@object_id
    AND a.index_id=@indexid

```

```

AND a.column_id=b.column_id
ORDER BY a.key_ordinal

OPEN col_name

SET @message=''

FETCH NEXT FROM col_name INTO @colname, @descr, @card

WHILE @@FETCH_STATUS = 0
BEGIN

    SET @message= @message + @colname
    IF ( @descr > 0 )
    BEGIN
        SET @message= @message + ' desc '
    END

    FETCH NEXT FROM col_name INTO @colname, @descr, @card

    if ( @@FETCH_STATUS = 0 )
        SET @message= @message + ','

END

CLOSE col_name
DEALLOCATE col_name

INSERT INTO @idx_schema VALUES ( @message + ' ');')

FETCH NEXT FROM idx INTO @idxname, @object_id, @indexid,@isuniq

END
CLOSE idx
DEALLOCATE idx

SELECT line FROM @idx_schema

```

The script determines the name and the type of the index and whether the index is unique or duplicate. The script uses the content in the sys.indexes table. The script also examines the sort order and whether the index column names abide by the Informix naming rules. Example 5-19 shows a sample output that the script generates.

Example 5-19 Sample CREATE INDEX statements generated by the T-SQL script

```

CREATE UNIQUE INDEX author_unq ON author ( author_id );
CREATE INDEX author_dup ON author ( name );

```

5.2.6 Implementing the views

Think of *views* as virtual tables. The view definition is based on a SELECT statement on one or more tables that are connected by a join. Querying data from views triggers the execution of the underlying SELECT statement and a result set is returned as the view. Data is not precomputed. In general, the structure of the CREATE VIEW statement is similar between SQL Server and Informix. Table 5-3 shows a comparison of the CREATE VIEW statements for SQL Server and Informix. Note the additional brackets that are generated by the script generator.

Table 5-3 View definition statement comparison

SQL Server	Informix
<pre>CREATE VIEW [dbo].[v_books] AS SELECT title, authors.name, brand_name, price, release_date FROM authors, booktab, brands WHERE authors.author_id = booktab.author_id AND booktab.brand_id = brands.brand_id CREATE VIEW v_books AS SELECT title, authors.name, brand_name, price, release_date FROM authors, booktab, brands WHERE authors.author_id = booktab.author_id AND booktab.brand_id = brands.brand_id</pre>	<pre>CREATE VIEW v_books AS SELECT title, authors.name, brand_name, price, release_date FROM authors, booktab, brands WHERE authors.author_id = booktab.author_id AND booktab.brand_id = brands.brand_id</pre>

You have to review the SELECT statement of the view definition carefully for the built-in functions that are available only in SQL Server. Check also the table join constructions, such as CROSS APPLY, that are not supported in Informix. You can replace these constructions with constructions, such as CONNECT BY. For built-in function mapping information, refer to Appendix C, “Function mapping” on page 419.

Example 5-20 on page 219 shows a query that allows you to extract the view definition from SQL Server using T-SQL. The script removes the brackets from the column and table descriptors. Additional manual efforts for checking the SELECT clause of the view are required.

Example 5-20 Select the view definition in the SQL Server with an SQL script

```
USE bookdb
GO

SELECT REPLACE(REPLACE(view_definition,'[',' ') ,']',' ')
FROM information_schema.views
```

5.2.7 Implementing the schema

The *schema* is the conceptual owner of database objects, such as views, tables, and permissions. The purpose of a schema and the schema management in the SQL language are the same in both database servers. In SQL, you have to create a schema explicitly. You cannot create objects with a non-existing schema. All the objects created without a schema specified in the DDL statements are assigned the schema name that was created most recently. You can assign a schema as the owner to new objects that are created in separate sessions. You only need to specify the schema as the owner in the create SQL statement.

Except for the view, SQL Server does not have a strict rule in the order of creating objects and referencing an object. You can reference an object before it is created.

Example 5-21 shows an example for the schema behavior in SQL Server. All the created objects are assigned to the bookdb schema.

Example 5-21 Schema and schema management in SQL Server

```
CREATE SCHEMA bookdb
GRANT SELECT ON authors TO public

CREATE VIEW v_rebook
AS
SELECT authors.author_id
FROM authors, booktab
WHERE authors.author_id = booktab.author_id

CREATE TABLE authors ( author_id t_id )
CREATE TABLE booktab ( author_id t_id )

/* attempt to create a table owned by a notexisting user */
/* return an error */
CREATE TABLE "not_exiting".authors ( author_id t_id )
```

The schema implementation in Informix is similar to SQL Server. In Informix, a recently created schema name is assigned to an object that has no schema

specified explicitly in the DDL statement. However, you are not required to create a schema explicitly with a CREATE SCHEMA statement. You can specify a non-existing schema as the object owner directly in the object create DDL statement, which is implicitly schema creation. Informix will create the schema for you. When referencing an object, if the object owner is not specified, the default owner is the current database user.

In Informix, the order of creating an object and referencing the object is strict. You cannot reference a nonexistent object. For example, in SQL Server, you can grant a permission to a table that is not created yet, which is not allowed in Informix. Thus, with Informix, you must consider the creation order of the database objects. For instance, you cannot grant a permission to an object that is not yet created.

Example 5-22 shows the schema creation and the assignment of the subsequently created objects in Informix.

Example 5-22 Schema definition and usage in Informix

```
CREATE SCHEMA AUTHORIZATION bookdb

CREATE TABLE authors ( author_id t_id )
CREATE TABLE booktab ( author_id t_id )
CREATE VIEW v_rebook
AS
SELECT authors.author_id
FROM authors, booktab
WHERE authors.author_id = booktab.author_id

GRANT SELECT ON authors TO public;

-- attempt to create a table owned by a notexisting user
-- successful
CREATE TABLE "not_exiting".authors_n ( author_id t_id )

-- Verification of the objects in the system catalog
SELECT tabname , owner FROM systables WHERE tabname IN
("authors","booktab","v_books");
tabname      owner
authors      bookdb
booktab      bookdb
v_rebook     bookdb
authors_n    not_exiting
```

If you extract the DDL of a database with the SQL Server script generator, you might see an SQL statement, such as CREATE SCHEMA [bookdb] AUTHORIZATION [dbo], in the output file. You can remove this SQL statement

from the file, because all the objects in the script are assigned already to an appropriate schema object.

5.2.8 Granting privileges and roles

In Informix, the permissions that are granted to users or roles are maintained in the system catalog. You must distinguish the database permissions from the database object permissions. For the database object-level permissions, such as the permissions for tables and views, the syntax of the GRANT and REVOKE SQL statements is almost the same. On the database level, the SQL Server has a much finer granularity of the permissions than Informix. You can grant permissions for almost every database and database object. In Informix, you can grant permissions only for database access, object creation, and DBA permissions. Permissions for backup and restore are handled by roles on the operating system user level in the Informix database server. By default, the user *informix* is granted the supervisor role. You can set up the system to split the roles among separate logins but not on the SQL level.

Table 5-4 shows a comparison of the permissions and how to grant the permissions.

Table 5-4 *Permission comparison*

SQL Server	Informix
Permissions on the database level: GRANT CREATE DATABASE TO guest GRANT CREATE TABLE TO guest GRANT CREATE VIEW TO guest GRANT CREATE PROCEDURE TO guest Permissions for database objects: GRANT EXECUTE ON [avg_dept_salary] TO [public] AS [dbo] GRANT DELETE ON [dbo].[authors] TO [guest] AS [dbo] GRANT INSERT ON [dbo].[authors] TO [guest] AS [dbo]	Permissions on the database level: GRANT DBA TO dbauser GRANT CONNECT TO connectuser GRANT RESOURCE TO resourceuser Permission at the object level: GRANT SELECT ON authors TO selectuser GRANT ALTER ON autors TO alteruser GRANT UPDATE ON authors TO updateuser GRANT ALL ON authors TO role_all AS informix REVOKE INSERT ON authors FROM insertusr REVOKE EXECUTE ON avg_dept_salary FROM public AS informix

In both database servers, you can create roles. You can apply permissions to roles, grant user accounts to roles, and generate role trees. The Informix and SQL Server methods of assigning roles to a user differ. Informix uses the GRANT statement to grant a role to a user or a role:

```
GRANT <lowerlevel_role>|<user> TO <higherlevel_role>
```

SQL Server uses a stored procedure:

```
sp_addrolemember('higher','lower')
```

You must replace this stored procedure in the SQL schema file with the Informix GRANT statement. Informix advances the role implementation with the default role functionality. At connection time, a database user is assigned a default role automatically. The default roles support project driven environments. Users are assigned to roles, and the role represents the current project on which they are working. All permissions to the underlying database objects are made to the role, not to the users. After the project is complete, the user is revoked from the role. This process simplifies security management.

5.2.9 Creating synonyms

Synonyms are used to create shortened names for local or remote database objects, such as tables or views. The syntax to create synonyms in both database servers is about the same. The object reference syntax differs:

- ▶ SQL Server identifies the remote objects with the following syntax:

```
[instance].[database].[schema].[object]
```

- ▶ Informix refers to the remote objects with the following syntax:

```
database@server:owner.object
```

The only functional difference between the two database servers is that SQL Server allows synonym references on functions, but Informix allows synonyms for sequences.

Example 5-23 shows the T-SQL code to generate the CREATE SYNONYM statements in Informix syntax from a given database.

Example 5-23 Generate the CREATE SYNONYM statements in the Informix SQL syntax

```
USE bookdb
GO
```

```
DECLARE @synonym_name VARCHAR(128)
DECLARE @final_name VARCHAR(128)
DECLARE @referred_object VARCHAR(128)
DECLARE @temp_referred_object VARCHAR(128)
```

```

DECLARE @cnt INTEGER
DECLARE @numpoints INTEGER
DECLARE @schema TABLE ( line VARCHAR(500))

DECLARE syn CURSOR FOR SELECT name , base_object_name FROM sys.synonyms

OPEN syn
FETCH NEXT FROM syn INTO @synonym_name, @refered_object

WHILE @@FETCH_STATUS = 0
BEGIN

SET @refered_object= REPLACE( REPLACE(@refered_object,']',''), '[','')

SET @temp_refered_object = @refered_object + '?';
SET @cnt = 1;
SET @numpoints = 0;

WHILE ( SUBSTRING(@temp_refered_object,@cnt,1) <> '?' )
BEGIN
PRINT SUBSTRING(@temp_refered_object,@cnt,1)
IF SUBSTRING(@temp_refered_object,@cnt,1) = '.'
BEGIN
SET @numpoints= @numpoints + 1
END
SET @cnt = @cnt + 1;
END

SET @final_name=@refered_object

IF @numpoints=2
BEGIN
SET @cnt=1
WHILE ( SUBSTRING(@refered_object,@cnt,1) != '.' )
begin
SET @cnt = @cnt + 1
END
SELECT @final_name=stuff(@refered_object,@cnt,1,':')
END

INSERT INTO @schema VALUES ( 'CREATE SYNONYM ' + @synonym_name + ' FOR ' +
@final_name + ' ; ' )

FETCH NEXT FROM syn INTO @synonym_name, @refered_object

END
CLOSE syn
DEALLOCATE syn

```

5.2.10 Rules for migrating schema files to Informix

In previous sections, we discussed the database objects and focused on the SQL statement and functional differences between SQL Server and Informix. We provided information about the SQL statements that you receive in an SQL script. In addition, we explained the statements that you needed to modify and how to modify those statements so that the scripts can run on an Informix database server successfully.

The following list is a summary of the changes that we identified when we migrated our sample `bookdb` database schema from SQL Server to Informix. You can use this list as a general checklist of the changes that are required for a database schema migration. Depending on your own environment, you might need to add other syntax replacements.

- Remove unnecessary SQL statements.

Remove the following statement types from your DDL script:

- GO statement
- All types of the SET statements
- All types of login and user statements, such as `create user`, `create login`, and `alter login`
- All types of `create schema` statements
- The `ALTER TABLE [schema].[name] CHECK CONSTRAINT` statement

- Change the following syntax:

- The DDL statement syntax from SQL Server either follows the American National Standards Institute (ANSI) standard or has brackets around the owner names, object names, data types, and most clauses, for example:

```
GRANT UPDATE ON [dbo].[book_history] TO [guest] AS [dbo]
GRANT CREATE TABLE TO guest
```

- You can remove all the brackets by a script. You also can develop a T-SQL job, which is based on the code sample that we provide, to generate the DDL in the Informix SQL syntax directly.
- In case you have to use case-sensitive user and database object names, replace the brackets with parentheses. Set the `DELIMIDENT=y` environment variable either in the client or server environment in order to keep the names case sensitive in Informix.

- ▶ Remove unsupported SQL clauses:
 - Remove the NULL clause from the data type definition of the table definitions.
 - Remove the options clause from the index definitions.
 - Remove the storage clauses from the index, constraint, and table definitions.
 - Remove the NONCLUSTERED and CLUSTERED clause from the index and constraint definitions.
- ▶ Review and replaced the unsupported data types with the equivalent data types based on your needs:
 - Check for the BIT and TINYINT data types.
 - Check for the IDENTITY, ROWVERSION, and TIMESTAMP data types.
 - Check for the DATETIME2, SMALLDATETIME, and TIME data types.
 - Replace the VARBINARY data type with the appropriate BLOB or CLOB data type.
- ▶ Apply changes to the statement clause order or syntax:
 - Check the constraint definition. The order specified differs in the two database systems.
 - Replace or remove the SET NULL or SET DEFAULT statements in the referential constraint's definition for delete.
 - Replace the column default definition in the ALTER TABLE statements with the column default clause in the table definition.
- ▶ In the ALTER TABLE statements, change all types of WITH CHECK ADD clauses with the MODIFY clause that is supported by Informix.
- ▶ Change the order for the CREATE SCHEMA statements. Make sure that you access objects only after you have already created them.
- ▶ Apply Informix-specific SQL statement options. Based on the target database server storage layout, specify the fragmentation and storage clause for tables and indexes.
- ▶ Review the SELECT clause in the view definitions for the built-in functions and join types, for example, you need to replace CROSS JOINS.
- ▶ Review the database-level grants:
 - Combine the grant and revoke permission with the RESOURCE privilege for the database-level statements, such as CREATE SCHEMA, CREATE TYPE, and CREATE TABLE.

- Remove the grant and revoke permissions for the BACKUP and RESTORE statements.
- Remove the grant and revoke permissions for service broker and full text search.
- Replace the role hierarchy definitions and assignment using `sp_addrolemember` with the SQL statement `GRANT <role>|<user> to <role>`.

5.3 Data movement

After you create all the database objects on the target Informix database server, you can then move data. You can either unload the data from the source database server and load the data to the target server, or you can set up a distributed environment to move the data directly from the source server to the target server.

Plan the data movement well to ensure a smooth and successful data migration:

- ▶ Consider a direct access between the source and target systems for smaller tables to avoid the unload and load. Use the Informix Enterprise Gateway for the distributed access.
- ▶ Select the data unload method:
 - Use the SQL Server bcp utility or write an sqlcmd-based SQL script to benefit from batch processing.
 - Use the SQL Server wizard for table-based unloads.
- ▶ Define the data unload strategy. Answer the following questions:
 - Can you unload all the data in one step? Do you have enough disk space?
 - How much time will the unload take? Can you run the unload serially or do you need to run several unloads in parallel?
 - Do your unloads affect production?
 - Can you unload a snapshot one time or do you need to set up a delta copy?
- ▶ If you use pipes for the unload and load, establish the copy process of the data from one system to the other using rcp. Consider data compression to minimize network traffic.
- ▶ Consider possible changes that are required in data representation during the unload file so that Informix can identify the content correctly, which might affect the unload utility selection.
- ▶ Choose the load utility on the Informix side.

To have a close estimate of the time that will be required for the data movement, you can start with a small subset of tables or a small amount of data. Test the unload jobs, and load the test data into the new target database. Check the data consistency for your loaded data. Make sure that you have a non-logging target if you use bulk load. If you can set up a non-logging target, load the data with a utility that switches the logging off automatically.

Take time measurements during the data movement test. From the results of those load tests, you can determine a much better estimate of the time that will be required for the load and the required disk space.

5.3.1 Unloading tables with the Import/Export wizard

The SQL Server Management Studio Import/Export wizard provides an easy-to-access interface to unload data by table. This wizard provides you with an interface where you can specify all necessary options, the target unload file specifications, the table or a query from which the data comes, and the layout of the data file.

Perform these tasks to unload the data using the Import/Export wizard:

1. From the Management Studio ObjectExplore window, select the database. Then, right-click the database to open the context menu.
2. Select **Task menu** → **Export Data**.
3. Go through all the option windows, and specify the necessary parameters for the unload operation.

Select either **Fixed width** for generating rows with fixed column width or **Delimited** for generating trimmed rows with a defined column separator. For better disk space utilization, delimited data form is the best choice.

If you select the delimited data format, make sure that you specify a vertical bar (|) in addition to the default setting of {CR}{LF} in the definition of the Row Delimiter.

Your decisions for the output file structure are critical, because they affect the output file size and the load job performance later.

If your source table contains date or datetime columns, review the format of the unload data. Refer to 5.3.4, “Changing the data representation” on page 233 for the necessary changes in the generated output file. In case you need to change the data format, do not export data by the table, specify a query, and apply the required format transformation in the SQL query. Most of the time, you can accomplish this transformation with a cast in a select list to the existing column. Give the cast expression an explicit name to execute the unload successfully.

Unfortunately, you have to specify and run the export operation for each table separately using the Export wizard.

5.3.2 Generating an SQL Server Integration Services package to perform the unload task

The SQL Server provides another method to unload data. You can use the SQL Server Integration Services package to perform this task. To unload the data using SQL Server Integration Services, you must define a package first. The package defines all the necessary details for the unload, such as the source and target definition, the data transformation rules, and the data flow description. You can start SQL Server Integration Services either on a command line or from an integrated console.

To create an SQL Server Integration Services package, perform the following steps:

1. Start the Business Intelligence Development Studio.
2. Choose the option for creating a new integration services project.
3. Create the flow components in the package:
 - Create the following components for a simple unload without transformation:
 - One Object Linking and Embedding (OLE) DB source component
 - One target flat file destination component
 - Create the following components for a simple unload with transformation:
 - One OLE DB source component
 - One data transformation component
 - One target flat file destination component
4. Initialize the OLE DB source. Add connection parameters for user, server name, source database, and source table.
5. Initialize the target flat file destination component:
 - Add the target file parameter, such as file name and location.
 - Specify the data layout, such as delimited or fixed format.
 - Specify the delimiter.
 - Define the mapping.
6. Initialize (if needed) the data transformation component. Add the data type mapping.
7. Save the package.
8. Debug the package execution in the GUI.

9. Execute the package either from the Business Intelligence Development Studio or from the command line by running the following command:

```
dtexec /f "<your_package_path>\package.dtsx"
```

5.3.3 Unloading data with SQL Server command-line tools

When you set the data unload strategy, make sure that you have enough disk space to keep all the data files until the data is loaded successfully and verified on the target database. An alternative is to use pipes to save disk space. Be aware that the unload step and the load step are tied together closely when using pipes. Start the write into the pipe first. If the source database server and the target database server are separate systems, set up a process to copy the created unload files to the target system.

The implementation of the unload consists of two essential tasks:

- ▶ Unload the data.
- ▶ Transform the unload data into the final data representation.

Data transformation enables the target Informix database server to identify correctly the content during the load. The ideal situation is to combine these two tasks into one step. Separating these two tasks into two independent steps means that you have to unload the data to the disk first and, then, apply all the required changes with a customized program or script to generate the final output file. This process can double the disk space requirements for each table object.

There are two SQL Server command-line-based facilities for unloading the data:

- ▶ Using the bcp utility to generate a delimited output file based on tables or select statements that are specified as the command-line parameters
- ▶ Using an SQL script within the sqlcmd utility and redirecting the output to the unload file

We discuss in detail how to use both utilities to create the unload files in the following sections. We focus on how you can create files with separate row formats, such as delimited and fixed. In addition, we show how you can change the format of the data easily.

Unloading data using the bcp utility

The bcp utility is a command-line utility. The command-line utilities provide advantages that the SQL Server Management studio does not have. You can create batches for scripting table unloads and include the batches in the Windows scheduling to start the batches at certain times.

The bcp utility can load and unload data. The unload supports the creation of delimited-column and fixed-position-column files. Both data file layouts are supported by most of the Informix load utilities. You can define your own delimiter and row end marker. In addition, you can create a format file to define the output layout one time to use multiple times. Using the bcp utility, the data source can be a table or the result set of a query.

In the following examples, we describe several common use cases for creating data unload output files.

Example 5-24 shows the data unload from a table in the delimited format. You can create the format file manually. You also can run the bcp utility without the -f and -t options and specify the appropriate values at the command line to create the format file.

Example 5-24 Use bcp with a format file to create a delimited unload file

```
# Unload the data from the authors table based on a format file
C:\project\output>bcp "bookdb.dbo.authors" OUT "c:\project\output\authors.unl"
-T -t "|" -f "c:\project\output\authors.fmt"
```

#Content of Format file c:\project\output\authors.fmt

```
10.0
9
1      SQLCHAR 0 11      " | " 1      author_id Latin1_General_CI_AS
2      SQLCHAR 0 40      " | " 2      name Latin1_General_CI_AS
3      SQLCHAR 0 20      " | " 3      firstname Latin1_General_CI_AS
4      SQLCHAR 0 12      " | " 4      phone Latin1_General_CI_AS
5      SQLCHAR 0 40      " | " 5      address Latin1_General_CI_AS
6      SQLCHAR 0 20      " | " 6      city Latin1_General_CI_AS
7      SQLCHAR 0 2      " | " 7      state Latin1_General_CI_AS
8      SQLCHAR 0 5      " | " 8      zip Latin1_General_CI_AS
9      SQLCHAR 0 1      " | \r\n" 9      contract      ""
```

#Generated outputfile

```
001-23-4567|Smith|Santo|(588) 001-40|16 Ucptv Pkqx|Concord|Ne||0|
022-33-6666|Kirstein|Vincenzo|(102) 884-69|3411 E|Trenton|Ne|11654|0|
033-44-2212|Katren|Britt|(082) 497-40|94 Uejg Be|Montgomery|Pe|37329|0|
066-33-5555|Almaden|Erick|(809) 140-30|8 UdsV Cej|Columbia|Pe|22940|0|
066-01-0008|Testname|Garret|(724) 446-39|4 Acq|Richmond|Ka|02505|0|
088-11-3222|Zhang Smith|Kurt|(619) 421-97||Austin|Mi|53681|0|
088-44-3333|Columbus|Jessie|(089) 776-17|440 Im|Augusta|Co|06445|0|
```

Next, we show an example that creates a fixed-format output file without column delimiters using the same environment. Example 5-25 on page 231 shows the bcp command and the options for the unload.

Example 5-25 Use bcp with a format file to create a fixed format unload file

```
#execute the bcp
C:\temp\ouput>bcp "bookdb.dbo.authors" OUT "c:\temp\ouput\unl" -T -f fixed.fmt
```

#Description of the format file

10.0

9

1	SQLCHAR	0	11	""	1	author_id	Latin1_General_CI_AS
2	SQLCHAR	0	40	""	2	name	Latin1_General_CI_AS
3	SQLCHAR	0	20	""	3	firstname	Latin1_General_CI_AS
4	SQLCHAR	0	12	""	4	phone	Latin1_General_CI_AS
5	SQLCHAR	0	40	""	5	address	Latin1_General_CI_AS
6	SQLCHAR	0	20	""	6	city	Latin1_General_CI_AS
7	SQLCHAR	0	2	""	7	state	Latin1_General_CI_AS
8	SQLCHAR	0	5	""	8	zip	Latin1_General_CI_AS
9	SQLCHAR	0	1	"\r\n"	9	contract	""

#Output file structure -- only the first columns

001-23-4567Smith	Santo	(588)
022-33-6666Kirstein	Vincenzo	(102)
033-44-2212Doe	Jane	(809)
066-01-0008Testname	Garret	(724)

The methods that are shown in Example 5-24 on page 230 and Example 5-25 unload the data as is without any data transformation. However, there are several data types to which you must apply a change in the format to load the data to an Informix database successfully.

Next, we demonstrate how to use a select statement as the input source into the bcp utility to generate the unload file with the data format transformed.

Example 5-26 shows the unload in combination with a simple data transformation. We change the data type in the bookdb table from datetime to datetime2 to obtain a fraction(7) format. This change is required for the subsequent load. We defined a select with a cast in the bcp for the input source option. We make sure that the cast expression column is assigned a name; otherwise, the data format is incompatible with Informix.

Example 5-26 Using an SQL statement for bcp to generate the unload

We changed the table definition for the table bookdb.dbo.booktab

The datetime columns are modified into a datetime2

```
CREATE TABLE booktab1
```

```
(
```

book_id	t_book_id	NOT NULL,
book_no	CHAR(10)	NOT NULL,
title	VARCHAR(80)	NOT NULL,

```

brand_id      INT          NOT NULL,
price         MONEY        NULL,
notes         VARCHAR(200) NULL,
author_id     t_id         NOT NULL,
release_date  DATETIME2    NOT NULL DEFAULT (getdate()),
change_date   DATETIME2    NULL,
)

#Run the bcp utility using the query
C:\project\output>bcp "SELECT book_id, CAST (release_date AS DATETIME) FROM
bookdb.dbo.booktab" queryout modified_date_time2.out -f "datetime_change.fmt"
-T

#Format file for the bcp unload
#Make sure that all the columns have an assigned name !
10.0
2
1      SQLCHAR 0 6      "|" 1      book_id Latin1_General_CI_AS
2      SQLCHAR 0 24     "|\r\n" 2      changed      ""

C:\temp\ouput>type modified_date_time2.out
051614|2001-04-07 20:25:00.000|
067418|2004-08-14 20:26:00.000|
110542|2002-08-25 09:03:00.000|

```

Unloading data using T-SQL scripts

The `sqlcmd` utility is an SQL Server command-line utility that interprets SQL and T-SQL statements in batches and generates the output. The benefits of using the `sqlcmd` utility are similar to using the `bcp` utility. You can embed the `sqlcmd` utility into command batch scripts and into the Windows Scheduler. The `sqlcmd` utility is more flexible than the `bcp` utility in using the T-SQL batches to generate the output file. You can benefit by generating T-SQL stage tables or temporary aggregating tables or by transforming the source data before generating the output.

Example 5-27 on page 233 shows the general method of using the `sqlcmd` utility to unload data. Consider the following environment changes when using the `sqlcmd` utility:

- ▶ Switch off the row count with the `set nocount on` statement.
- ▶ Apply the `-h -1` option to the `sqlcmd` call. This option suppresses the column name header and the line separator, giving the select output pure ASCII-based data.

Another general rule of using the `sqlcmd` utility to unload data is to avoid generating statement-related messages, because you want only the real data in

the output. For example, do not use the USE database statements in your script. Specify the database with the -d option in the command line.

Example 5-27 Using the sqlcmd utility to create a fixed format unload data file

```
#Use a file for the T-SQL or SQL statements
C:\project\sqlcmd>type sq.txt
set nocount on
SELECT book_id ,    CAST (release_date AS DATETIME2(5))
FROM dbo.booktab1

#Execute the sqlcmd utility with the following options
C:\temp\ouput>sqlcmd -d bookdb -i sq.txt -h -1

#Output file
051614 2001-04-07 20:25:00.00000
067418 2004-08-14 20:26:00.00000
110542 2002-08-25 09:03:00.00000
990238 2005-05-12 03:24:00.00000
996699 2005-03-23 20:37:00.00000
```

5.3.4 Changing the data representation

Identifying the data and data types to which you need to apply a conversion is a required step in a migration project, regardless of the data unload utility and strategy that you choose. Informix and SQL Server have many kinds of data type mapping. However, you need to change certain data formats on the unload to enable the target server to load the data successfully. In this section, we provide a data type overview and discuss the format differences.

Integer data types

SQL Server includes the following integer data types:

- ▶ TINYINT
- ▶ SMALLINT
- ▶ INTEGER
- ▶ BIGINT
- ▶ TIMESTAMP
- ▶ ROWVERSION

You might also consider the BIT type as an integer data type. In general, the integer data types have the same format in both database systems and require no conversion during the unload. Note that the TIMESTAMP and ROWVERSION data types are unloaded in hexadecimal value. If the hexadecimal value causes problems during the load, unload the data with a SELECT *columnname*+0 statement from the table expression to change the hexadecimal format to decimal.

Floating point data types

The floating point types are NUMERIC, DECIMAL, FLOAT, REAL, MONEY, and SMALLMONEY. They are compatible and do not require modification.

Date and datetime data types

Date and datetime data types define columns for DATE, DATETIME, DATETIME2, SMALLDATETIME, DATETIMEOFFSET, and TIME. There are several data types to which you need to apply a change in the format. Table 5-5 shows the default representation, the expected format, and how to change the format using simple SQL built-in functions.

Table 5-5 Standard data type format definitions and possible ways to modify the format

	SQL Server	Informix
DATE	<pre>SELECT date FROM table</pre> 2010-03-11 No change in the format required during the unload	<pre>export DBDATE=Y4DM</pre> <pre>SELECT date FROM table</pre> 2010-03-11 Prerequisite for the load: set DBDATE according to the format in the data files
DATETIME	<pre>SELECT release_date FROM booktab</pre> 2001-04-07 20:25:00.000 No change in the format required during the unload	<pre>SELECT release FROM systables</pre> 2010-03-12 00:04:45.000
DATETIME2	The default fraction representation is datetime2(7) <pre>SELECT standard_datetime2 FROM table</pre> 2001-04-07 20:25:00.0000000 Change the representation with <pre>SELECT CAST(standard_datetime2 AS DATETIME2(5)) FROM table</pre> 2001-04-07 20:25:00.00000	Maximum supported fraction scale is 5

	SQL Server	Informix
SMALLDATETIME	SELECT smalldt FROM table 2010-03-12 00:27:00 No change in the format required during the unload	Define the target column as DATETIME YEAR TO SECOND
TIME	Default fraction format: time(7) SELECT standard_time FROM table 20:25:00.0000000 Change representation with: SELECT CAST(standard_time AS TIME(5)) FROM table 20:25:00.00000	Define the target column as DATETIME HOUR TO FRACTION (5)
DATETIMEOFFSET	Default representation. SELECT dt_offset FROM table 2001-04-07 20:25:00.0000000 +00:00 Change the representation with SELECT CAST(dt_offset as DATETIME) FROM table 2001-04-07 20:25:00.00000	Define the target column as datetime year to fraction (5).

Character data types

The format of the character values depends on the local language settings. Make sure that the target system has a similar setting for the language or code page of the language settings that are used on the SQL Server. You can influence the language settings on various levels in Informix:

- ▶ Server (instance)-based: Set the SERVER_LOCALE environment variable
- ▶ Database-based: Set the DB_LOCALE environment variable
- ▶ Client-based: Set the CLIENT_LOCALE environment variable

For more information about the local language support setting, see *IBM Informix GLS User's Guide*, G229-6373, at the IBM Informix V11.50 Information Center:

<http://publib.boulder.ibm.com/infocenter/idshep/v115/index.jsp>

5.3.5 Loading data into the Informix database server

Unloading the data from the SQL Server and loading the data into the target database server are two steps in the data migration. To perform this data unloading and loading in separate steps requires additional disk space. If disk space is a concern, an alternative data movement strategy is to use pipes. You can set up pipes to act as the writer on the source server and as the reader on the target server. Another good way to run the unloads in parallel with the daily workload on the production source system is to split the tables into small sets for unloading and loading.

Depending on the amount of data that you need to move, the layout of the unload file, the unload media used (pipes or disk), and the layout of the target Informix tables, you can choose one of many data load utilities that Informix provides:

- ▶ **High-Performance Loader**

High-Performance Loader is a job-based command-line utility that provides the ability to define fully parallelized load and unload jobs. This utility also supports multiple disk files and pipes as input and output destinations.

- ▶ **External Table**

Introduced in Informix 11.50.FC6, External Table provides a fully parallelized SQL interface to load data to and unload data from multiple disk files and pipes. The advantage of the External Table is that it can be embedded easily into various SQL applications and scripts and can be executed and triggered remotely.

- ▶ **The dbload utility**

The dbload utility is a command-line utility that you can use to manage the load of data in separate formats, such as fixed format or delimited. The data format is defined in a control file.

- ▶ **The LOAD statement**

The Informix dbaccess command-line utility provides a serial load interface to load the single-delimited data into a target table. You can embed the LOAD statement easily in any type of SQL script that is intended to run in dbaccess.

In the following sections, we introduce these tools. For more detailed information about the load and unload utilities and their usage in migration activities, refer to the *IBM Informix High-Performance Loader User's Guide, v11.50*, SC23-9433, and the *IBM Informix Migration Guide*, SC23-7758, which is available at this website:

<http://www-01.ibm.com/support/docview.wss?uid=swg27013894>

The dbimport utility is another utility that you can use for data movement. For our discussion, we do not consider this utility as one of the data movement tools, because it relies on particular file and directory structures for data and schema definitions. If you want a utility to unload or load a complete database, the dbimport utility might be your best choice.

Loading data with the High-Performance Loader utility

The High-Performance Loader, which ships with the Informix database server, is intended for loading and unloading a large quantity of data in parallel. The load and unload operations are based on jobs. You can have separate data sources (such as pipes, tapes, or files) and file or pipe arrays for the load. The utility contains a graphical user interface (GUI) to create and monitor the jobs. You can also maintain the job definition and execution using a command-line executable. The High-Performance Loader can load delimited, binary, ASCII, and multibyte data sources. If you need to load huge volumes of data, the High-Performance Loader is a good utility to use.

Loading data into the database with the High-Performance Loader utility requires a number of steps to create and execute the load job. These steps are required regardless of the interface that you use for the setup. You can set up the load job with the ipload GUI utility (available only on UNIX) or with the onpladm utility.

Perform these steps to set up a High-Performance Loader data load job:

1. Create a job under a project.
2. Define the data source as a device, which can be an array, a tape, or a pipe.
3. Define the source file format, such as delimited, ASCII, or binary.
4. Create the SQL query to define the target database and the table.
5. Map the input stream position and fields to the target columns.
6. Define filter constraints in case not all rows will be inserted.

After you create the job, you can run it from either ipload or the GUI. If you used onpladm to define the tasks, you must use onpload to run the jobs.

In High-Performance Loader, you can define both data load and data unload jobs. For loading data, create a load job. Choose the express mode if the data volume is large. This mode takes all the possible settings for performance improvement during the load, such as disabling logging, as well as the indexes

and constraints. After the load, all objects that are disabled during the load by the High-Performance Loader are re-established automatically. After running an express load, the only requirement for the administrator is to take a Level 0 archive. Load all the tables, and then, perform the backup one time to save time.

Next, we look at several brief examples about how to set up and execute a data load job with the help of the onpladm utility. The first two examples are for the Windows environment. The output of the job execution can vary on UNIX platforms. We start with a simple job definition. This job loads a delimited data file into a target database table in express mode, meaning that the database server disables the logging of each insert operation automatically.

Example 5-28 shows the job definition and execution. On Windows, make sure that you specify a log file for the job execution to see the status. On UNIX, the output is generated automatically on /dev/stdout.

Example 5-28 How to set up and execute a job using the High-Performance Loader utility

```
#Define the job
onpladm create job load_authors -d tabledata.unl -D books_on_ids -t authors -fl
#execute the job
C:\temp\HPL>onpload -j load_authors -fl -l output

C:\temp\HPL>type output
Mon Mar 08 23:27:15 2010

SHMBASE          0x0c000000
CLIENTNUM        0x49010000
Session ID 1

Load Database    -> books_on_ids
Load Table       -> authors
Device Array     -> load_authors
Record Mapping   -> load_authors
Convert Reject   -> c:\temp\load_authors.rej
Filter Reject    -> c:\temp\load_authors.flt
Set mode of index PK_author_id to disabled
Reset mode of indexes "dbo".PK_author_id to original enabled mode
Table 'authors' will be read-only until level 0 archive

Database Load Completed -- Processed 40 Records
Records Inserted-> 40
Detected Errors--> 0
Engine Rejected--> 0

Mon Mar 08 23:27:20 2010

#you can additionally run the job with
```

```
C:\temp\HPL>onpladm run job load_authors -f1
```

This definition maps only columns one-to-one from the delimited data file, and we have only one data file. If you want to have an array of multiple data files as the input, you must modify the array definition by writing the definition in a file and creating the new device object in the database server. The definition file is specified using the -F option. After that, create a job using this new object. Make sure that you use the -fla option when specifying the array. Example 5-29 describes in detail all the necessary definitions and commands for creating and executing the job.

Example 5-29 Definition of a device array and the usage in a load job

```
#device definition
c:\temp\HPL>type devicefile
BEGIN OBJECT DEVICEARRAY load_table_array
BEGIN SEQUENCE
TYPE FILE
FILE C:\temp\HPL\tabledata.unl
TAPEBLOCKSIZE 0
TAPEDEVICESTR 0
PIPECOMMAND
END SEQUENCE
BEGIN SEQUENCE
TYPE FILE
FILE C:\temp\HPL\tabledata1.unl
TAPEBLOCKSIZE 0
TAPEDEVICESTR 0
PIPECOMMAND
END SEQUENCE
BEGIN SEQUENCE
TYPE FILE
FILE C:\temp\HPL\tabledata2.unl
TAPEBLOCKSIZE 0
TAPEDEVICESTR 0
PIPECOMMAND
END SEQUENCE
END OBJECT

#Create the Device
C:\temp\HPL>onpladm create object -F devicefile
Successfully created object DEVICEARRAY load_table_array

#Check existence
C:\temp\HPL>onpladm list device
load_table_array

#Create the Job we use -fla specifying the device array
```

```
C:\temp\HPL>onpladm create job load_arr -d load_table_array -D books_on_ids -t authors -fla  
Successfully created Job load_arr
```

```
C:\temp\HPL>onpload -j load_arr -fl -l output
```

Next, we provide an overview of what is required to use a pipe as the input source. The steps for creating the pipes as an input source and the definition of the load jobs are similar. Example 5-30 shows the detailed commands using the High-Performance Loader on Linux.

Example 5-30 Using a pipe as an input for High-Performance Loader

```
#Defintion of the pipes in a device array  
#cat pipedesc  
  
BEGIN OBJECT DEVICEARRAY pipedesc  
BEGIN SEQUENCE  
TYPE PIPE  
FILE  
TAPEBLOCKSIZE 0  
TAPEDEVICESTRIDE 0  
PIPECOMMAND "cat /tmp/tabledata"  
END SEQUENCE  
BEGIN SEQUENCE  
TYPE PIPE  
FILE  
TAPEBLOCKSIZE 0  
TAPEDEVICESTRIDE 0  
PIPECOMMAND "cat /tmp/tabledata1"  
END SEQUENCE  
BEGIN SEQUENCE  
TYPE PIPE  
FILE  
TAPEBLOCKSIZE 0  
TAPEDEVICESTRIDE 0  
PIPECOMMAND "cat /tmp/tabledata2"  
END SEQUENCE  
END OBJECT  
  
$create object -F pipedesc  
Successfully created object DEVICEARRAY loadfrompipe1  
  
$onpladm create job load_from_pipe -d pipedesc -D books_on_ids -t authors -fla  
Successfully created Job load_from_pipe  
  
$onpload -j load_from_pipe -fla
```

Mon Mar 08 22:20:10 2010

SHMBASE 0x0c000000
CLIENTNUM 0x49010000
Session ID 32

Load Database -> books_on_ids
Load Table -> authors
Device Array -> pipedesc
Record Mapping -> load_from_pipe
Convert Reject -> /tmp/load_from_pipe.rej
Filter Reject -> /tmp/load_from_pipe.flt
Set mode of index PK_author_id to disabled
Reset mode of indexes "dbo".PK_author_id to original enabled mode
Table 'authors' will be read-only until level 0 archive

Database Load Completed -- Processed 120 Records
Records Inserted-> 120
Detected Errors--> 0
Engine Rejected--> 0
Mon Mar 08 22:20:15 2010

The High-Performance Loader utility includes a rich set of options for setting up the load jobs. Although we focus our discussion primarily on the load of delimited files, capabilities exist for setting up column offset-based load files, which are also referred to as *fixed format*, as well as other default column mapping and row filtering based on filter values. For more detailed information about High-Performance Loader, refer to the *IBM Informix High-Performance Loader User's Guide, v11.50, SC23-9433*.

External Table

Unlike the High-Performance Loader utility, External Table is based on SQL statements. External Table is introduced in Informix V11.50.FC6. The external tables are not table objects that contain data in the database server. Instead, they act more as descriptions of an interface that defines a data source or a data target outside of the database server. With External Table, you can complete the following tasks:

- ▶ Create an external table that is based on a reference table definition or with your own column definitions.
- ▶ Drop an external table.
- ▶ Define the storage objects that are assigned to the table, such as one or more pipes, one or more files, or a mix of the two.
- ▶ Choose the layout of the data, such as delimited or fixed format in the space object.

- ▶ Apply most of the SELECT options to an external table, such as projection, selection, and table join.
- ▶ Select data from a table or use a table as a source or target in an insert into *target* select from *source* clause.
- ▶ Apply any parallelization features, such as PDQ, to the work with external tables to split the complete workload throughout multiple threads.

The most common use case for external tables is the following insert statement:

```
INSERT INTO <target_table> SELECT * from <external_table>
```

You can use the external tables in any client database application and embed the load into an SQL script. The performance benefit in comparison to the serial client-based utilities, such as the LOAD statement and the dbload utility, is that the database server processes the load itself. Even if you use the external table in the insert statement, there is no interaction between the client and the server that exchange row data. Also, you are allowed to select data from the external table in any client; however, expect traffic between the client and the database server in that case.

Example 5-31 shows examples with various options that you can use in the CREATE EXTERNAL TABLE statement.

Example 5-31 Creating external tables in Informix

```
-- Create the table in the same layout as Authors table
CREATE EXTERNAL TABLE EXT_AUTHORS SAMEAS AUTHORS USING
(DATAFILES("DISK:/migration/authors_1.unl"), FORMAT 'delimited');

-- Long version, not use SAMEAS
CREATE EXTERNAL TABLE ext_authors (
  author_id VARCHAR(11) ,
  name VARCHAR(40) ,
  firstname VARCHAR(20) ,
  phone CHAR(12) NOT NULL,
  address VARCHAR(40),
  city VARCHAR(20),
  state CHAR(2),
  zip CHAR(5),
  contract INTEGER NOT NULL
) USING
(DATAFILES("PIPE:/migration/authors.pipe"), FORMAT 'delimited');

-- Using more than one data file
DROP TABLE ext_authors;
CREATE EXTERNAL TABLE ext_authors SAMEAS authors USING
(datafiles ("disk:/tmp/authors1.unl","disk:/tmp/authors2.unl"));
```

```
--Create table for fixed format data
CREATE EXTERNAL TABLE "dbo".authors
(
    author_id VARCHAR(10) EXTERNAL CHAR(20),
    name VARCHAR(40) EXTERNAL CHAR(40),
    firstname VARCHAR(20) EXTERNAL CHAR(20),
    phone CHAR(12) EXTERNAL CHAR(12)
)
using
(
    DATAFILES ( "disk:/tmp/fixed_format_authors"),
    FORMAT "fixed",
    REJECTFILE "/tmp/reject"
);
```

In Example 5-31 on page 242, we describe only how to create and drop an external table. With the create and drop, you define the location and the structure of your data. Now, you must load data into the database server, which requires that you embed the external table into SQL statements, such as **SELECT** and **INSERT**. Example 5-32 shows several ways that you can use the external tables in Informix.

Example 5-32 Apply INSERT and SELECT statements to external tables

```
-- Simple insert for load and unload your data
INSERT INTO authors SELECT * FROM ext_authors;
INSERT INTO ext_authors SELECT * FROM authors;

-- Simple SELECT: returns the data to any client
SELECT * from ext_authors;

--Projection
INSERT INTO authors(author_id,name,firstname)
    SELECT author_id,name,firstname FROM ext_authors;

--selection
INSERT INTO authors(author_id,name,firstname)
    SELECT author_id,name,firstname FROM ext_authors
    WHERE zip MATCHES "2*";

-- Apply any data transformation
INSERT INTO authors(author_id,name,state)
    SELECT author_id, name || firstname,
CASE
WHEN zip MATCHES '21*' THEN 'StateA'
WHEN zip MATCHES '22*' THEN 'StateB'
WHEN zip MATCHES '23*' THEN 'StateC'
ELSE 'MiddleofNowhere' END
```

```

FROM ext_authors
WHERE zip MATCHES "2*";

--Join external and standard tables
SELECT * FROM ext_authors ext, authors in WHERE
ext.author_id = in.author_id;

```

Loading data with the dbload utility

The dbload data loading utility is an executable that is provided with the Informix distribution. You can use the dbload utility for serial loading of data files in delimited or fixed format. A control file is provided as an input parameter at execution time that defines the layout of the load file, column mapping, and the target database table. To have more effective data loading, consider the following notes when you define the dbload utility input parameters:

- ▶ Define the transaction sizes, and commit after a defined amount of inserted rows.
- ▶ Skip rows in the data files.
- ▶ Lock the table for the load.
- ▶ Create log file specification.
- ▶ Specify fault tolerance.

The key input for the dbload utility is the specification of the control file. There, you determine if you want to load data from a delimited or column-oriented (fixed format) file. In addition, you specify the file name of the load file and define the target table where the rows will be loaded. You can specify multiple load files in one control file, which enables the dbload utility to load more than one table at execution time.

Example 5-33 show two sample control files for handling delimited rows.

Example 5-33 Sample dbload control files

```

#Control file with loadfile with delimited rows
FILE authors.unl DELIMITER '|' 9;
INSERT INTO authors;

#Loading multiple Files with one dbload start
FILE brands.unl DELIMITER '|' 3;
INSERT INTO brands;
FILE booktab.unl DELIMITER '|' 9;
INSERT INTO booktab;
FILE book_history.unl DELIMITER '|' 10;
INSERT INTO book_history;

```

The control file for column space-oriented rows differs slightly. Here, you specify the name of the column and the offset for each column that the row contains. The space-oriented files or fixed format rows cannot contain NULL values. You must specify value representation for the NULL columns. The dbload utility replaces the value with the NULL value during the load of the data to the table. Example 5-34 shows samples of how to specify these values.

Example 5-34 How to define column offsets in a control file

```
#Control File with column size oriented rows.
#Data selected form SQL Server:
job_id job_desc min_level max_level
-----
1 Secretary 44 139
2 Developer 84 120
3 Secretary 88 134
4 Teamleader 45 149
5 Software Architect 39 186
6 Manager 82 165
7 Consultant 89 112
8 Software Architect 69 151
9 Developer 26 177
10 Guru 86 175
11 XXXX 87 192

#
#Define the start of the column in the row
#Keep in mind that you have to add the byte for the space to the
#first two column sizes
$cat controlfile
FILE jobs.unl
(
job_id 1-7,
job_desc 8-58 NULL = 'XXXX',
min_level 59-68,
max_level 69-78
);

INSERT INTO dbo.jobs VALUES (job_id, job_desc, min_level, max_level);
```

Because most of the data types have a fixed length, or in the case of a varchar the select output is set to the maximum size, you can define easily the control file for most of your data exports. The control file specification also supports the user-defined data type.

Example 5-35 shows how the definitions in the control file are structured for a user-defined data type `adr` in the delimited and column size-based input files.

Example 5-35 Handling user-defined row types

```
#Table definition statement in Informix
CREATE ROW TYPE address ( address CHAR(100),city CHAR(100),zip INTEGER);
CREATE TABLE authors (author_id INTEGER, name CHAR(100), firstname CHAR (100),
adr address);

#Load file delimited
001-23-4567|Smith|Santo|ROW('16 Gryrzau','Concord',37329)|

#Load File column oriented
001-23-4567 Smith      Santo      ROW('16 Gryrzau','Concord',37329)

#Controlfile for a delimited load file
FILE rowtype.txt
DELIMITER '|' 4;
INSERT INTO authors;

#Controlfile for column sized load file
FILE rowtype.txt
(
author_id 1-12,
name 13-26,
firstname 27-37,
address 38-70);

INSERT INTO dbo.authors VALUES (author_id, name,firstname,address);
```

Setting up the control file according to the load file is the prerequisite for the execution of the `dbload` utility. When complete, you can start the `dbload` utility with the following command:

```
dbload -d <database_name> -c <controlfilename> -l <logfilename>
```

The SQL LOAD statement provided by the dbaccess utility

The SQL LOAD statement is an Informix extension to the SQL standard and is available only in the `dbaccess` utility. You can either use the `dbaccess` utility and run the LOAD statement step-by-step from the menu, or you can use the LOAD statement in an SQL batch file. For the batch, save all your SQL statements in a file and redirect the file to `dbaccess` at execution time.

The LOAD statement expects a single delimited file on disk but also can read delimited rows from a named pipe. The default delimiter is a vertical bar sign (`|`). You can change the delimiter with the `DBDELIMITER` environment variable. You must set this variable before you start the `dbaccess` utility. The load itself is

executed serially. All data is read from the file by the client and sent to the database server to insert into the table.

Example 5-36 shows the expected row layout in the data file and how to load the data with the LOAD statement in the dbaccess utility.

Example 5-36 Using LOAD to load data into Informix

```
#UNLOAD file
$type c:\temp\authors.dat
066-01-0008|Testname|Garret|(724) 446-39||Richmond|Ka|02505|False|
088-44-3333|Columbus|Jessie|(089) 776-17|440|Augusta|Co|06445|False|
088-11-3222|Zhang Smith|Kurt|(619) 421-97||Austin|Mi|53681|False|
125-32-9487|Smith2|Jospeh|(961)921-20|533 Ju Ohfo|Helena|No|39036|False|
125-74-8368|Smith3|Jospeh|(964)362-86|707 Itsmr Iq|Columbia||24586|True|
587-99-4215|Smith4|Britt|(701) 209-41||Tallahassee|Il|46123|True|

#Execution from in an command line call
$echo "LOAD FROM c:\temp\authors.dat INSERT INTO 'dbo'.authors;"| dbaccess -e
books_on_ids

#Output of the execution
Database selected.
load from 'c:\temp\authors.dat' INSERT INTO authors;
6 row(s) loaded.
Database closed.
```

Special considerations for loading BLOB and SBLOB data

Loading BLOB or smart BLOB (SBLOB) data requires special considerations. Not all the data loading utilities support the direct load of BLOB or SBLOB data types. In this section, we provide guidance about how to develop a solution to load the BLOB and SBLOB data.

For loading BLOB data, refer to the blobload.ec file in the `$INFORMIXDIR/demo/esqlc` directory, which is distributed with the Informix Client Software Development Kit (SDK). This file contains the demonstration source code for inserting BLOB that is located in a file to a database table. Basically, the BLOB is handled in ESQL/C by a specific `loc_t` structure. This structure contains the information about the BLOB data, such as size and file location. After initialization, you can use the variable as a host variable similar to other built-in data type-based variables.

Example 5-37 shows sample code to load the BLOB data. We want to insert only the BLOB data that is stored in a file. In our ESQL/C file, we prepare an insert SQL statement for loading data. We use the *images* variable to provide details about the location and file size.

Example 5-37 How to insert BLOB data into a table in ESQL/C

```
#Informix Database schema
CREATE TABLE blob_tab ( the_blob BYTE );

#Sample fragments of the esql/c program
#include sqlca;
#include <stdio.h>
main( int argc, char **argv)
{
    $loc_t images;
    $char the_statement[200]
    $char blobfile[200];

    $database blob_database;
    sprintf(blobfile,"%s",argv[1]);
    strcpy(the_statement,"INSERT INTO blob_tab VALUES ( ? )");
    $prepare sql_statement from :the_statement
    images.loc_loctype = LOCFNAME;      /* blob is named file */
    images.loc_fname = blobfile;        /* here is its name */
    images.loc_oflags = LOC_RDONLY;     /* contents are to be read in Informix*/
    images.loc_size = -1;                /* read to end of file */
    images.loc_indicator = 0;           /* not a null blob */
}

$EXECUTE sql_statement USING :images;
if ( sqlca.sqlcode != 0L)
    printf("SQL error %ld occured during INSERT\n",sqlca.sqlcode);
$CLOSE DATABASE
}
```

Similar to the standard BLOB data type, the SBLOB data type requires special consideration in Informix. Informix Client SDK, which ships with a complete Informix distribution, provides a library function set that you can use to load SBLOB data. Using the SBLOB data types does not require the existence of the SBLOB spaces in the Informix database server; however, these spaces are dbspaces for maintaining the SBLOB data types.

As an exercise, we developed a small sample ESQL/C program to load SBLOB data. Example 5-38 on page 249 shows the program code, the base table definition, and a sample *onspaces* call to create a dbspace. The file name containing the SBLOB is provided at execution time. The database is opened first. The SBLOB descriptor needed for communication with the database server

is initialized in the next step. After that, the description of the SBLOB column is requested from Informix. A file descriptor is created, and the data is sent to the database server before the insert occurs. The SBLOB descriptor is used as the host variable for the insert to make sure that the appropriate SBLOB data is assigned to the column.

Example 5-38 Inserting data into SBLOB data types in Informix

```
##Create a sample SBSPACE -- more options for logging are available
onspaces -c -S sblobspace -g 4 -p <path of the OS file> -o 0 -s 200000

##Create a table with an SBLOB
CREATE TABLE "informix".sblob_tab
(
    catalog_num SERIAL8 NOT NULL ,
    advert_descr "informix".clob
) PUT advert_descr IN ( sblobspace)
( EXTENT SIZE 20, KEEP ACCESS TIME)
EXTENT SIZE 16 NEXT SIZE 16 LOCK MODE PAGE;

##The sample esql/c program for a load of an SBLOB
#include sqlca;
#include <stdio.h>

main(int argc, char **argv)
{
    EXEC SQL BEGIN DECLARE SECTION;
    int8 estbytes;
    int error, numbytes, lofd, ic_num, buflen = 256;
    char buffer[150000];
    ifx_lo_create_spec_t *create_spec;
    fixed binary 'clob' ifx_lo_t descr;
    EXEC SQL END DECLARE SECTION;

    FILE *fp ;

    if ( argc < 2 ) exit(2);

    fp=fopen(argv[1],"r");
    if ( fp ) {
        estbytes=fread(buffer,100000,1,fp);
        fclose (fp);
    }
    else exit(1);

    $DATABASE sblob_db;

    /* create the sblob descriptor */
    if ( ifx_lo_def_create_spec(&create_spec) < 0 )
```

```

printf(" Error ifx_lo_def_create_spec %d \n",error);

/* get the column description from Informix */
if (ifx_lo_col_info
    ("sblob_db@on10fc4tcp:sblob_tab.advert_descr", create_spec) < 0 )
    printf(" Error ifx_lo_def_create_spec %d \n",error);

/* create the SLOB structure and initialize with server data */
if ((lofd=ifx_lo_create
    (create_spec,LO_RDWR|LO_NOBUFFER,&descr, &error)) == -1)
    printf(" Error ifx_lo_create %d\n",error);

/* send the BLOB to the server */
numbytes=ifx_lo_write(lofd, buffer, estbytes, &error);

if (numbytes<size )
    printf(" Error ifx_lo_write  %d\n",error);

/* Do the insert we use a serial , first column value is 0 */
EXEC SQL INSERT INTO catalog VALUES (0, :descr);
ifx_lo_close(lofd);

$DISCONNECT ALL;
}

```

Performance considerations for loading the data

You can influence the execution time that is required for the data load into Informix significantly by setting an appropriate environment. We discussed most of the topics on which you need to focus in the previous sections. In this section, we summarize these performance points.

When planning and executing the data load process, remember the following considerations:

- ▶ Run multiple loads in parallel instead of loading the data serially.
- ▶ Prefer the load utilities (High-Performance Loader or external tables) that parallelize the work for each table automatically and can take advantage of database server parallelization facilities, such as PDQ.
- ▶ Use fragmentation on the target tables to allow parallel loading by spreading the work across multiple threads using PDQ. The client applications can use PDQ to benefit from the multithreaded reads and writes from the fragmented table. The fragmentation elimination can improve the query performance.
- ▶ Disable logging during the load manually or by using the High-Performance Loader utility data loading in express mode. The logging is disabled automatically for each accessed table.

- ▶ Create the indexes after loading the data, or disable the index during the data load to save significant overhead time. Compare the time creating the index tree in parallel at the end of the load in one sort, instead of inserting each entry step-by-step, which causes a lot of split and rebalance operations in the index during the load phase.
- ▶ Certain constraints are maintained internally by indexes. You need to disable these constraints during the load. The inconsistency will be detected and fixed when the constraints are enabled again.
- ▶ After loading all your data, re-enable the indexes and constraints by applying an UPDATE STATISTICS statement to all the new databases. This statement brings the table statistics to the current status and enables the database optimizer to determine the most efficient query plan for accessing data.

5.3.6 Data movement with distributed access

In the previous sections, we emphasized data movement between both database servers by unloading and loading the data. Another possible strategy to transfer the data is with *distributed access*. The benefit of this particular strategy is that the data move occurs in one step. Thus, no additional disk space is required for the unload.

If you use distributed access, you need to set up an environment that is based mostly on ODBC data sources, which includes a coordinating instance that maintains data access and flow on both sides. In the following sections, we introduce IBM products that provide distributed access capability.

IBM Informix Enterprise Gateway Manager

Informix Enterprise Gateway Manager, which is available on UNIX and Windows, enables distributed access between an Informix database server and other database servers, such as SQL Server. The distributed access is based on ODBC definitions that are set up on the same systems where the Informix Enterprise Gateway Manager is installed. The Enterprise Gateway Manager daemon listens on a specified port, translates the incoming SQL requests into the ODBC operations, and sends the requests to the target database server.

You can install the Informix Enterprise Gateway Manager on the client system, the database server system, or any other gateway system. After the product is installed and configured, the Informix database server can access the data with SQL statements, such as INSERT, from the remote side. The SQL definitions for accessing tables on a remote, non-Informix server follow similar rules as the distributed queries for accessing a remote Informix database server.

To set up a distributed environment between Informix and SQL Server using the Informix Enterprise Gateway Manager requires the following high-level steps:

1. Install Informix Enterprise Gateway Manager.

The installation requires 80 MB of disk space.

You need to extract the distribution media and then run the executable. The executable that you run depends on your environment:

- Run **setup.exe** on Windows.
- Run **./install_{egm}** on UNIX.

The installation process installs all necessary files:

- On Windows, the installation sets up the configuration settings automatically in the registry for the gateway service and starts the service.
- On UNIX, you have to manually apply the appropriate entries for the gateway daemon to the server `sqlhosts` file and start the `egmd` daemon.

2. Configure Informix Enterprise Gateway Manager:

a. Set up the ODBC definition for the remote database server (SQL Server):

- On UNIX, modify the distributed `ODBC.ini` file in the `<your_egm_install>/egm/odbc` directory, and set the `ODBCINI` environment variable with the file location.
- On Windows, use the ODBC Manager to set up a system data source name (DSN) for the remote access.

b. Make sure that you can access the configured data source on the ODBC side. Use the `demoodbc` demonstration program that ships with the product on all platforms to access the remote non-Informix database server. You can find the `demoodbc` program in the `egm/odbc/demo` directory of your installation directory for Enterprise Gateway Manager.

c. Add the user mapping for the gateway user. The gateway daemon can run under a user other than your configured SQL Server system data source.

- i. Edit the option file to specify the user at the gateway and at the data source, the data source name, and the target password.
- ii. Use the `egmdba -ua <file>` command line to add new user configurations.

3. Check the configuration of the distributed environment on the gateway by running the following command:

```
egmdba -ts DataSourcename gateway_daemon_name user table
```


After you complete these steps successfully, you can access the remote database server from any of your applications or from the dbaccess utility. You can access the data from the database server by specifying the table with the following syntax:

```
<DSN>@<gateway_daemon_name>:owner.table
```

Example 5-39 shows a data transfer from SQL Server to Informix.

Example 5-39 Data transfer with Informix Enterprise Gateway Manager

```
dbaccess << EOF
database books_on_ids
INSERT INTO authors SELECT * from SQLSERVER@gateway:dbo.authors;
INSERT INTO booktab SELECT * from SQLSERVER@gateway:dbo.booktab;
EOF
```

IBM InfoSphere Information Server

There are additional IBM solutions outside of the Informix product family that provide distributed access environments based on ODBC. IBM InfoSphere™ Information Server is one of these solutions. IBM InfoSphere DataStage® is a full-function information movement and data integration tool.

InfoSphere DataStage is intended to be used as an Extraction, Translation, and Loading (ETL) utility that fulfills exactly the requirements of data movement. You can access both databases with *stages* that are provided by the product.

InfoSphere DataStage requires that the database schema is already migrated successfully from one database to another database. In addition to extracting the data from the source database server, you can also define data sources of various types, such as flat files with database data. You can define several translation jobs that can be used to translate data representation automatically. Job scheduling is another useful feature that allows you to schedule and run jobs at specific times. You can run multiple jobs in parallel and specify *job chains* so that the successor can run only if the current job completes successfully.

The job scheduling concept enables you to execute the load in off-peak times in order to get exclusive access to the data or, at least, to have more system resources for the data scan. Scheduling multiple jobs can be useful to achieve a better utilization of the source database server and a significant reduction of the unload time.

For more information, visit the IBM InfoSphere Information Server product site:

<http://www.ibm.com/software/data/infosphere/info-server/overview/>

Additionally, refer to *IBM InfoSphere DataStage Data Flow and Job Design*, SG24-7576, which is available at:

<http://www.redbooks.ibm.com/abstracts/sg247576.html>

IBM Optim Integrated Data Management

The IBM Optim™ Integrated Data Management solution is another solution that you can use for data migration. Unlike the Informix Enterprise Gateway Manager and InfoSphere DataStage, the Optim solution has no real-time distributed environment with a parallel access to the source database server and target database server. Optim can access the source data server, which in our migration environment is SQL Server, with an ODBC connection. The data on the source server is saved to the extract files. You can apply data transformation during the unload process. You can then verify and load the data into the target Informix database.

Figure 5-2 on page 255 depicts a typical Optim configuration for accessing multiple heterogeneous database servers.

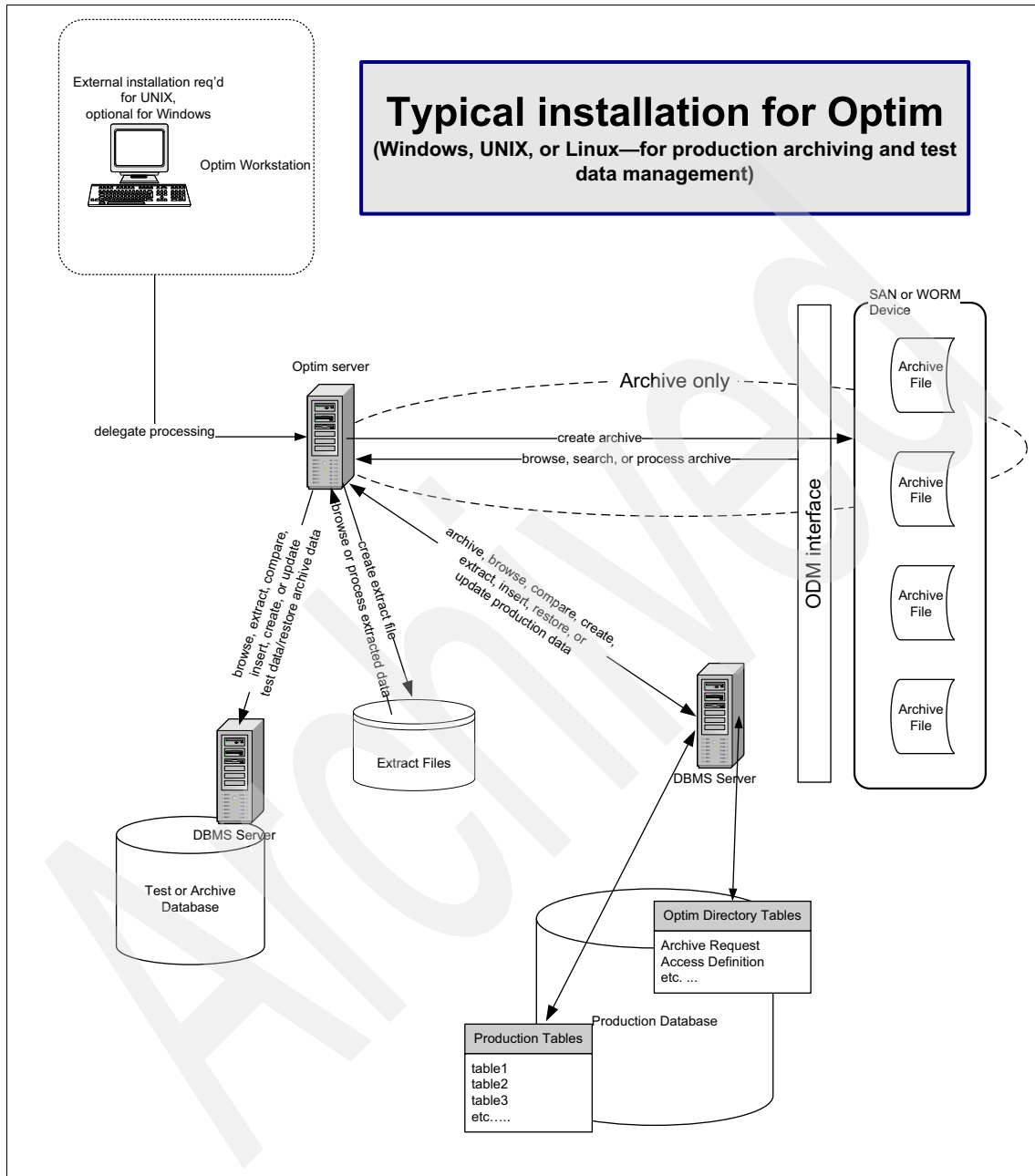


Figure 5-2 Access distributed databases with Optim

For more detailed information about Optim Integrated Data Management, refer to this website:

<http://www-01.ibm.com/software/data/data-management/optim-solutions/>

Application conversion

With the successful completion of the data movement to the target Informix database server, you have set up the base for the next step. Now, you must enable access to the data for your applications.

In this chapter, we give you an overview of the commonly used programming interfaces that are supported by IBM Informix. We discuss the requirements for a database client application migration and provide guidance for modifying the SQL Server applications to run with Informix.

Depending on the application programming interface that is used, the requirements differ for a database client application migration. The major focus in this chapter is a detailed discussion about available database interfaces and the differences between the access techniques in the source database server and the target database server. We describe the conversion of applications written with C/C++, Visual Basic, Open Database Connectivity (ODBC), Visual C++, .NET, Active Server Page (ASP), Java, and PHP (Hypertext Preprocessor).

6.1 Heterogeneous application environments

Most of the modern software infrastructures contain a mix of various applications and servers. There are business driven applications that are especially developed by the company's own development teams. There are also a wide variety of products and solutions available that have been provided by independent software companies. They provide a complete solution for unified company processes, such as HR operations, order and sales processing, and accounting systems. In cases where internal processes that cannot be unified are defined, most of the software providers also define customizing facilities for the software.

In the middle of the complete application architecture are the servers. Commonly, web servers, application servers, and database servers are in use. Our special focus is directed toward the database server. We are interested in the available ways to exchange data between an application and the database server or across the server. How data is exchanged is important when considering porting applications across database providers.

If you take an inventory of the types of applications that you run in your company's database client environment, you likely define two types of applications:

- ▶ Applications where you own the source (in-house developed) and changes can be internally applied
- ▶ Applications that are defined on standard interfaces where you have to rely on the database server certification by the software provider

Both types of applications can use the same unified database interfaces, but the strategy for how to migrate the particular type of application differs. We take a closer look at both types of applications in the following sections.

6.2 Informix 11 support for native client development APIs

The majority of IBM Informix development application programming interfaces (APIs) have been grouped together into the IBM Informix Client Software Development Kit (CSDK) with an associated runtime deployment component called IBM Informix Connect.

In addition, there are quite a few other stand-alone APIs and tools that Informix provides. In this section, we introduce these development APIs, their specifics,

and their supported functionality. We provide more details later in this chapter when we discuss the specific porting issues for each supported database interface.

6.2.1 Embedded ESQL/C

ESQL/C allows the easy integration of structured query language (SQL) statements with C programming language applications. The SQL statement handling is a combination of an ESQL/C language preprocessor, which takes the ESQL/C statements and converts them into ESQL/C library function calls, in combination with an ESQL/C runtime library.

This approach can be helpful when you deal with many SQL-related activities in a C-based application, and you need to focus on the SQL programming more than on complex call-level interfaces to achieve the same goal. Even though ESQL/C provides a tight integration with C applications, it still allows you to focus on the actual SQL problem solution.

IBM Informix ESQL/C supports the American National Standards Institute (ANSI) standard for an embedded SQL for C, which also makes ESQL/C a good technology foundation for any database application migrations to Informix.

6.2.2 Embedded ESQL/COBOL

IBM Informix ESQL/COBOL is an SQL application programming interface (SQL API) that lets you embed SQL statements directly into COBOL code. It consists of a code preprocessor, data type definitions, and COBOL routines that you can call. It can use both static and dynamic SQL statements. When you use static SQL statements, the program knows all the components at compile time.

ESQL/COBOL is currently only available on AIX, HP/UX, Linux, and Solaris.

6.2.3 IBM Informix JDBC 3.50 Driver

Java database connectivity (JDBC) is the Java specification of a standard API that allows Java programs to access database management systems (DBMSs). The JDBC API consists of a set of interfaces and classes written in the Java programming language. Using these standard interfaces and classes, programmers can write applications that connect to databases, send queries that are written in structured query language (SQL), and process the results.

The JDBC API defines the Java interfaces and classes that programmers use to connect to databases and send queries. A JDBC driver implements these interfaces and classes for a particular DBMS vendor.

There are four types of JDBC drivers:

- ▶ Type 1: JDBC-ODBC bridge, plus ODBC driver
- ▶ Type 2: Native API, partly Java driver
- ▶ Type 3: JDBC-Net, pure Java driver
- ▶ Type 4: Native protocol, pure Java driver

For more information about this topic, see the *IBM Informix JDBC Driver Programmer's Guide*, SC23-9421.

IBM Informix JDBC 3.50 Driver is a native protocol, pure Java driver (type 4). When you use IBM Informix JDBC Driver in a Java program that uses the JDBC API to connect to an Informix database, your session connects directly to the database or database server without a middle tier, and it is typically used on any platform providing a standard Java virtual machine (JVM).

The current IBM Informix JDBC Driver is based on the JDBC 3.0 standard, provides enhanced support for distributed transactions, and is optimized to work with IBM WebSphere® Application Server. It promotes accessibility to IBM Informix database servers from Java client applications, provides openness through XML support (JAXP), fosters scalability through its connection pool management feature, and supports extensibility with a user-defined data type (UDT) routine manager that simplifies the creation and use of UDTs in Informix 11. This JDBC 3.5 Driver also includes Embedded SQL/J, which supports embedded SQL in Java.

6.2.4 IBM Informix .NET Provider

.NET is an environment that allows you to build and run managed applications. A *managed application* is an application in which memory allocation and deallocation are handled by the runtime environment. Another good example for a managed environment is a JVM.

The .NET key components are Common Language Runtime and .NET Framework Class Library, such as ADO.NET and ASP.NET.

ADO.NET is a set of classes that provides access to data sources and has been designed to support disconnected data architectures. A *DataSet* is the major component in that architecture, and it is an in-memory cache of the data that is retrieved from the data source. ADO.NET differs from ODBC and OLE DB, and each provider exposes its own classes that inherit from a common interface (for example, *IfxConnection*, *OleDbConnection*, and *OdbcConnection*).

The Informix .NET Provider

The IBM Informix .NET Provider is a .NET assembly that lets .NET applications access and manipulate data in IBM Informix databases. It implements several interfaces in the Microsoft .NET Framework that are used to access data from a database.

Using the IBM Informix .NET Provider is more efficient than accessing an IBM Informix database through either of these two methods:

- ▶ Using the Microsoft .NET Framework Data Provider for ODBC along with the IBM Informix ODBC Driver
- ▶ Using the Microsoft .NET Framework Data Provider for Object Linking and Embedding (OLE) DB along with the IBM Informix OLE DB Provider

The IBM Informix .NET Provider can be used by any application that can be executed by the Microsoft .NET Framework.

The following list details examples of programming languages that create applications that meet this criteria:

- ▶ Visual BASIC .NET
- ▶ Visual C# .NET
- ▶ Visual J# .NET
- ▶ ASP.NET

The IBM Informix .NET Provider runs on all Microsoft Windows platforms that provide full .NET support. You must have the Microsoft .NET Framework SDK, Version 1.1, or later, and the IBM Informix Client SDK, Version 2.90, or later, installed.

6.2.5 IBM Informix ODBC 3.0 Driver

The IBM Informix Open Database Connectivity (ODBC) Driver is based on the Microsoft ODBC 3.0 standard, which is, in turn, based on Call Level Interface specifications developed by X/Open and International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC). The ODBC standard has been around for a long time and is still widely used in database-oriented applications. The current IBM Informix ODBC Driver is available for Windows, Linux, and UNIX platforms, and it supports pluggable authentication modules (PAMs) on UNIX and Linux, plus Lightweight Directory Access Protocol (LDAP) authentication on Windows.

IBM Informix ODBC driver-based applications enable you to perform the following types of operations:

- ▶ Connect to and disconnect from data sources.
- ▶ Retrieve information about data sources.
- ▶ Retrieve information about the IBM Informix ODBC Driver.
- ▶ Set and retrieve IBM Informix ODBC Driver options.
- ▶ Prepare and send SQL statements.
- ▶ Retrieve SQL results and process them dynamically.
- ▶ Retrieve information about SQL results and process it dynamically.

6.2.6 IBM Informix OLE DB Provider

Microsoft OLE DB is a specification for a set of data access interfaces designed to enable a variety of data stores to work together seamlessly. OLE DB components are data providers, data consumers, and service components. Data providers own data and make it available to consumers. Each provider's implementation differs, but they all expose their data in a tabular form through virtual tables. Data consumers use the OLE DB interfaces to access the data.

You can use the IBM Informix OLE DB Provider to enable client applications, such as ActiveX Data Object (ADO) applications and web pages, to access data on an Informix server.

Due to the popularity of the Microsoft .NET Framework, Informix developers on the Microsoft platform typically prefer the .NET Database Provider and integrating existing OLE DB-based applications through a Microsoft .NET Provider for OLE DB.

6.2.7 IBM Informix Object Interface for C++

The IBM Informix Object Interface for C++ encapsulates Informix database server features into a class hierarchy.

Operation classes provide access to Informix databases and methods for issuing queries and retrieving results. Operation classes encapsulate database objects, such as connections, cursors, and queries. Operation class methods encapsulate tasks, such as opening and closing connections, checking and handling errors, executing queries, defining and scrolling cursors through result sets, and reading and writing large objects.

Value interfaces are abstract classes that provide specific application interaction behaviors for objects that represent IBM Informix database values (value objects). Extensible value objects let you interact with your data.

Built-in value objects support ANSI SQL and C++ base types and complex types, such as rows and collections. You can create C++ objects that support complex and opaque data types.

6.2.8 Additional APIs for accessing Informix 11

The following sections describes additional APIs for accessing Informix 11.

PHP support

PHP (Hypertext Preprocessor) is a powerful server-side scripting language for web servers. PHP is popular for its ability to process database information and create dynamic web pages. The term *server-side* refers to the fact that PHP language statements, which are included directly in your Hypertext Markup Language (HTML), are processed by the web server.

The following list details the PHP drivers for accessing Informix that are available:

- ▶ **PDO_IBM**
PDO_IBM is a driver that implements the Informix PHP Data Objects (PDO) interface to enable access from PHP to IBM databases.
- ▶ **Unified ODBC**
Unified ODBC uses the standard ODBC interface for database communication and is commonly combined with a generic ODBC driver.
- ▶ **The Informix PHP driver (ifx interface)**
The Informix PHP driver uses a native database connection to Informix that is provided by ESQL/C connection libraries.
- ▶ **Informix PDO**
Informix PHP Data Objects (PDO) is an object-oriented database development interface that is available from PHP 5 and that uses a native database connection to the database server.

You can obtain PHP in multiple ways. Either you download the source, compile the base package and all necessary programming interfaces, and plug it into an pre-existing web server, such as Apache. Or, download a precompiled version, such as Zend Core for IBM, or install the XAMPP package.

We suggest downloading and installing the Zend Core for IBM to use PHP with Informix, because Zend Core for IBM is a full integration of the available driver for the database server.

Perl database interface

To better understand how the interface works, let us examine the PERL database interface (DBI). A Perl program uses a standard API to communicate with the DBI module for Perl, which supports only dynamic SQL. It defines a set of methods, variables, and conventions that provides a consistent database interface that is independent of the actual database being used. DBI gives the API a consistent interface to any database that the programmer wants to use. *DBD::Informix* is a Perl module, which, when used in conjunction with DBI, allows Perl to access the Informix database.

Figure 6-1 illustrates the Perl/Informix environment.

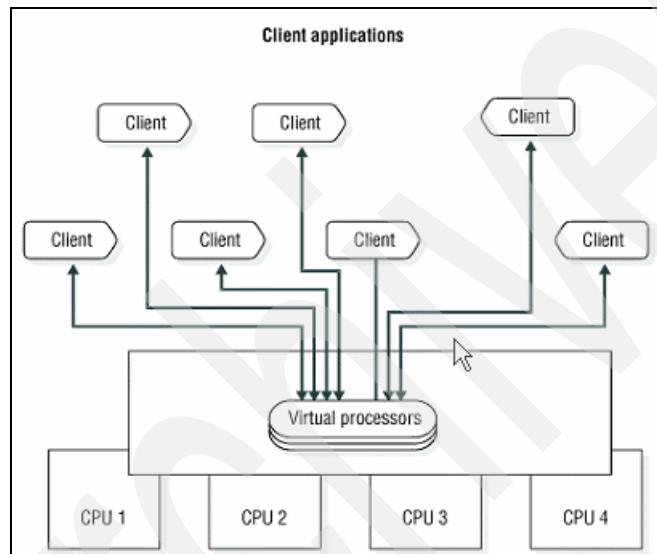


Figure 6-1 Perl/Informix invocation and data flow

Tcl/Tk and the Informix (isqltcl) extension

An extension is available to access the Informix database server within a Tool Command Language and Tk GUI toolkit (Tcl/Tk) programming environment. The name of the extension is *isqltcl*. You can obtain it on the web by using the following URL for the download:

<http://isqltcl.sourceforge.net/>

6.3 IBM Data Server Common Client for Informix

An IBM data server client is an application that allows you to run commands and SQL statements against IBM data servers, including IBM Informix and DB2. It allows you to connect to a remote Informix or DB2 server and access its databases.

Several types of IBM data server clients and drivers are available. Each type provides a particular type of support. The following IBM data server client and driver types are available to you:

- ▶ IBM Data Server Driver for JDBC and SQLJ
- ▶ IBM Data Server Driver for ODBC and Call Level Interface (CLI)
- ▶ IBM Data Server Driver Package
- ▶ IBM Data Server Runtime Client
- ▶ IBM Data Server Client

Each IBM data server client and driver provides a particular type of support:

- ▶ For Java applications only, use IBM Data Server Driver for JDBC and SQLJ.
- ▶ For applications using ODBC or CLI only, use IBM Data Server Driver for ODBC and CLI, which is referred to as CLI driver, also.
- ▶ For applications using ODBC, CLI, .NET, OLE DB, open source, or Java applications, use IBM Data Server Driver Package.
- ▶ For client support for running and deploying applications, use IBM Data Server Runtime Client.
- ▶ If you need support for database administration, and application development using an application programming interface (API), such as ODBC, CLI, .NET, or JDBC, use IBM Data Server Client.

However, note that the IBM data server client does not support two connection types to Informix database servers:

- ▶ OLE DB applications
- ▶ C/C++ applications

You can continue to use the IBM Informix Client SDK for these two types of applications.

Table 6-1 on page 266 shows the platform availability of the IBM data server client:

Table 6-1 Platform availability of the IBM data server client

Operating system	32/64-bit	Processor
AIX	64-bit	pSeries®
HP-UX	64-bit	Itanium®-based HP Integrity Series
Linux	32-bit	x86-32 on AMD and Intel®
Linux	64-bit	x86-64 on AMD64 and Intel EM64T
Linux	64-bit	POWER® (PowerPC®, iSeries®, or pSeries)
Linux	64-bit	System z®, System z9®, or zSeries®
Solaris SPARC	64-bit	UltraSPARC or SPARC64
Solaris x86-64	64-bit	Intel 64 or AMD64
Windows x86	32-bit	Intel or AMD
Windows x86	64-bit	AMD64 and Intel EM64T

You can download the compressed file for the latest version of the IBM Data Server Packages from the following website:

http://www.ibm.com/support/entry/portal/Downloads/Software/Information_Management/IBM_Data_Server_Client_Packages

6.3.1 IBM Data Server Driver for JDBC and SQLJ

IBM Data Server Driver for JDBC and SQLJ is the default driver for Java stored procedures and user-defined functions. This driver provides support for client applications and applets that are written in Java using JDBC to access local or remote servers, and SQLJ for embedded static SQL in Java applications. Because SQLJ can interoperate with JDBC, an application program can use JDBC and SQLJ within the same unit of work.

IBM data server support for JDBC lets you write Java applications that access local Informix data or remote relational data on a server that supports Distributed Relational Database Architecture (DRDA). For connections to Informix, SQL statements in JDBC or SQLJ applications run dynamically. You can access data in Informix database systems using JDBC, SQL, or pureQuery.

Informix supports only the type 4 version of the IBM Data Server Driver for JDBC and SQLJ, which means that sessions connect directly to the database or database server, without a middle layer.

Two versions of the IBM Data Server Driver for JDBC and SQLJ are available for Informix:

- ▶ IBM Data Server Driver for JDBC and SQLJ, Version 3.50, is JDBC 3.0-compliant.
- ▶ IBM Data Server Driver for JDBC and SQLJ, Version 4.0, is JDBC 3.0-compliant and supports several JDBC 4.0 functions.

6.3.2 IBM Data Server Driver for ODBC and CLI

IBM Data Server Driver for ODBC and CLI is a lightweight deployment solution that is designed for independent software vendor (ISV) deployments. This driver, which is also referred to as the *CLI driver*, provides runtime support for applications using the ODBC API, or the CLI API, without needing to install the Data Server Client or the Data Server Runtime Client. This driver is available only as a tar file, not as an installable image. Messages are reported only in English.

The IBM Data Server Driver for ODBC and CLI provides these functions:

- ▶ Runtime support for the CLI API
- ▶ Runtime support for the ODBC API
- ▶ Runtime support for the XA API
- ▶ Database connectivity
- ▶ LDAP Database Directory support
- ▶ Tracing, logging, and diagnostic support

The IBM Data Server Driver for ODBC and CLI supports open source drivers for IBM data servers.

The Call Level Interface (CLI) component of the IBM Data Server Driver for ODBC and CLI supports connections to Informix servers for certain open source drivers. The CLI component of this driver is required if you plan to use one of the following open source drivers for IBM data servers:

- ▶ PDO for Data Server Clients for PHP connections

Informix supports database access for client applications written in the PHP programming language by using an Informix PHP Data Object (PDO) extension that functions as a database extraction layer.

The PDO extension for IBM data servers is called *PDO_IBM* and is supported on Informix Version 11.10 or later. You can use *PDO_IBM* for connections to an Informix database server.

You can download *PDO_IBM* from the PHP Extension Community Library (PECL) website:

http://pecl.php.net/package/PDO_IBM

To use the PDO_IBM, you must first install the IBM Data Server Driver for ODBC and CLI.

► Ruby Gem for Data Server Clients

Informix supports database access for client applications written in the Ruby programming language with two open source packages. Each package, known as a *ruby gem*, is a precompiled file that combines the Ruby driver and the Ruby on Rails adapter in a standard framework for the Ruby runtime environment.

The *IBM_DB* gem is the Ruby gem for IBM data servers. The *IBM_DB* gem is supported on Informix Version 11.10 or later. Based on the DRDA protocol, it allows Ruby applications to access Informix database servers.

To use the *IBM_DB* gem, you must first install the IBM Data Server Driver for ODBC and CLI.

The Ruby Informix Gem works only for connection with Informix. You can download Ruby Informix Gem from the Rubyforge website:

<http://rubyforge.org/projects/rubyibm>

6.3.3 IBM Data Server Driver Package

IBM Data Server Driver Package provides a lightweight deployment solution providing runtime support for applications using ODBC, CLI, .NET, OLE DB, open source, or Java APIs without needing to install Data Server Runtime Client or Data Server Client. This driver has a small footprint. It is designed to be redistributed by independent software vendors (ISVs) and to be used for application distribution in mass deployment scenarios that are typical of large enterprises.

The IBM Data Server Driver Package includes these capabilities:

- Support for applications that use ODBC, CLI, or open source (PHP or Ruby) to access databases.
- Support for client applications and applets that are written in Java using JDBC, and for embedded SQL for Java (SQLJ).
- IBM Informix support for .NET, PHP, and Ruby.
- Application header files to rebuild the open source drivers.
- On Windows operating systems, IBM Data Server Driver Package also provides support for applications that use .NET or OLE DB to access databases. In addition, this driver is available as an installable image, and a merge module is available to allow you to embed the driver easily in a Windows Installer-based installation.

On Linux and UNIX operating systems, IBM Data Server Driver Package is not available as an installable image.

6.3.4 IBM Data Server Runtime Client

The IBM Data Server Runtime Client provides a way to run applications on remote databases. GUI tools do not ship with the IBM Data Server Runtime Client.

The IBM Data Server Runtime Client includes these capabilities:

- ▶ Base client support to handle database connections, SQL statements, XQuery statements, and commands.
- ▶ IBM Informix support for PHP, Ruby, .NET, and JDBC.
- ▶ Support for common database access interfaces: JDBC, ADO.NET, OLE DB, ODBC, Command Line Interface (CLI), PHP, and Ruby. This support includes drivers and capabilities to define data sources. For example, for ODBC, installing an IBM data server client installs the ODBC driver and registers the driver. Application developers and other users can use the Windows ODBC Data Source Administrator tool to define data sources.
- ▶ Lightweight Directory Access Protocol (LDAP) exploitation.
- ▶ Support for common network communication protocols: TCP/IP and Named Pipe.
- ▶ Support for installing multiple copies of a client on the same computer. These copies can be the same or separate versions.
- ▶ License terms that allow free redistribution of IBM Data Server Runtime Client with your application.
- ▶ Smaller deployment footprint compared to that of the full IBM Data Server Client in terms of the installation image size and the required disk space.
- ▶ A catalog that stores information for connecting to databases and servers.
- ▶ Packaging advantages on Windows operating systems: You can package the client with your application to provide connectivity for that application. Also, the client is available as Windows Installer merge modules that enable you to include the runtime client (RTCL) dynamic link library (DLL) files in your application installation package. This approach also enables you to include only the parts of the client that you need with your application.

6.3.5 IBM Data Server Client

IBM Data Server Client includes all the functionality of IBM Data Server Runtime Client, plus functionality for database administration, application development, and client/server configuration.

The IBM Data Server Client includes the following capabilities:

- ▶ A larger deployment footprint compared to that of IBM Data Server Runtime Client in terms of installation image size and required disk space. However, on Windows operating systems, you can prune the IBM Data Server Client image to reduce the installation image size.
- ▶ IBM Informix support for PHP, Ruby, .NET, Java Common Client (JCC) driver, and JDBC
- ▶ First Steps documentation for new users
- ▶ Visual Studio tools
- ▶ IBM Data Studio
- ▶ Application header files
- ▶ Precompilers for various programming languages
- ▶ Samples and tutorials

The IBM Data Server Client, together with the IBM Data Server Package and the IBM Data Server Runtime Client, includes the .NET driver, IBM Data Server Provider for .NET.

The IBM Data Server Provider for .NET extends data server support for the ADO.NET interface. The provider delivers high-performing, secure access to IBM data servers.

In addition to the IBM Data Server Provider for .NET, the IBM Database Add-Ins for Visual Studio enable you to quickly and easily develop .NET applications for IBM data servers using Microsoft Visual Studio. You can also use the add-ins to create database objects, such as indexes and tables, and to develop server-side objects, such as stored procedures and user-defined functions.

The Informix provider is included in the IBM Data Server Provider for .NET client package. It is sometimes referred to as the *Common .NET Providers*. With the Informix .NET Provider (IBM.Data.Informix.dll), your .NET applications can access IBM Informix Version 11.10 and later.

The IBM Informix .NET Provider has been certified to work on Windows Server 2003, Windows XP, Windows Vista, and Windows Server 2008. The IBM Informix .NET Provider requires that Microsoft .NET Framework SDK 1.1 or higher is

already installed on the computer. You can also install the IBM Data Server Provider for .NET when you install IBM Informix Client SDK (CSDK) or IBM Informix Connect.

6.4 Visual Studio Add-ins

Visual Studio is an Integrated Development Environment (IDE) from Microsoft. The Informix Database Add-Ins for Visual Studio are a collection of features that integrate seamlessly into your Visual Studio development environment so that you can connect and work with Informix.

This section provides an overview of the features that have been added to Informix Add-ins for Visual Studio to work with IBM Informix (Version 11 and later). We also discuss the installation and configuration, as well as simple examples to work with Visual Studio Add-ins.

6.4.1 Overview of Visual Studio Add-ins

Developers working on the Visual Studio environment can now use the IBM Database Add-ins for Visual Studio to connect to Informix and take advantage of the rich set of features offered by the IBM Database Add-ins, as well as use the standard Visual Studio features.

Several key features are now supported for Informix V11.10 to provide these capabilities:

- ▶ **Integrate with Microsoft Server Explorer:** Add a data connection to an Informix database server using the Microsoft Server Explorer. The data connection lets you browse the database objects and its properties.
- ▶ **Develop Windows applications:** Build Windows applications for Informix using standard drag-and-drop mechanisms, as supported by the Visual Studio 2005 environment.
- ▶ **Develop websites:** Build websites for Informix using standard Visual Studio 2005 functionalities.
- ▶ **Create and delete database objects:** Non-modal designers are provided for creating several types of database objects, as listed. Non-modal designers allow you the flexibility to simultaneously work on designing several database objects. Deletion of these objects is also supported.
 - Create and delete tables.
 - Create and delete views.
 - Create and delete stored procedures.

- Create and delete user-defined functions.
 - Create and delete triggers.
 - Create and delete indexes.
 - Create and delete user privileges for the database objects.
- ▶ View and update data for tables and views: Show, as well as update, data for tables and views.
 - ▶ Execute procedures and functions: Execute stored procedures and functions, and view the returned result sets.
 - ▶ Create and delete Informix databases: Create and delete Informix databases directly from the Visual Studio 2005 IDE.
 - ▶ View and execute scripts: View the create scripts/Data Definition Language (DDL) for database objects, and execute arbitrary scripts on the database.

For a detailed discussion of Visual Studio Add-ins, refer to the following web pages:

- ▶ <http://www.ibm.com/developerworks/data/zones/vstudio/>
- ▶ <http://www.ibm.com/developerworks/data/library/techarticle/dm-0710jayakumar/>

6.4.2 Installing Visual Studio Add-ins

You are required to install one of the following products:

- ▶ IBM Data Server Client (contains both Add-ins and Informix .NET provider).
- ▶ IBM Database Add-ins for Visual Studio (if you already have IBM Data Server Driver installed).

Note that one of the installation requirements is IBM Data Server Provider; therefore, you need to download this file, as well. You can download all these files from the following web pages:

- ▶ http://www.ibm.com/support/entry/portal/Downloads/Software/Information_Management/IBM_Data_Server_Client_Packages
- ▶ <http://www-306.ibm.com/software/data/informix/tools/csdk/>

The common method for installing a client and driver is to run the installation program that is provided on the downloaded images.

6.4.3 Configuring Visual Studio Add-ins

In Visual Studio 2005 IDE, the Server Explorer lists the data connections that have been added. In Visual Studio 2008 IDE, you need to select **View** → **Server Explorer** to get into the Server Explorer View.

You can then right-click **Data Connection** to add data connections to Informix using the **Add Connection** → **context menu**, as shown in Figure 6-2, and Figure 6-3.

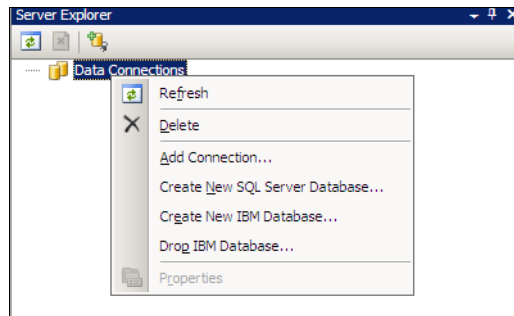


Figure 6-2 Add connection

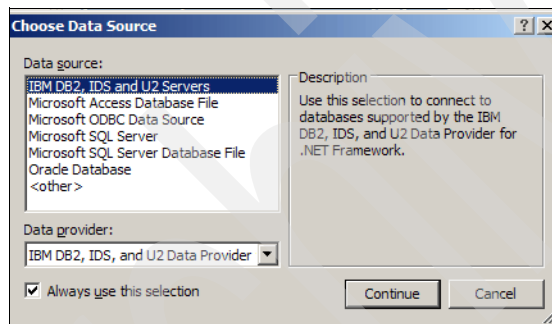


Figure 6-3 Data source selection context menu

Invoking this menu brings up the Add Connection dialog, as shown in Figure 6-4 on page 274. The Add Connection dialog is specific to the IBM Database Add-ins and allows the following customizations:

- ▶ An option to specify a schema filter that uses the LIKE search criteria. The schema filter applies to all the folders in the tree, and hence, only objects belonging to this schema are displayed in the tree.
- ▶ An option to exclude system schemas.
- ▶ Apart from the schema filter, you can achieve a second level of filtration by specifying folder filters for each of the object folders.
- ▶ An option to include or exclude the various folders in the tree.
- ▶ An option to automatically refresh the connection when the Visual Studio IDE is loaded.

- An option to always run procedures to discover result sets.
- An option to persist the password across Visual Studio sessions.
- An option to test the connection using the Test Connection button.

Add Connection

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source:
IBM DB2, IDS and U2 Servers (IBM DB2, IDS, and U2 Data Provider) Change...

Specify the following to connect to IBM data:

1. Select or enter server name:
IDS_ServerName:portNumber Refresh

2. Enter information to log on to the server:
User ID: testUser
Password: *****
☒ Save my password

3. Select or enter a database name
myIDSDatabase

☐ Specify connection options:
☐ Automatic refresh of connection when IDE loads
☒ Run procedures to discover result sets

☐ Specify filtering options:
Schema filter: ☒ Exclude system schemas
Table type filter: ☐

Include Folder	Folder Name	Object Name Filter
☒	Tables	
☐	...	

Advanced...

Test Connection OK Cancel

Figure 6-4 Add Connection dialog

IDS_ServerName refers to the host name of the server where Informix resides. You can enter the IP address here. The port number refers to the port to which Informix is listening. Use the port with the DRDA protocol. For more detail about the configuration of DRDA protocol in Informix connectivity, refer to 7.2.3, “Configure connectivity” on page 327.

The connection string is automatically generated. To see a read-only view of the connection string in the Advanced Properties dialog, select **Advanced** in the Add Connection Dialog. After a connection is created, a new node for the connection is created and displayed in the Server Explorer. Expand the connection node to display the various nodes for the database objects (tables, views, functions, and

procedures), depending on the folder filters that were specified. You can further expand each of the folders to view an enumeration of the contained database objects. The properties window in the IDE displays the properties for the object that is in focus.

You can optionally modify an existing data connection using the **Modify Connection** → **context** menu, which invokes the same Add Connection dialog pre-filled with the connection information.

6.4.4 Server Explorer integration

After adding a new connection, all schema information is asynchronously pre-fetched and cached, which allows rapid access to this information when designing applications. These fetched server objects are listed as folders under the corresponding connection.

Table enumeration, which is part of the integration, provides useful information about a database object. Expanding the Tables folder lists the available tables in the database, depending on whether the Filter option or Exclude system schema option is supplied in the Add Connection window. As shown in Figure 6-5 on page 276, by expanding a particular table, you can see the list of columns with which the table was built.

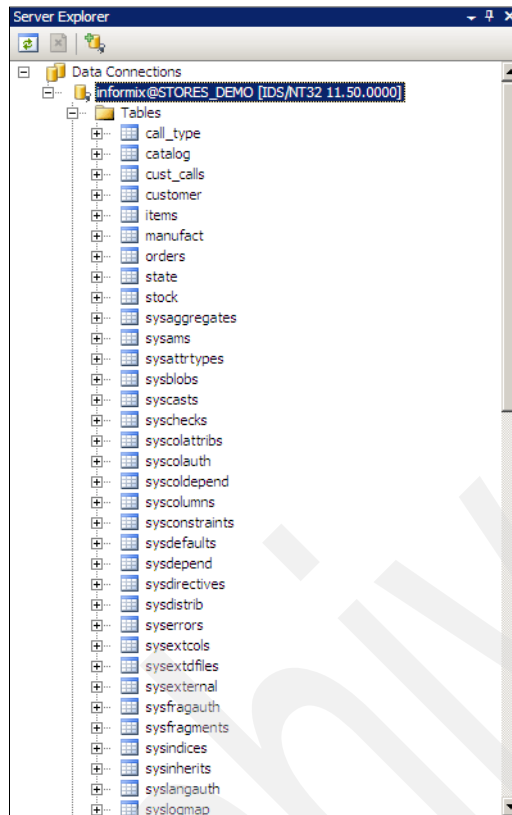


Figure 6-5 Table enumeration

The Property window lists the important properties of the database object that was selected in the Server Explorer. For example, if you select the `fname` column of the `customer` table of the `stores_demo` database, the corresponding Property window appears (see Figure 6-6 on page 277).

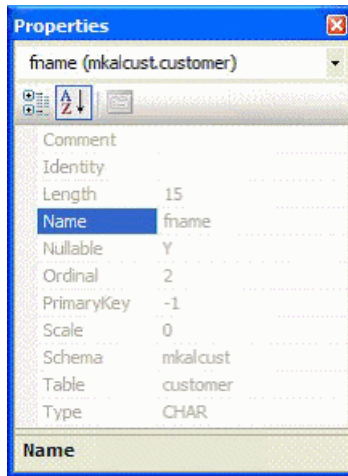


Figure 6-6 Properties window

The IBM Database Add-ins for Visual Studio provides several IBM Designers that have been customized to create and work with the various Informix database objects, such as tables, views, stored procedures, functions, triggers, and indexes. The designers, unlike wizards, are non-modal and, hence, allow you the flexibility to simultaneously design various database objects.

In Server Explorer, right-click the data object with which you want to work. Common views, such as Show Script and Show Data View, are available. See Figure 6-7 on page 278 for the common designers option.

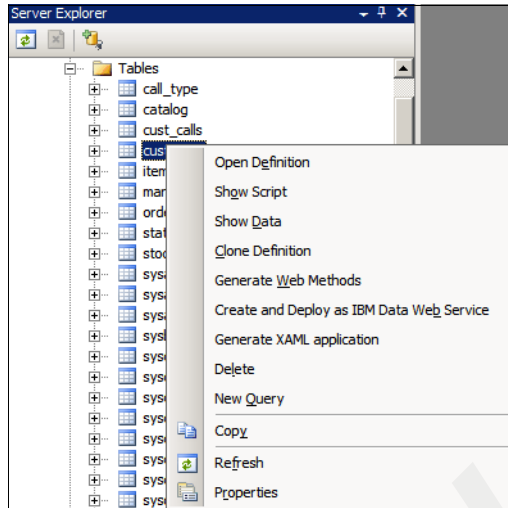


Figure 6-7 IBM Designers for Informix database objects

6.4.5 Windows application development

You can develop applications seamlessly for IBM Informix using the new paradigm introduced in Visual Studio for building Windows applications. Follow this simple procedure to build a Windows application:

1. Add a connection to an Informix database.
2. Create a new Windows application using the **File** → **New** → **Project** menu. Call it customerApp, as shown in Figure 6-8 on page 279. In this example, we create a C# application.

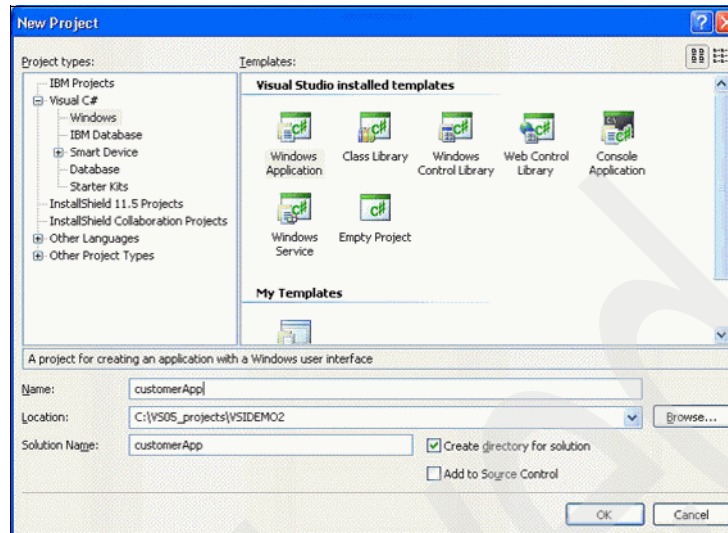


Figure 6-8 Creating a new C# Windows application

3. Create a data source for the customer table:
 - a. Using the top-level menu, choose **Data** → **Add New Data Source** to invoke the wizard. See Figure 6-9.

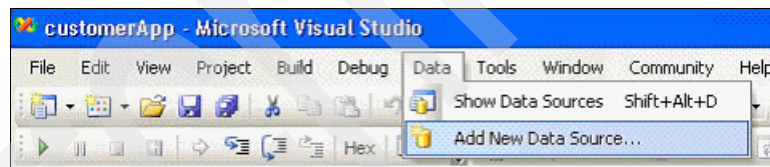


Figure 6-9 Add New Data Source wizard

- b. In the wizard, keep the default selection for Database. Click **Next**.
 - c. From the list of connections, select the connection to see what was created earlier. Select the option to include sensitive data in the connection string, as shown in Figure 6-10 on page 280. Click **Next**.

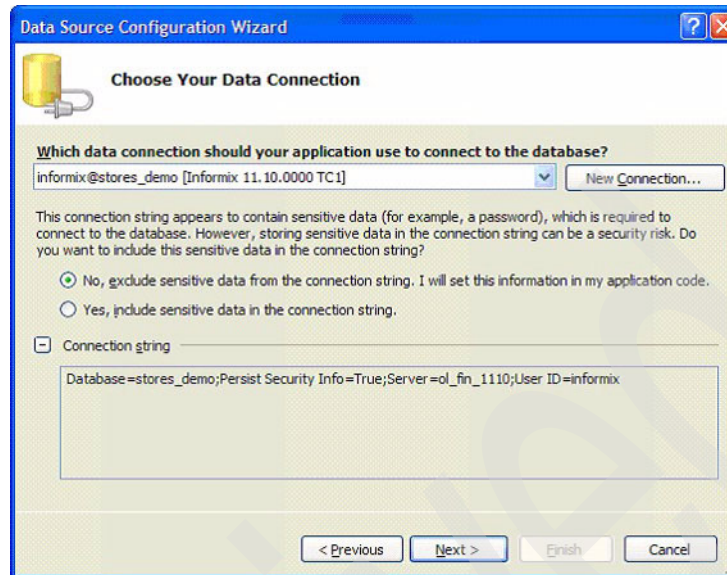


Figure 6-10 Select database

- d. Optionally, check the box to save the connection string as stores_demoConnectionString. Click **Next**.
- e. Select the Customer table from the list of tables, as shown in Figure 6-11 on page 281, and click **Finish**.

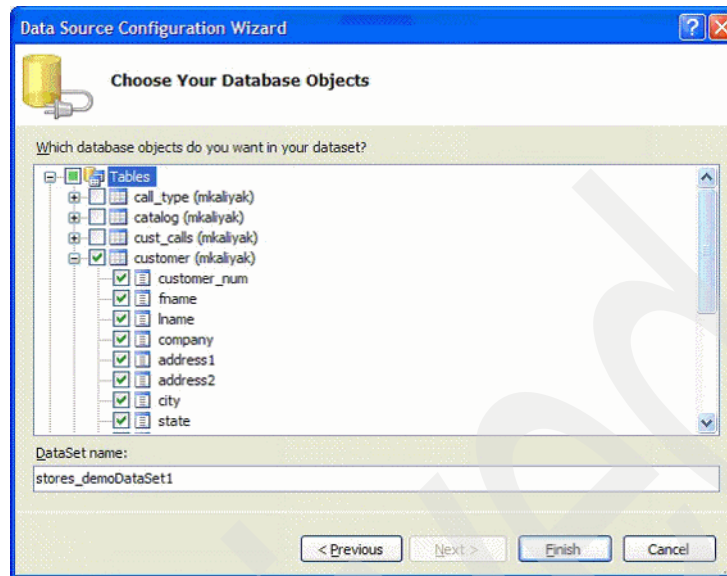


Figure 6-11 Select table

- f. A data source for the Customer table is added in the Data Sources window (Figure 6-12). You can make the Data Sources window visible by using the top-level menu, and selecting **Data** → **Show Data Sources**.

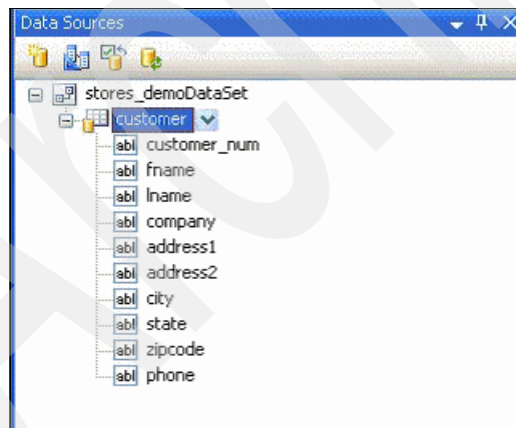


Figure 6-12 Customer data source added

4. Design the Windows application:
 - a. Right-click **Form1.cs**, and open it in design mode by choosing **View Designer**. See Figure 6-13 on page 282.

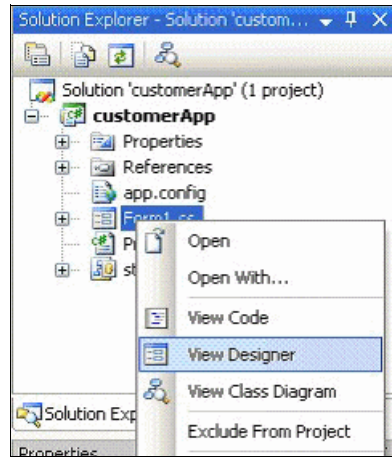


Figure 6-13 Form in design mode

- b. Drag and drop the department data source from the Data Sources window onto the form. Notice the default-generated user interface, as shown in Figure 6-14.

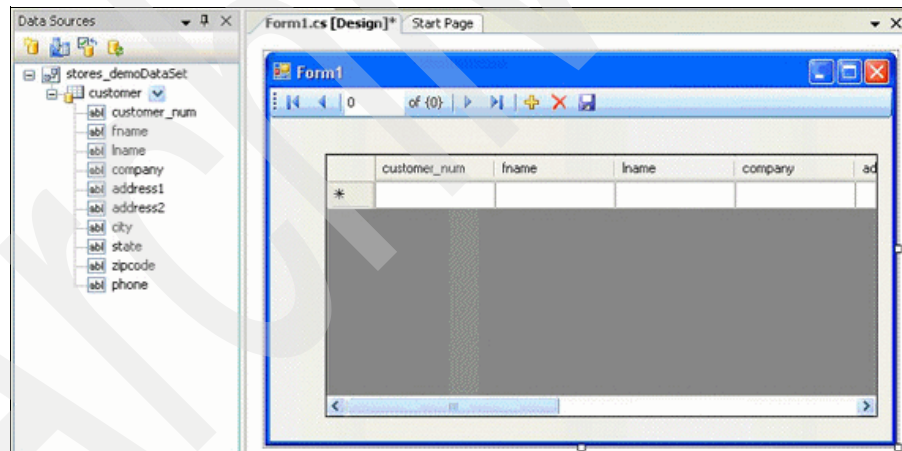


Figure 6-14 Designed Windows form

5. Run the Windows application, as shown in Figure 6-15 on page 283.

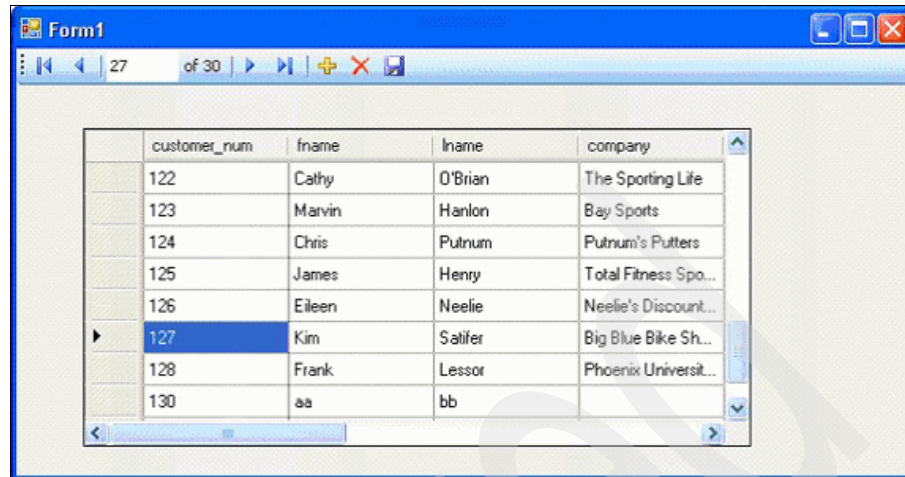


Figure 6-15 Running customerApp

You can use the pre-created user interface to navigate the rows. You can also add new rows, delete rows, and update existing rows. Click **Save** to save the changes to the server.

6.4.6 Windows application development through DataSet creation

Follow these steps to develop an application using DataSet creation:

1. Using the Microsoft Solution Explorer, right-click **customerApp**, and select **Add → New Item**. See Figure 6-16.

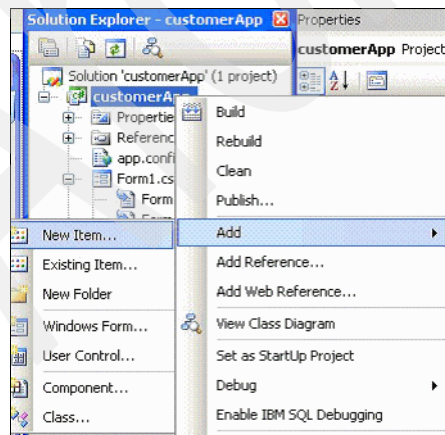


Figure 6-16 Add New Item

2. Select **DataSet**. Click **Add**. The Microsoft DataSet designer opens. See Figure 6-17.

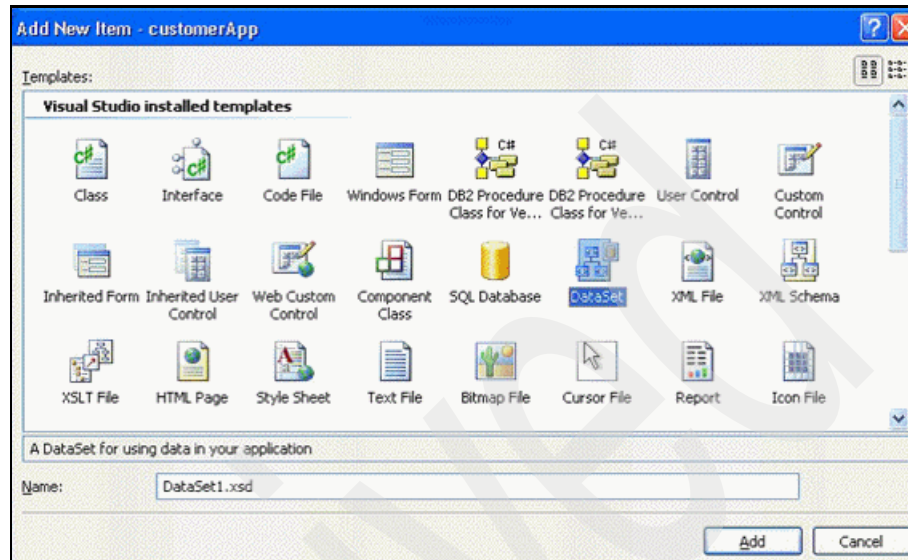


Figure 6-17 Add New DataSet

3. From the Microsoft Server Explorer, drag and drop another table, which is named orders, onto the open designer. This action creates the data source for orders.
4. You can drag and drop the newly created data source onto a fresh form, as mentioned in Steps 4 and 5 of the Windows application development section, by running the form. You can perform insert, update, and delete operations in the table orders.

IBM Informix provides new support in the Visual Studio tooling. The new features include seamless integration with Microsoft Server Explorer and rapid Windows application development for Informix servers that use Informix .NET 2.0 Provider. You can take your .NET application development experience to the next level.

6.5 Migrating in-house applications

In-house-developed (user-built) applications are unique in every case. There are a variety of languages used in these applications and each language has its unique way of using APIs. The languages use either native database interfaces, such as those database interfaces for embedded SQL, or access the database with common standards, such as ODBC, JDBC, or the .NET Framework. In this

section, we focus on application migration where you also want to apply code changes.

We describe the steps for converting in-house applications from SQL Server to Informix, and we provide examples in C/C++ and Java, which show you how to convert the database calls.

We start the in-house database application conversion discussion with a brief introduction and the definition of a planning strategy. Then, we follow with a verification of the existing programming techniques for using database interfaces. We complete the topic with an overview of porting issues typically required by the most commonly used database application development environments. We also introduce a sample database application task and show the differences in the specification of parameters and implementation in the source.

Because each database client API provides a wide variety of functions and parameters, which can be used for connection management, executing statements, processing the result sets, and error handling, presenting them all is beyond the scope of this document. We instead provide links to other sources of information required for your specific porting needs.

6.5.1 Application migration planning for in-house applications

Before you migrate existing user written applications in your database environment, develop a strategy for the migration. This strategy defines the major steps you must follow in this process. In addition, the strategy needs to include the definition of goals, user resources, and timestamps to track the status and to result in a successful migration.

This migration strategy must include the following tasks:

- ▶ Determine software and hardware availability and compatibility.
- ▶ Educate developers and administrators.
- ▶ Analyze application logic and source code.
- ▶ Set up the target environment.
- ▶ Change database-specific items.
- ▶ Test the applications.
- ▶ Tune the applications.
- ▶ Roll out the migrated applications.
- ▶ Educate the users.

Also, create a project plan, which includes sufficient time and resources for each task. IBM and IBM Business Partners can help you with this activity to ensure that you have a well-defined and complete project plan.

Determine software and hardware availability and compatibility

The architecture profile is one of the outputs of the first tasks in the migration planning assessment. When preparing the architecture profile, determine the availability and compatibility of all software and hardware in the new environment.

Educate developers and administrators

Understanding the new products is essential for analyzing the source system. Ensure that the staff gets educated and has the skills for all products and the system environment that you will use for the migration project.

Analyze application logic and source code

In this analysis phase, identify all the SQL Server proprietary features and the affected sources. You also need to analyze the database calls within the application for the usage of database APIs.

Set up the target environment

You must set up the target system, whether it is the same or another system, for application development. The environment can include the following parts:

- ▶ The Integrated Development Environment (IDE)
- ▶ Database framework
- ▶ Repository
- ▶ Source code generator
- ▶ Configuration management tool
- ▶ Documentation tool

A complex system environment usually consists of products from a number of vendors. Check the availability and compatibility issues before starting the project.

Change database-specific items

Regarding the use of the database API, you need to change the database calls in the applications:

- ▶ Language syntax changes

The syntax of database calls varies by programming language. In 6.5, “Migrating in-house applications” on page 284, we discuss the varieties of C/C++ and Java applications. For information regarding other languages, contact your IBM marketing representative or IBM technical support.

- ▶ **SQL query changes**

The SQL Server syntax is predominantly the same as the Informix syntax. Most situations require minimal or no changes. More information about possible SQL query changes is available in Chapter 4, “SQL considerations” on page 67.

- ▶ **Changes in calling procedures and functions**

Sometimes, you must change procedures to functions and vice versa. In such cases, you have to change all the calling commands, and the logic belonging to the calls, in the database and the applications.

- ▶ **Logic changes**

Because of architectural differences between SQL Server and Informix, changes in the program flow might be necessary. Most of the changes will relate to the difference in concurrency models.

Test the applications

A complete application test is necessary after the database conversion, along with any application modifications, to ensure that the database conversion is complete and that all the application functions work properly.

It is prudent to run the migration several times in a development system to verify the process. Run the same migration on a test system with existing test data and, then, on a copy or subset of production data. When all results are positive, consider running the process in a production environment.

Tune the applications

Tuning is a continuous activity for the database, because the data volume, number of users, and applications change over time. After the migration, perform application tuning. Having a good understanding of the architectural differences between SQL Server and Informix is required for a good understanding of how to perform application tuning. For more details, refer to the *IBM Informix Dynamic Server Performance Guide, v11.50, SC23-7757*.

Roll out the migrated applications

The rollout procedure varies depending on the type of application and the type of database connection that you have. Prepare the workstations with the proper drivers (for example, IBM Informix Client SDK or IBM Informix Connect) and server that are consistent with the Informix version.

Educate the users

In dealing with changes in the user interface, the business logic, and the application behavior that are caused by system improvements, user education is

required. Provide sufficient user education, because the acceptance of the target system is largely determined based on the skills and satisfaction of the users.

6.5.2 Converting C/C++ applications

Microsoft Embedded SQL for C (ESQL/C) translates embedded SQL statements to the appropriate DB-Library API function calls. The Microsoft ESQL/C API is not available after SQL Server 2000.

IBM Informix embedded SQL (ESQL/C) is supported to work with the latest Informix version. It is part of the IBM Informix Client SDK product and provides a rich set of programming techniques, such as static and dynamic SQL, the full support of handling all supported Informix data types, and an interface for user-defined routines (UDRs) and user-defined types (UDTs).

In the following section, we give you guidance about where to focus on existing application code to achieve a successful migration to the target database server. This discussion includes the following areas:

- ▶ Connection and authentication
- ▶ Declaration and usage of host variables
- ▶ Exception handling
- ▶ SQL execution status monitoring

Connecting to the database and user authentication

Microsoft ESQL/C supports a simple singleton connection. IBM Informix ESQL/C is multithreaded and able to manage multiple connections in the same client.

Simple singleton connect

You can use the CONNECT TO statement, along with the server name, database name, user name, and password, to connect to an SQL Server database. If the server name is not specified, the local server is assumed. The password is optional, and the user name can be replaced by \$integrated to use Windows Authentication.

Users are treated differently in Informix. For example, there are no separate user objects in the server, and the authentication is done by the database server with the help of the base operating system. There are two methods to handle connection requests to a database. Connection requests can be made with a trusted connection to the remote database server; there is an entry for the server where the client resides on the target .rhosts or hosts.equiv file. In this case, the user does not need to specify a password. The second way is to use the connect SQL statement and specify a user and a password to use a non-trusted connection.

Example 6-1 shows the SQL statement to use to establish the connection to the database server in an client application.

Example 6-1 CONNECT TO syntax using ESQL/C in SQL Server and Informix

```
// SQL Server CONNECT TO Syntax Examples:
EXEC SQL CONNECT TO servername.dbname USER "userid";
EXEC SQL CONNECT TO servername.dbname USER "userid.password";
EXEC SQL CONNECT TO servername.dbname USER $integrated;

# // Informix CONNECT TO Syntax options:
#Using a trusted connection
$database <database_name>
$connect to "database@dbservername";

#Using a connection with user and password specification
$connect to "database@dbservername" USER :userid USING :password;
```

Note that password needs to be declared as a host variable. You can use a hard-coded user ID and database name, and you are allowed to specify them with a host variable. In the example, we used a static database name and a host variable for the user.

IBM Informix Client SDK 3.50 supports the use of single sign-on (SSO) authentication to Informix Version 11.50. With SSO support, users enter their passwords one time when they log in to the operating system, and thereafter, they have access to multiple software systems without needing to reenter their user name and password.

Multiple connections in the same client

IBM Informix ESQL/C supports the specification of multiple concurrent transactions in the same client at the same time. But there are differences in the specification and in the handling of attaching the statements to the appropriate session.

IBM Informix ESQL/C provides the naming of a session, but the reference of the current statement is handled differently. The client provides a statement to specify a name for the session, as shown in Example 6-2.

Example 6-2 Named connections in Informix ESQL/C

```
$Connect to "database_name" as "CONN_NAME" USER :user USING :password;
$set connection "CONN_NAME";
$SELECT col1 INTO :var1 FROM table_name;
$set connection "CONN_NAME1";
```

Host variable declaration

Host variables are C or C++ language variables that are referenced within SQL statements. They allow an application to pass input data to and to receive output data from the database server. After the application is precompiled, host variables are used by the compiler as any other C/C++ variable.

Host variables must be compatible with Informix SQL data types (accepted by the esql precompiler that ships with the IBM Informix Client SDK product), but they also must be acceptable for the programming language compiler.

Because the C program manipulates the values from the tables using host variables, the first step is to convert the SQL Server host variable definitions to Informix data types. See Appendix B, “Data types” on page 409 for more details.

General definition of data types

Following the ANSI standard for embedded programming, you must declare all host variables in a C program in a special declaration section. This way, the IBM Informix ESQL/C precompiler can identify the host variables and the data types, for example:

```
EXEC SQL BEGIN DECLARE SECTION;
char emp_name[31];
int ret_code = 0;
EXEC SQL END DECLARE SECTION;
```

You are also allowed to use declarations that do not follow the ANSI standard, such as the following example:

```
int some_c_variable=0;
$int some_esqlc_variable=0;
char some_c_char[100];
$char some_esqlc_char[100];
```

Define host variables that are based on user-defined C structures

There are no differences in the definition of host variables based on user-defined C structures. Make sure that the structure is included in the declare section for the host variables.

For the implementation in IBM Informix ESQL/C, you must include the typedef in the declare section. This way, the preprocessor is able to identify the types of the members of the structure that are used in the subsequent SQL statements. See Example 6-3 on page 291.

Example 6-3 Define a host variable based on a structure in ESQL/C

```
$include sqlca;
main()
{
EXEC SQL BEGIN DECLARE SECTION;
int cust_no=110;

struct customer {
    integer customer_num;
    char lname[20];
    char fname[20];
};
typedef struct customer customer_t;
customer_t cust_var;
EXEC SQL END DECLARE SECTION;

$database stores_demo;
$select customer_num, lname,fname
into :cust_var.customer_num, :cust_var.lname, :cust_var.fname
from customer where customer_num=:cust_no;
printf(" Output customer_num: %d , name %s fname %s \n",
    cust_var.customer_num, cust_var.lname, cust_var.fname);
}
```

Using pointers in host variables

To advance the example, we also can use a pointer as a host variable in the embedded SQL statement. We show the definition of the host variable as a pointer and its usage in Example 6-4.

Example 6-4 Using a pointer to a structure as a ESQL/C host variable

```
$include sqlca;

#include <stdio.h>
#include <malloc.h>

main()
{
    $int cust_no=110;
    $struct customer {
        integer customer_num;
        char lname[20];
        char fname[20];
    };
    $typedef struct customer customer_t;
    $customer_t *cust_var;

    cust_var=(customer_t *)malloc(sizeof(customer_t));
```

```

$database stores_demo;
$select customer_num, lname, fname
      into :cust_var->customer_num, :cust_var->lname, :cust_var->fname
      from customer where customer_num=:cust_no;

printf(" Output customer_num: %d , name %s fname %s \n",
      cust_var->customer_num, cust_var->lname, cust_var->fname);
}

```

Using stored procedures in an embedded C program

Both SQL Server and Informix have a similar method to invoke a stored procedure; see the following example:

```

/* singleton row with cursor */
EXEC SQL declare c1 cursor for execute procedure SP_name (:arg_in1);
EXEC SQL open c1;
while (SQLCODE == 0)
{
EXEC SQL fetch c1 into :arg_out1;
/* singleton row */
EXEC SQL execute procedure SP_name (:arg_in1) into :arg_out1;
}

```

Default stored procedures that are defined in the Informix database server do not allow the OUT parameter in the parameter list. You can only use them in user-defined functions (UDF) that are written in C or Java and provided by an external library. For an example of how to create a UDF, refer to Appendix C, “Function mapping” on page 419.

Exception handling in embedded environments

The exception handling in IBM Informix ESQL/C is again similar to SQL Server, with both products using the same concept of separating the error routines from the mainline logic. There are separate WHENEVER statements that you can use to define program behavior when there is an error in Informix, as shown in the following code:

```

EXEC SQL WHENEVER SQLERROR GOTO error_routine;
EXEC SQL WHENEVER SQLERROR STOP;
EXEC SQL WHENEVER ERROR STOP;
EXEC SQL WHENEVER SQLWARNING CONTINUE;
EXEC SQL WHENEVER NOT FOUND CALL not_found_routine;

```

Although the WHENEVER statement is prefixed by EXEC SQL, it is not an executable statement. Instead, a WHENEVER statement causes the precompiler to generate code in a program to check the SQLCODE attribute from the SQL Communication Area (SQLCA) after each SQL statement, and to perform the action specified in the WHENEVER statement. SQLERROR means that an SQL

statement returns a negative SQLCODE, indicating an error condition. SQLWARNING indicates a positive SQLCODE (except +100). NOT FOUND specifies SQLCODE = +100, indicating that no data rows were found to satisfy a request.

A compilation unit can contain as many WHENEVER statements as necessary, and you can place WHENEVER statements anywhere in the program. The scope of one WHENEVER statement reaches from the placement of the statement in the file onward in the character stream of the file until the next suitable WHENEVER statement is found, or end-of-file is reached. No functions or programming blocks are considered in that analysis. For example, you can have two separate SELECT statements. One SELECT statement must return at least one row, and the other SELECT statement might not return any rows. You need two separate WHENEVER statements, as shown in Example 6-5.

Example 6-5 WHENEVER statements

```
EXEC SQL WHENEVER NOT FOUND GOTO no_row_error;
      EXEC SQL SELECT address
                INTO :address
                FROM test_table
                WHERE phone = :phone_num;

.....
EXEC SQL WHENEVER NOT FOUND CONTINUE;
EXEC SQL SELECT commis_rate
                INTO :rate :rateind
                WHERE prod_id = :prodId;
      if (rateind == -1) rate = 0.15;
.....
...
```

Another alternative, in a comparison of using the WHENEVER statement, is to check SQLCODE explicitly after each EXEC SQL statement, because that method allows more context-sensitive error handling.

Error messages and warnings

The SQL Communication Area (SQLCA) data structure in IBM Informix ESQL/C provides diagnostic information and event handling.

Informix contains the SQL error code in the sqlcode member of the global SQLCA structure. If a further description of this error is required, you can use the rgetmsg() function. This function returns the message based on the provided error code. Example 6-6 on page 294 shows the usage of the function in a small C program.

Example 6-6 Using `rgetmsg()` function

```
$database stores_demo;  
$select customer_num, lname, fname  
        into :cust_var.customer_num, :cust_var.lname,      :cust_var.fname  
        from customer1 where customer_num=:cust_no;  
  
printf(" Output customer_num: %d , name %s fname %s \n",  
       cust_var.customer_num, cust_var.lname, cust_var.fname);  
  
printf("%ld \n",sqlca.sqlcode);  
  
rgetmsg((short)SQLCODE, errmsg, sizeof(errmsg));  
printf(errmsg, sqlca.sqlerrm);
```

Building ESQL/C-based applications

The IBM Informix Client SDK is the development environment in which you have to precompile, compile, and link your final application. This product supplies you with the build utilities, the libraries that are needed for the successful execution of the application, and a number of code samples for various programming tasks.

Building the application based on ESQL/C requires a number of internally referenced libraries in a certain order during the link process, which is why it is not advisable to use your own link scripts. Use the `esql` utility that ships with the Informix SQL to perform this task.

You can build your executable with shared libraries. At execution time, the application needs to know where the libraries reside. Depending on the operating system used, there are environment variables, such as `LIB_PATH`, where the linker looks for the libraries. Shared linked applications are smaller than static applications, but they require the installation of IBM Informix Client SDK or IBM Informix Connect on the target system. In addition, you can build your application as static. In this case, all used functionality is statically linked to the executable.

For more information about building embedded C applications, see the *IBM Informix ESQL/C Programmer's Manual*, SC23-9420.

6.5.3 Converting Visual Basic applications

For Visual Basic (VB) applications, the connection parameters are the same as the connection parameters in Example 6-7 on page 295. These providers do not require many changes in the application code when converting to Informix; however, you must investigate any additional usage restrictions that these providers might have.

SQL Server OLE DB:

```
OleDbConnection con = new OleDbConnection("Provider=SQLOLEDB;  
Data source=localhost;Integrated Security=SSPI;Initial Catalog=bookdb")
```

Informix OLE DB:

```
OleDbConnection con = new OleDbConnection("Provider=IFXOLEDBC;  
Data Source=bookdb@ServerName;UID=myusername;" + PWD=mypassword");
```

SQL Server ODBC:

```
OdbcConnection con = new OdbcConnection("Driver={SQL Server};  
Server=localhost;Trusted_Connection=yes;Database=bookdb");
```

Informix ODBC:

```
OdbcConnection con = new  
OdbcConnection("Driver={INFORMIX 3.30 32  
BIT};Host=hostname;Server=Servername;Database=bookdb;Service=service-name;Proto  
col=olsoctcp;Database=bookdb;Uid=myusername;Pwd=mypasswd ");
```

6.5.4 Converting ODBC applications

Open Database Connectivity (ODBC) is similar to the CLI standard. Applications based on ODBC can connect to the most popular databases. Thus, the application conversion is relatively easy. You have to perform the conversion of database-specific items in your application, such as the following items:

- ▶ Proprietary SQL query changes
- ▶ Possible changes in calling stored procedures and functions
- ▶ Possible logical changes
- ▶ Testing, rollout, and education tasks

Your current development environment will be the same. For a more detailed description of the necessary steps, refer to 6.6.2, "Migrating applications based on ODBC" on page 318.

6.5.5 Converting Visual C++ applications

Applications that are written to run with Visual C++ use the Microsoft Data Access Components (MDAC) framework to communicate with the SQL Server database. The MDAC framework consists of these components:

- ▶ ActiveX Data Objects (ADO)
- ▶ Open Database Connectivity (ODBC)
- ▶ Object Linking and Embedding Database (OLE DB)

- ▶ Data Access Objects (DAO)
- ▶ Remote Data Objects (RDO)

The IBM Informix OLE DB Provider (CSDK 3.50) is built and tested with Microsoft Data Access Components (MDAC), Version 2.8. To convert the Visual C++ application, you first need to install the IBM Informix OLE DB Provider. IBM Informix OLE DB Provider is distributed with IBM Informix Connect and the IBM Informix Client Software Development Kit (CSDK).

Important: After installation, you must run the `coledbp.sql` script on Informix against the `sysmaster` database as user `informix`. IBM Informix OLE DB Provider requires that the stored procedures are added to the server by the `coledbp.sql` script. The script is located in the `$INFORMIXDIR/etc` directory.

The second step is to configure by running `setnet32` utility and add a data source name (DSN) in the ODBC Data Source Administrator. You can follow the instructions that explain how to use `setnet32` and to add a DSN entry in 6.6.2, “Migrating applications based on ODBC” on page 318.

The third step is to modify the data source names of the connection string in your application. Data source names must be in the following format:

[database] [@server]

The brackets indicate that the enclosed items are optional. If the database name is missing, the client user name is used. If the `@server` name is missing, the default database server, which corresponds to the value that is specified by the client's `INFORMIXSERVER` registry entry, is used.

Table 6-2 on page 297 describes the ADO connection string. You need to specify keywords in the connection string for the IBM Informix OLE DB Provider by using the format `keyword=value`. Delimit multiple keywords with a semicolon. See Table 6-2 on page 297 for the keywords, values, and descriptions.

Table 6-2 ADO keywords that are supported by IBM Informix OLE DB Provider

Keyword	Value	Description
DSN	Name of the database alias	The Informix database alias in the database directory
UID	User ID	The user ID used to connect to the Informix database
PWD	Password	The password for the user ID
CLIENT_LOCALE	Locale	The client locale for the application
DB_LOCALE	Locale	The database locale for the application
UNICODE	True or False	Indicates whether to use IBM Informix Global Language Support (GLS) Unicode. See this website for more information: http://publib.boulder.ibm.com/infocenter/idshelp/v10/topic/com.ibm.oledb.doc/oledb49.htm#ii-using-67137
RSASWS or REPORTSTRINGASWSTRING	True or False	Enables you to control the data mapping for wide strings. See this website for more information: http://publib.boulder.ibm.com/infocenter/idshelp/v10/topic/com.ibm.oledb.doc/oledb50.htm#ii-using-50470
FBS or FETCHBUFFERSIZE	Numeric	The size in bytes of the buffer size that is used to send data to or from the database. The range of values is 4096 (default) - 32767. If you want to set the fetch buffer size at 32 KB, for example, set the connection string as "FBS=32767" or "FETCHBUFFERSIZE=32767". If the value of "FBS" or "FETCHBUFFERSIZE" is not in the range between 4096 - 32767, by default, the value will be changed to 4096 internally and no error message is returned.

Important: The keywords are case sensitive.

The following example shows a sample connection string to work with the Informix database:

DSN example (user must first create the DSN in ODBC Administrator):
 connectString= "Provider=Ifoledbc;Data Source=stores7@ol_11_50;User ID=user_id;Password=user_password"

```
DSN-less example:  
Dsn='';Driver={INFORMIX 3.30 32  
BIT};Host=hostname;Server=myServerAddress;Service=service-name;Protocol=olsoctc  
p;Database=myDataBase;Uid=myUsername;Pwd=myPassword;
```

6.5.6 Converting .NET applications

The following operating systems for developing and deploying .NET Framework 1.1 applications are supported:

- ▶ Windows 2000 Server
- ▶ Windows XP (32-bit edition)
- ▶ Windows Server 2003 (32-bit edition)

The following operating systems for developing and deploying .NET Framework 2.0 applications are supported:

- ▶ Windows 2000 Server, Service Pack 3
- ▶ Windows XP, Service Pack 2 (32-bit and 64-bit editions)
- ▶ Windows Vista (32-bit and 64-bit editions)
- ▶ Windows Server 2003 (32-bit and 64-bit editions)
- ▶ Windows Server 2008

Supported development software for .NET Framework applications

In addition to an Informix client, you need one of the following options to develop .NET Framework applications:

- ▶ Visual Studio 2003 (for .NET Framework 1.1 applications)
- ▶ Visual Studio 2005 (for .NET Framework 2.0 applications)
- ▶ .NET Framework 1.1 Software Development Kit and .NET Framework, Version 1.1, Redistributable Package (for .NET Framework 1.1 applications)
- ▶ .NET Framework 2.0 Software Development Kit and .NET Framework, Version 2.0, Redistributable Package (for .NET Framework 2.0 applications)

In addition to an Informix client, the following two options are needed to deploy .NET Framework applications:

- ▶ .NET Framework, Version 1.1, Redistributable Package (for .NET Framework 1.1 applications)
- ▶ .NET Framework, Version 2.0, Redistributable Package (for .NET Framework 2.0 applications)

.NET data providers

IBM Informix Client SDK, Version 3.50, which is the current version, includes three .NET data providers:

- ▶ Informix .NET Data Provider
This data provider is a high performance, managed ADO.NET Data Provider that you need to use Informix databases. ADO.NET database access using the Informix .NET Data Provider has fewer restrictions and provides significantly better performance than the OLE DB and ODBC .NET bridge providers.
- ▶ OLE DB .NET Data Provider
This bridge provider feeds ADO.NET requests to the Informix OLE DB provider (by way of the COM interop module).
- ▶ ODBC .NET Data Provider
This bridge provider feeds ADO.NET requests to the IBM ODBC driver.

For a better illustration of the available Informix .NET data providers, see Figure 6-18.

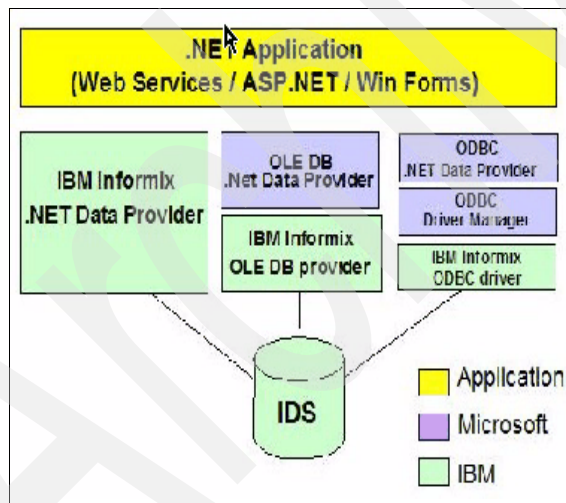


Figure 6-18 Informix .NET in the general .NET Framework

Informix .NET namespace

The IBM.Data.Informix namespace contains the Informix .NET Data Provider. To use the Informix.NET Data Provider, you must add the Imports or using statements for the IBM.Data.Informix namespace to your application .DLL, as shown in Example 6-8 on page 300.

```
[Visual Basic]  
Imports IBM.Data.Informix
```

```
[C#]  
using IBM.Data.Informix;
```

Also, you must add references to IBM.Data.Informix.dll to the project.

If you want to use Informix data types in your .NET application, you can use the IBM.Data.IfxTypes namespace. For a complete mapping between the Informix data types and .NET data types, refer to the *IBM Data Server Provider for .NET Programmer's Guide*, SC23-9848.

Visual Basic .NET conversion example

In general, converting a .NET application from SQL Server to Informix is simple. In most cases, it entails replacing the classes that are available in the SQL Server .NET Data Provider with functionally equivalent classes that are available in the Informix .NET Data Provider, for example, replacing SqlConnection with IfxConnection.

Additionally, information with examples for specific database SQL statement programming techniques is available at these IBM developerWorks web pages:

- ▶ <http://www.ibm.com/developerworks/data/zones/informix/library/techarticle/0309nair/0309nair.html>
- ▶ <http://www.ibm.com/developerworks/data/library/techarticle/dm-0503padmanabhan/>

Important: The database names and tables in SQL Server are case sensitive. Therefore, on Informix side, you need to perform these tasks:

- ▶ Set the DELIMIDENT environment variable to y on the server side.
- ▶ Put all the database object names in double quotation marks (") when migrating database objects. Also, refer to 5.2, "Database schema movement" on page 197.

6.5.7 Converting ASP applications

Active Server Pages (ASP) is a server-side scripting engine. It has been superseded by ASP.NET. The ASP script uses ActiveX Data Objects (ADO) to connect to the database, for example:

```
Session("ConnectionString") = "dsn=SQL_SERVER;uid=<username>;pwd=<  
password>;DATABASE=bookdb;..."
```


Replace the connection string with the equivalent ODBC database source name (DSN) for Informix. For more information about DSN and ODBC, see 6.6.2, “Migrating applications based on ODBC” on page 318.

6.5.8 Converting Java applications

For Java programmers, the IBM Informix Client SDK offers two APIs: JDBC and SQLJ.

JDBC is a mandatory component of the Java programming language, as defined in the Java 2, Standard Edition (J2SE) specification. Enabling JDBC applications for Informix requires an implementation of the various Java classes and interfaces, as defined in the standard. This implementation is known as a *JDBC driver*. Informix provides a fully capable JDBC Type 4 driver for client applications in UNIX and Windows environments.

SQLJ is a standard development model for data access from Java applications. The SQLJ API is defined in the SQL 1999 specification. IBM Informix JDBC supports both JDBC and SQLJ APIs in a single implementation. JDBC and SQLJ can interoperate in the same application. SQLJ provides the unique capability to develop using static SQL statements and control access.

The Java code conversion is rather easy. The API is well defined and database independent. For instance, the database connection logic is encapsulated in standard Java 2 Platform, Enterprise Edition (J2EE) DataSource objects. DS-specific components, such as user name and database name, are then configured declaratively within the application.

However, you need to change your Java source code regarding the following components:

- ▶ The API driver (JDBC or SQLJ)
- ▶ The database connect string

For complete information regarding the Java environment, drivers, programming, and other relevant information, consult *IBM Informix JDBC Driver Programmer's Guide*, SC23-9421.

Java access methods to Informix

Informix has rich support for the Java programming environment. You can access Informix data by putting the Java class into a module in one of the following ways:

- ▶ Informix database server:
 - Stored procedures (JDBC or SQLJ)
 - SQL functions or user-defined functions (JDBC or SQLJ)

- ▶ Browser
 - Applets based on JDBC (JDBC)
- ▶ J2EE Application Servers (such as WebSphere Application Server):
 - Java ServerPages (JSPs) (JDBC)
 - Servlets (SQLJ or JDBC)
 - Enterprise JavaBeans (EJBs) (SQLJ or JDBC)

JDBC drivers for Informix

The IBM Informix Driver for JDBC and SQLJ supports the following situations:

- ▶ All of the methods that are described in the JDBC 3.0 specifications.
- ▶ SQLJ statements that perform equivalent functions to most JDBC methods.
- ▶ Connections that are enabled for connection pooling. WebSphere Application Server or another application server does the connection pooling.
- ▶ Java user-defined functions and stored procedures
- ▶ Global transactions that run under WebSphere Application Server, Version 5.0 and later.
- ▶ Support for distributed transaction management. This support implements the Java 2 Platform, Enterprise Edition (J2EE) Java Transaction Service (JTS) and Java Transaction API (JTA) specifications, which conform to the X/Open standard for distributed transactions.

JDBC driver declaration

To connect from a Java application to SQL Server using the JDBC driver, perform the following steps:

1. Import the SQL Server driver.
2. Register the driver manager.
3. Connect with a user ID, the password, and the database name.

Example 6-9 shows an SQL Server JDBC connection.

Example 6-9 SQL Server JDBC Connection

```
import java.sql.*;
class rsetClient
{
    public static void main (String args []) throws SQLException {
        // Load SQL Server driver
        try
        {
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        }
    }
}
```

```

catch (Exception e)
{
    e.printStackTrace();
}
// Connect to the database
// Connection parameter definition details see below
Connection conn = DriverManager.getConnection(url);
// ...
}
}

```

In a similar way, Example 6-10 demonstrates the registration and connection to Informix with only slight changes in the driver name.

Example 6-10 Informix JDBC connection

```

import java.sql.*;

class rsetClient
{
    public static void main (String args []) throws SQLException {

        // Load Informix JDBC application driver
        try

            Class.forName("com.informix.jdbc.IfxDriver");
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
        // Connect to the database
        // Connection parameter definition details see below
        Connection conn = DriverManager.getConnection(url);
        // ...
    }
}

```

JDBC Type 4 connectivity URL specification

For IBM Informix Driver for JDBC and SQLJ connectivity, the getConnection method must specify a user ID and a password, through parameters or through property values, as illustrated in Example 6-11.

Example 6-11 getConnection syntax for Type 4 connectivity

```

getConnection(String "url; user; password;");

```

The URL definition using the IBM Informix JDBC driver requires specific parameters. The URL heading must be headed with the `jdbc:informix-sqli://` literal. After this literal, specify the machine name of the database server and the port number for the Informix listener thread. Finally, you must specify the database server.

Example 6-12 shows you how to set a complete URL string followed by the user name and password specification in the user and password parameters.

Example 6-12 Setting the user ID and password in the user and password parameters

```
// Set URL for data source
String
connection="jdbc:informix-sqli://machine:1533:informixserver=dbservername;user=
informix;password=passwd;
Connection con = DriverManager.getConnection(url);
```

For more information, see the *IBM Informix JDBC Driver Programmer's Guide*, SC23-9421.

Using single sign-on access control with the Informix JDBC Driver

You can use single sign-on (SSO) access control with JDBC by using the `DriverManager.getConnection()` method and setting the Communication Support Module (CSM) in the connection URL to the service principal. Using SSO access control replaces the user ID and password option.

First, you must ensure that the database server is set up for SSO authentication. For more information, refer to the “Single Sign-on Access Control” section in the *IBM Informix Security Guide*, SC23-7754.

Follow these steps to set up SSO access control with Informix JDBC Driver:

1. Modify the connection URL so that it includes the service principal. The service principal consists of the database server name and the SSO realm:

```
CSM=(SSO=database_server@realm,ENC=true)
```

The `ENC=true` setting means that Generic Security Services (GSS) encryption is enabled. The `ENC=true` setting is optional, because, by default, its value is true. If you do not want to enable GSS encryption, set the value to false: `ENC=false`.

For the complete syntax of the connection URL, refer to “JDBC Type 4 connectivity URL specification” on page 303.

2. Create a login configuration file with the following code:

```
com.sun.security.jgss.initiate {
    com.sun.security.auth.module.Krb5LoginModule required
    useTicketCache=true
    doNotPrompt=true;
}
```

3. Run the application with the `java.security.auth.login.config` property set to the login configuration file's full path name, followed by the `TestSso` class. The following example shows that `lfmxLog.conf` is the login configuration file:

```
java -Djava.security.auth.login.config=mydirectory/lfmxLog.conf TestSso
```

Stored procedure calls

The handling of input and output parameters in stored procedure calls differs between SQL Server and Informix. The following examples explain the types of procedure calls and the usage of parameters and result sets.

Stored procedure with an input parameter

Assume that a stored procedure has been created in SQL Server, as shown in the following code:

```
create procedure sp_testcall_1
    @parm1 int,
    @parm2 int
```

This code shows the creation of a procedure in Informix:

```
CREATE PROCEDURE sp_testcall_1( parm1 INTEGER , parm2 INTEGER )
```

The procedures have two input parameters and no output parameters. There is no difference in the call between SQL Server and Informix. In both cases, the parameter values must be set before the stored procedure can be executed. Example 6-13 demonstrates this point.

Example 6-13 Java call of SQL Server or Informix procedure with input parameter

```
// Connect to the database
// Connection parameter definition details see below
Connection conn = DriverManager.getConnection(url);
// ...
//
CallableStatement stmt;
String SP_CALL = "{call sp_testcall_1(?,?)}";

stmt = conn.prepareCall(SP_CALL);
stmt.setInt(1,10);
stmt.setInt(2,15);
stmt.execute();
```

```
//  
...
```

Stored procedure with a result set

The next example shows a procedure without an input parameter, but the procedure defines a result set as an output parameter. The result set is an opened cursor defined in the procedure. The rows are fetched in the Java application with a loop.

The SQL Server stored procedure is defined, as shown in the following code:

```
create procedure sp_testcall_3  
    @lname char(30) CURSOR VARYING OUTPUT  
AS  
SET NOCOUNT ON;  
    SET @cursor1 = CURSOR  
    FORWARD_ONLY STATIC FOR  
        SELECT last_name  
        FROM dbo.employee;  
    OPEN @cursor1;  
GO
```

Example 6-14 shows the corresponding Informix procedure code.

Example 6-14 Stored procedure definition in Informix

```
create procedure sp_testcall_3 ( ) returning char(30);  
define lname char(30);  
  
FOREACH cursor1 FOR  
    Select last_name into lname from employee  
    RETURN lname WITH RESUME;  
    END FOREACH  
END procedure ;
```

With IBM Informix JDBC, you do not need to register the result set with the method `registerOutParameter()` in the Java application. To get the result set, call the method `getResultSet()`, as shown in Example 6-15.

Example 6-15 Java call of Informix procedure with result set

```
String SP_CALL = "{call sp_spcall_v3}";  
  
// Connect to the database  
Connection conn = DriverManager.getConnection (url);  
  
try {  
    CallableStatement stmt = conn.prepareCall(SP_CALL);
```

```

        ResultSet rs = null;
        stmt.executeQuery();
        rs = stmt.getResultSet();
        while(rs.next())
        {
            // ...
        }
    }
}

```

Stored procedure with an input parameter and result set

In IBM Informix JDBC, you define input parameters and result sets, as shown in Example 6-16. Input parameters are numbered, beginning with 1, and are independent from the retrieval of result sets.

Example 6-16 Java call to Informix procedure with input parameter and result set

```

String SP_CALL = "{call sp_spcall_v4(?)}";

Connect to the database
    Connection conn = DriverManager.getConnection (url);

try {
    CallableStatement stmt = conn.prepareCall(SP_CALL);
    stmt.setInt(1, emp_id);
    ResultSet rs = null;
    stmt.executeQuery();
    rs = stmt.getResultSet();
    while(rs.next())
    {
        System.out.println (rs.getString (1));
        // ...
    }
}

```

Stored procedure converted from a function

The Informix function looks similar to the appropriate procedure definition. As in the case of the definition of a procedure, the cursor is defined in the function. The current row values are returned and the cursor remains open.

The next call of the function in the same result set retrieves the next row, as shown in the following example:

```

create function sp_testcall_3 ( cust_no integer) returning char(30);
define lname char(30);
FOREACH cursor1 FOR
    Select last_name into lname from customer where
        customer_num=cust_no;

```

```
        RETURN lname WITH RESUME;  
    END FOREACH  
END function ;
```

The Informix stored procedure works similarly to the SQL Server function. The returning clause is similar to the SQL Server returns clause, which allows multiple return values.

Informix also provides FOREACH, the procedural equivalent of using a cursor. To execute a FOREACH statement, the database server takes these actions:

1. It declares and implicitly opens a cursor.
2. It obtains the first row from the query contained within the FOREACH loop or else the first set of values from the called routine.
3. It assigns to each variable in the variable list the value of the corresponding value from the active set that the SELECT statement or the called routine creates.
4. It executes the statement block.
5. It fetches the next row from the SELECT statement or called routine on each iteration, and it repeats steps 3 and 4.
6. It terminates the loop when it finds no more rows that satisfy the SELECT statement or called routine. It closes the implicit cursor when the loop terminates.

6.5.9 Converting PHP applications

PHP (Hypertext Preprocessor) is a commonly used scripting-based language in web server environments. It contains, in most environments, a core system that is represented as a library, which is attached to the web server processes. In case PHP is invoked in the HTML of the current site of the browser, the library is invoked for parsing the script and executing the statements. To access the database server, you either must use the existing language interfaces that are provided by the core PHP system or download additional libraries for your preferred programming environment and attach them to the core system. The language interfaces are available in a procedural or object-oriented style.

In this section, we describe the areas to consider for the migration of an existing application. If you are interested in a much more detailed discussion of PHP application development, in view of the discussion about the database interfaces that are available for developing application clients with Informix, refer to *Developing PHP Applications for IBM Data Servers*, SG24-7218.

PHP setup considerations

Before using PHP in your client applications, ensure that PHP is available on the client application side and that you have identified the target PHP database interface according to your needs or according to the currently used SQL Server interface.

Download and setup directions

There are two options to install PHP. You can install a PHP package that is provided with the distribution of your operating system. All Linux systems have a core PHP system that is included in the distribution. Or, you can download the source of the current version from the following web page:

<http://www.php.net/downloads.php>

After you download the source, you have to build the package and install the package. You need to plug the package into the existing web server and attach the appropriate file type mapping to the PHP system. The configuration file that must be changed depends on the web server that you use. For Apache, change the `httpd.conf` file. If you need additional interfaces, such as Informix PDO, that are not in the currently installed distribution, you can obtain the source from the following web page:

<http://pecl.php.net>

The Informix PDO database programming interface is located on the following web page:

http://pecl.php.net/package/PDO_INFORMIX

You also must change the `php.ini` file to introduce new interfaces into the PHP core. Consider this file as a type of registry.

Existing PHP database interfaces for Informix and SQL Server

These PHP database interfaces support the access to SQL Server:

- ▶ SQL Server PDO

PDO_DBLIB is a driver that implements the Informix PHP Data Objects (PDO) interface to enable access from PHP to SQL Server through the FreeTDS library.

- ▶ ODBC PDO

The SQL Server Driver for PHP relies on the SQL Server ODBC Driver to handle the communication with SQL Server. SQL Server Driver for PHP is only supported on Windows.

There are several PHP database interfaces that support the access to Informix:

- ▶ **PDO_IBM**

This extension functions as a database extraction layer to enable access from PHP to Informix. PDO_IBM is a PECL extension. PECL is a repository for PHP extensions that provides all known extensions and hosting facilities for downloading and developing PHP extensions.

- ▶ **Informix (ifx)**

This extension is based on a native connection. The interface requires, at build time, the installation of the IBM Informix Client Software Development Kit (CSDK) and binds the communication libraries that are needed for the message flow between the server and the client. The interface is available at PHP, Version 3 and later. It supports all major functionality that is provided by the server and is procedurally oriented.

- ▶ **unixODBC**

This extension uses an external ODBC manager. Additionally, an ODBC connection library for the access to the database is required. This library is provided by Informix SQK or Connect. The interface supports a procedural interface and provides functionality that is similar to the ODBC interface.

- ▶ **Informix PDO**

This interface is a new object-oriented interface that requires at least the core PHP 5 version. It combines the benefits of using the object orientation with the function set of the Informix database server. It uses native connections, so it provides fast communications performance.

- ▶ **ODBC PDO**

This extension is a combination of using a ODBC manager and providing an ODBC-based programming interface with an object-oriented programming style.

Migrating existing PHP programs also includes a target interface decision. Most of the time, your existing PHP scripts develop over time and are based on procedural interfaces. The current programming style tends to use object orientation, which also requires, in addition to changing the database interface, a change of the complete script logic. Include an inventory of which interface and style is used and which style to use for the target style for your new needs at the beginning of the migration.

Sample PHP application environment migration

To demonstrate the differences between PHP programming in SQL Server and Informix, we show a sample program that demonstrates the following information:

- ▶ Connecting to a database through PHP
- ▶ Calling a stored procedure with an input parameter
- ▶ Returning an output parameter

We use the SQL Server-based stored procedure, as shown in Example 6-17.

Example 6-17 SQL Server stored procedure Greeting

```
create procedure Greeting
    @name varchar(18),
    @message varchar(50) OUTPUT
AS
    SELECT @message = GETDATE();
    SET @message = "Hello " + @name + ", the time now is " + @message + "!";
GO
```

Connecting to SQL Server using PHP

Use the `sqlsrv_connect` function to establish a connection to SQL Server. The following example establishes a connection to SQL Server:

```
$serverName = "(local)";
$connectionOptions = array("Database"=>"bookdb");

/* Connect using Windows Authentication. */
$conn = sqlsrv_connect( $serverName, $connectionOptions);
```

Connecting PHP applications to Informix

The PHP database interfaces that support connections to Informix can differ in their requirements for the specification of the connection parameters. Therefore, we provide several examples so you can decide which type of connection you prefer in your production environment. Remember that there is, for specific interfaces, a difference between trusted and untrusted connections, which we have seen previously in the discussion about the embedded interface.

Example 6-18 shows the details.

Example 6-18 Connection strings using separate PHP interfaces

```
#Informix procedural PHP interface
/* standard untrusted */
$link = ifx_connect("sysmaster@srv","informix","123456");
/* persistent untrusted*/
$link = ifx_pconnect("sysmaster@srv","informix","123456");
/* standard trusted */
```

```

$link = ifx_connect("sysmaster@srv");
/* persistent trusted */
$link=ifx_pconnect("sysmaster@srv");

/* Informix PDO */
/* standard connect */
$dbh = new PDO("informix:; database=sysmaster; server=srv;",
"informix", "123456");
/* standard connect trusted user */
$dbh = new PDO("informix:; database=sysmaster; server=srv;");
/* persistent connect untrusted user */
$dbh = new PDO("informix:; database=sysmaster; server=srv;"
,"informix","123456",array(PDO::ATTR_PERSISTENT=> true));
/* persistent connect trusted user */
230 Developing PHP Applications for IBM Data Servers
$dbh = new PDO("informix:; database=sysmaster; server=srv;"
,NULL,NULL,array(PDO::ATTR_PERSISTENT=> true));

/* unixODBC */
/*trusted*/
$cn = odbc_connect
("Driver=Informix;Server=srv;Database=sysmaster",NULL,NULL);
/* not trusted */
$cn = odbc_connect
("Driver=Informix;Server=srv;Database=sysmaster","informix","123456");
/* persistent not trusted*/
$cn = odbc_pconnect
("Driver=Informix;Server=srv;Database=sysmaster","informix","123456");
/* persistent trusted*/
$cn = odbc_pconnect
("Driver=Informix;Server=srv;Database=sysmaster",NULL,NULL);

```

As you can see, PHP distinguishes between persistent and non-persistent connections. The connection can open after PHP is finished, because the client of the database server is not the PHP script. It is the web server process that attaches the PHP library. You are able to specify the user and password for untrusted connections, but you also can set up trusted connections, which is inadvisable in a production environment in terms of security.

sqlsrv_connect and ifx_connect

The following syntax shows sqlsrv_connect preparing and executing a statement in SQL Server:

```
(resource $conn, string $tsql [, array $params [, array $options]])
```

If we use a mapping into a procedural informix PHP interface, we must convert `sqlsrv_connect` to the Informix function `ifx_connect`, which takes the following required and optional parameters (in brackets []):

(string database, [string username], [string password], [array options])

Handling statements in PHP applications

After a successful connection, you must look at the handling of the statements in your existing application and how to move the implementation to the target PHP interface. We want to have a closer look at how to execute the statements and how to bind variables for the execution.

Using parameters for preparing statements

At the time of the statement definition, there are two types of statements. Either you have a complete statement, or you want to prepare the statement for the execution with a later binding of additional values, which change in every loop. SQL Server uses `sqlsrv_prepare` to create a resource that is associated with the specified connection. In Informix, we use Informix PDO for preparing the call of the SP with one placeholder for later binding, as shown in Example 6-19.

Example 6-19 Preparing the statement for execution

```
/* Informix PDO example code piece */
$sph= $dbh->prepare(" EXECUTE PROCEDURE greetings ( ?);");
$name = "Informix";
$sph->bindParam(1, $name,PDO::PARAM_STR, 20);
$stmt->execute();
$name = "SQL Server";
$stmt->execute();
```

There is another way of implementing the logic of executing this procedure. You can generate the complete string for the procedure call into a string. After that, you can apply the string for an immediate execution. Example 6-20 shows the sample code.

Example 6-20 Build up the statement in a string and execute it without a parameter

```
$statement="EXECUTE PROCEDURE greetings ( " . $name . " );";
$dbh->exec($statement);
```

Parameter binding

Using the question mark character (?) placeholder during statement preparation requires a parameter bind before the statement can be executed. You can use the `bindParam` method of the statement handle. You need to specify the position, the name of the variable, and the type. Depending the type, an additional length

specification is required. Example 6-21 shows several possible parameter bindings. Be aware that the binding must be done before the execution of the query is issued.

Example 6-21 Specification of parameter to an already prepared statement

```
$sph->bindParam(1, $_POST["parm1"],PDO::PARAM_STR, 20);
$sph->bindParam(2, $_POST["parm2"],PDO::PARAM_STR, 20);
$sph->bindParam(3, $_POST["parm3"],PDO::PARAM_INT);
$sph->bindParam(4, $_POST["parm4"],PDO::PARAM_INT);
$sph->bindParam(5, $DEBUG,PDO::PARAM_INT);
```

Retrieving the result set

After we execute the stored procedure, we must verify the result. Unlike the OUT parameter in SQL Server, the output is returned in the result set Informix PDO and has to proceed further. In our simple example, we only print the message that is returned by the stored procedure. For the final code that handles the result, refer to Example 6-22.

Example 6-22 Retrieving the result returned by the stored procedure in Informix

```
$sph= $dbh->prepare(" EXECUTE PROCEDURE greetings (?);");
$name = "Informix";
$sph->bindParam(1, $name,PDO::PARAM_STR, 20);

$sph->execute();
$error=$dbh->errorInfo();
if ( $error["1"]) print_r ($error);
$row=$sph->fetch(PDO::FETCH_NUM);
if ($row)
print_r ( $row) ;

$sph->execute();
$error=$dbh->errorInfo();
if ( $error["1"]) print_r ($error);
/* fetch as an assoc array */
$row=$sph->fetch(PDO::FETCH_ASSOC);
print_r ( $row) ;
```

Cursors and array specification for the result set

In the previous example, only the Informix PDO code showed the handling of a cursor in PHP. Now, we introduce an example that is based on a simple select query that returns several values in SQL Server.

Example 6-23 Handling a result set in SQL Server

```
<?php
$serverName = "(local)";
$connectionOptions = array("Database"=>"bookdb");

/* Connect using Windows Authentication. */
$conn = sqlsrv_connect( $serverName, $connectionOptions)
$sql = "SELECT DEPTNUMB,DEPTNAME FROM org_table";
$stmt = sqlsrv_query( $conn, $sql);
/* Retrieve each row as an associative array and display the results.*/
while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_ASSOC))
{
    ...
}

/* Free statement and connection resources. */
sqlsrv_free_stmt( $stmt);
sqlsrv_close( $conn);
?>
```

Example 6-24 shows the implementation of a similar query in Informix PDO. You can generate the result set in various representations using assoc arrays, enumerated arrays, objects, and so on. The style of the result set is specified with the input parameter used in the fetch method for the statement object.

Example 6-24 Handling the cursor and the various representations of the result

```
<?php
$dbh = new PDO("informix:host=lech; service=1000; database=vps;
server=svr; protocol=onsoctcp ", "informix", "123456");
if (!$dbh) { exit(); }
$stmt=$dbh->query('SELECT DEPTNUMB,DEPTNAME FROM org_table');

/* assoc Array */
$row=$stmt->fetch(PDO::FETCH_ASSOC);
if ($row)
printf(" %s %s \n",$row["DEPTNUMB"],$row["DEPTNAME"]);

/* a numerated Array */
$row=$stmt->fetch(PDO::FETCH_NUM);
If ($row)
printf(" %s %s \n",$row[0],$row[1]);

/* into an object */
$row=$stmt->fetch(PDO::FETCH_OBJ);
if ( $row)
printf(" %s %s \n",$row->DEPTNUMB,$row->DEPTNAME);
```

```

/* hybrid array structure enum and assoc */
$row=$stmt->fetch(PDO::FETCH_BOTH);
if ($row)
printf(" %s %s \n",$row["DEPTNUMB"],$row[1]);

/* hybrid array structure/object enum/assoc/object */
$row=$stmt->fetch(PDO::FETCH_LAZY);
if ( $row)
printf(" %s %s \n",$row["DEPTNUMB"],$row->DEPTNAME);

$deptnumb=0;
$deptname="";
$stmt->bindColumn(1, $deptnumb, PDO::PARAM_INT);
$stmt->bindColumn(2, $t_model, PDO::PARAM_STR,50);
/*bound output variable */
$row=$stmt->fetch(PDO::FETCH_BOUND);
printf(" %s %s \n",$deptnumb,$deptname);
exit;
?>

```

You can handle all cursor-based statements in the same way. This method is also valid for procedures that return values. For SQL statements that do not return a result set, such as insert, delete, and update statements and all DDL statements, you only need to apply exception handling and error code checking.

Handling errors

In SQL Server, PDO provides the `sqlsrv_error` function for determining the status of the last executed database statement. The required parameters differ depending on the context where you use `sqlsrv_error`. You use either the connection handle or the statement handle, or you do not use a parameter if the connection was not established. The function returns the current error status, which can proceed further, as shown in Example 6-25.

Example 6-25 Error handling in SQL Server

```

$conn = sqlsrv_connect( $serverName, $connectionOptions);
if( $conn === false )
{ echo "Connection not successful\n";
  die( print_r ( sqlsrv_errors() ) ); }

$stmt = sqlsrv_query ( $conn, $sql, $params);
if ($stmt === false) {
    echo "Error in execution.\n";
    die ( print_r (sqlsrv_errors()));
}

```

Using PDO, you can catch exceptions that occurred in a specific defined code area. In addition to the exception, you can get the SQL error that occurred in the processing of the current object. Look at Example 6-26 for the difference between using an exception and checking an error status.

Example 6-26 Exceptions and status determination in Informix PDO

```
/* Catch an exeption with Informix PDO */
try
{
$dbh->beginTransaction();
}
catch (PDOException $e )
{
printf("Error: %s \n",$e->getMessage());
}
```

Error: There is already an active transaction.

```
/* Determine an error with Informix PDO */
$stmt=$dbh->query('SELECT * FROM nonexistingtable');
$error=$dbh->errorInfo();
print_r($error);

Array
(
[0] => 42S02
[1] => -206
[2] => [Informix][Informix ODBC Driver][Informix]The specified table
(nonexistingtable) is not in the database. (SQLPrepare[-206] )
)
```

6.6 Migrating third-party applications

In this section, we discuss how to set up a migration plan for database applications that were bought from an external source with limited or no access to the implementation logic. Using an ODBC application on Windows that is implemented with SQL and ODBC standards, we show you how to change the database connectivity module to enable the client application to access the target database server.

Finally, we provide tips about how to apply a new database connection to a JDBC-based application.

6.6.1 Planning packaged application migration

For third-party package applications, the vendor delivers the application and ensures the database connectivity to Informix. In comparison with an internally developed, source code owned application, the migration is limited to the following major tasks:

- ▶ Checking software and hardware availability and compatibility
- ▶ Educating developers and administrators
- ▶ Analyzing customized changes in the application and database
- ▶ Setting up the target environment
- ▶ Changing customized items
- ▶ Testing data migration and the customized items
- ▶ Rolling out the migrated application

To keep the support from the vendor, you have to meet the vendor's prescribed migration plan and process.

6.6.2 Migrating applications based on ODBC

Current third-party database application packages use one of two common implementation techniques for accessing the database server. Either they rely on the SQL standards and try to implement all the SQL statements by following this standard, or they provide a database-specific connection library that is needed to attach to the application (such as a cartridge).

In any migration project where you want to move the application from one database server to another database server, you must determine which implementation of the database connectivity is provided. In both cases, you have to contact the software manufacturer for this particular solution to make sure that the target database server is supported and discover which version of the target database server is supported. In case a connection library is needed, you typically obtain the connection library from the application vendor.

If the application is implemented with a standard SQL interface, such as SQL-92 or later standards, and it is not based on Java, it likely uses the ODBC interface for the connection. The ODBC standard enables the database provider to implement the database connectivity only one time and to certify a specific database for the product. You can use the ODBC standard on both Windows and UNIX.

If you want to move an application that is based on ODBC, you need to apply a new ODBC driver. This type of database application can be based either on a pure ODBC connection library, a .NET Windows-based application using the ODBC .NET Data Provider, or a PHP application using the ext/ODBC or

PDO_ODBC database interface. You can install a new IBM Informix ODBC driver by installing either the IBM Informix Client SDK or the IBM Informix Connect product. You need the data source name (DSN). After the installation, there is an easy to use server connection specification and ODBC driver registration. When you have enabled the new IBM Informix ODBC DSN, change the connection database server in the application by the specification of the new DSN. After changing the DSN, you can start testing the integration of the application in the new database environment.

You can set up the new ODBC driver in two steps:

1. Set up the communication parameter for Informix with the setnet32 utility.
2. Create a new ODBC DSN in the ODBC manager on Windows.

Figure 6-19 shows the setup window for the setnet32 utility.

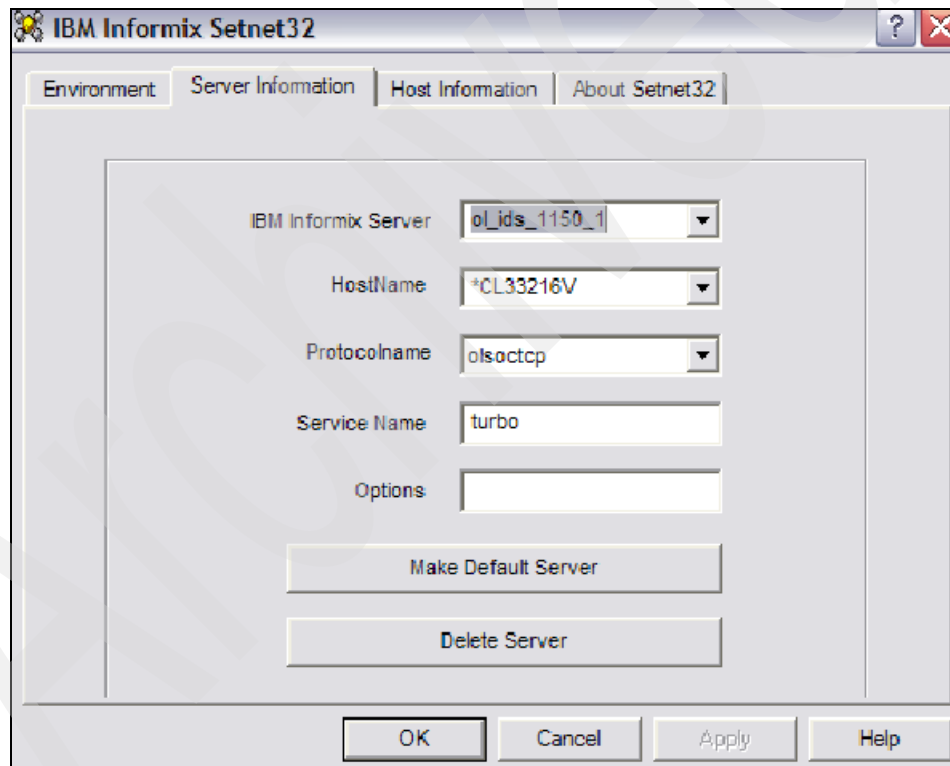


Figure 6-19 Using setnet32 to specify the remote Informix database server

After specifying the server parameter, you add a new DSN to the ODBC administrator in Windows.

Figure 6-20 shows the appropriate window for setting up an Informix data source name on Windows.

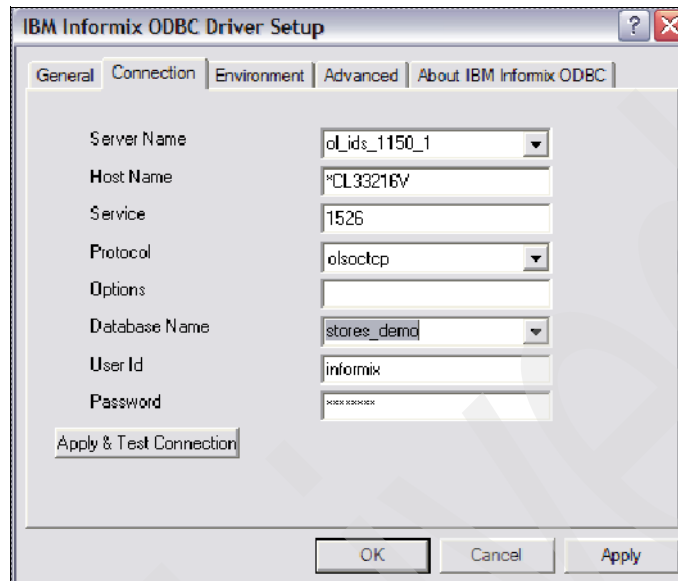


Figure 6-20 Setting up an Informix DSN with the ODBC manager on Windows

6.6.3 Migrating database applications based on JDBC

Java and Eclipse-based applications commonly involve a JDBC driver for the maintenance of the database access. In general, the JDBC driver represents a jar file that needs to be included in the CLASSPATH of the client application environment. Then, the specification of a URL for the target database server and a user ID and password for the user issuing the connection are required.

In summary, you must perform the following steps to add a new JDBC connection to your application:

1. Install the IBM Informix JDBC driver on the client application machine.
2. Depending on the application, select one of the following actions:
 - Add the driver location to the CLASSPATH. The name of the jar file is `ifxjdbc.jar`.
 - Select the Informix driver. Certain applications include JDBC drivers for a set of database providers so that you can select Informix from that set.
 - Register the new driver in the application, and specify the driver location.

3. Define a new database source in the application, and specify the necessary connection parameter for the URL, along with the user ID and the password.
4. Test the connection, and start application testing with that connection.

Figure 6-21 shows an example of a configuration window for the specification of JDBC-based database access to Informix. We created this window by using Rational Data Architect. The configuration for your application will be similar.

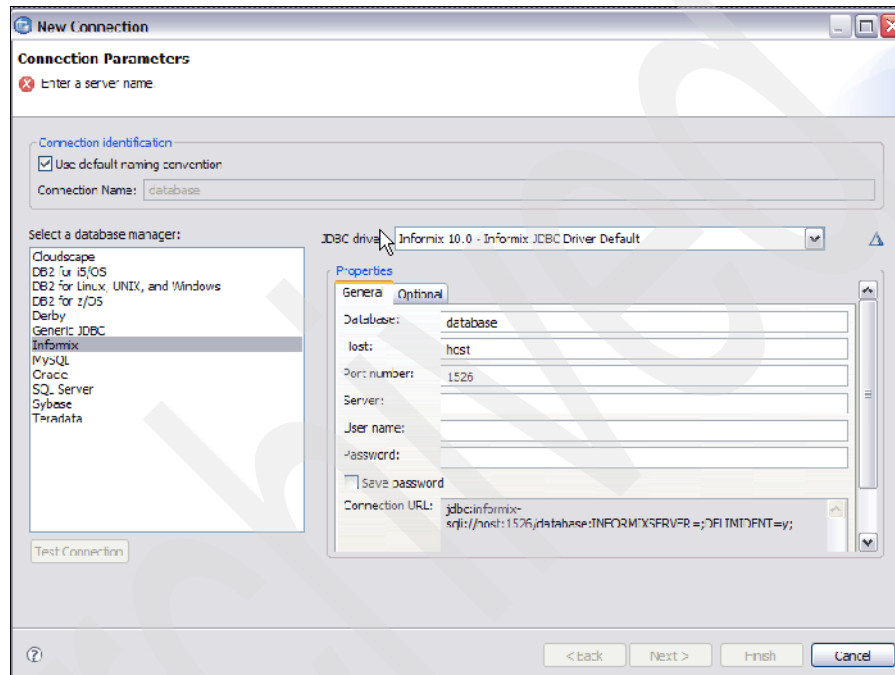


Figure 6-21 Configure a new database connection that is based on a JDBC driver

Informix configuration and administration

The administration cost of any database server is a major consideration for CIO and IT managers. The number of database administrators (DBAs) required to administer the enterprise databases can play a factor in budgeting and business continuity. IBM Informix ease of use, high uptime, and low maintenance allow a single DBA to manage hundreds of Informix instances, in effect, reducing maintenance cost. IBM Informix provides you a way to cut highly specialized, but required, database administration costs while still keeping services and systems running at optimal capacity. Informix maintains a high performance while continuing to encompass competitive features, such as compression, external tables, a secondary server that can be updated, improved SQL tracing, and improved security options.

In this chapter, we introduce Informix features by discussing the following topics:

- ▶ Installation, setup, and initialization
- ▶ Administration and monitoring tools
- ▶ High availability and disaster recovery
- ▶ Data replication, enabling business continuity

7.1 Installation considerations

The ease, speed, and small footprint of the Informix database server installation continues to impress DBAs that have worked on other relational database management system (RDBMS) servers. In this section, we point out installation considerations for the Informix database server. For full installation instructions and details about each of these considerations, see the *IBM Informix Dynamic Server Installation Guide for UNIX, Linux, and Mac OS X*, GC23-7752, and the *IBM Informix Dynamic Server Installation Guide for Windows*, GC23-7753.

Important: You must obtain root privileges before performing many of the installation-related tasks.

The following list is intended for new installations. Review this list thoroughly in preparation for installation:

- ▶ Operating system readiness
- ▶ Creation of the *informix* user and *informix* group
- ▶ Disk space requirements
- ▶ Installation methods (typical, custom, and silent install)
- ▶ Product installation directory (\$INFORMIXDIR) and its permissions
- ▶ Role separation planning (You *cannot* turn off role separation after you enable it.)
- ▶ Installable features of Informix
- ▶ Local versus Domain Windows installation
- ▶ Multiple residency considerations: Multiple independent database server environments on the same computer

Important: You can install IBM Informix products by using RPM Package Manager (RPM) on certain Linux operating systems.

An optional selection during installation is to select whether to create a demonstration database server instance. If you choose to create a demonstration database server in the installation application, Informix will be initialized automatically after the installation is complete.

If you do not choose the demonstration option, you can configure and initialize the database server manually after installation is complete, which we discuss in the next section.

7.2 Database server setup and initialization

Upon completion of the installation, you can configure Informix for initialization where operating system (OS) resources are allocated to the database server based on the specified configuration parameters values used during initialization. You can tune and modify the initial configuration at any time in the future, as needed. In this section, we discuss the essential configuration and environment settings that are required to initialize the database server, create user databases, and enable client connections.

7.2.1 Configuring the database server

Configuration refers to setting specific parameters that customize the database server for your data processing environment, such as the volume of data, number of tables, types of data, hardware, number of users, and security needs.

You can customize the database server so that it functions optimally in your particular data processing environment. The configuration settings to use depends largely on your environment. Decision Support System (DSS) configuration settings differ from Online Transaction Processing System (OLTP) configuration settings.

Informix has only one configuration file, which is called the `onconfig` file, for the instance. The configuration parameters are stored in the `onconfig` file. The product installation script sets the default values for most of the configuration parameters. The `onconfig.std` template file in the `etc` subdirectory of `informixdir` contains the initial values for many of the configuration parameters. Make a copy of this template, and tailor the copy for your specific configuration.

You can use a text editor to change the configuration parameters in the `onconfig` file.

You can configure the minimum parameters (See Table 7-1 on page 326) to initialize and bring an Informix instance online with default settings.

Table 7-1 Minimum required configuration parameters

Onconfig parameter	Description	Comments
ROOTPATH	The ROOTPATH is the full path to a file or a link to a device or a device name (for example, /home/links/rootchunk).	The file or device needs 660 permissions and informix:informix for ownership and group.
SERVERNUM	The servernum specifies a relative location in shared memory. The value that you choose must be unique for each database server on your local computer.	This value ranges from 0 - 255.
DBSERVERNAME	The dbservername is a unique name that is associated with a specific instance of the database server.	Client applications use dbservername in the INFORMIXSERVER environment variable and in SQL statements to establish a connection to a database server.

For information about the configuration parameters and how to monitor them, see *IBM Informix Dynamic Server Administrator's Reference*, SC23-7749.

7.2.2 Set environment variables

To start, stop, or access a database server, each user must hold the required database access privileges and must set the appropriate environment variables. Certain environment variables are required, and other environment variables are optional.

Tip: Set the environment variables in the appropriate startup file for your shell file or Windows.

You can include a reference to the \$INFORMIXDIR environment variable in the onconfig file, and it needs to be the first path name value in the path name specifications.

Table 7-2 on page 327 shows the environment variables that you must set before you access the database server or perform most administration tasks.

Table 7-2 Required environment variables

Environment variable	Description
INFORMIXSERVER	This environment variable specifies the name of the default database server. It has the value specified for the DBSERVERNAME or DBSERVERALIASES configuration parameter.
INFORMIXDIR	This environment variable specifies the directory where you installed your IBM Informix database server.
ONCONFIG	This environment variable specifies the name of the active onconfig file. All users who use database server utilities, such as onstat, must set the ONCONFIG environment variable to the active onconfig file name, for example: ▶ On UNIX: \$INFORMIXDIR/etc/onconfig.bookdb ▶ On Windows: %INFORMIXDIR%\etc\onconfig.bookdb
JVPHOME	If using J/Foundation, this environment variable specifies the directory where you installed the IBM Informix JDBC Driver.
CLASSPATH	If using J/Foundation, this environment variable specifies the location of the jvphome/krakatoa.jar file so that the Java Development Kit (JDK) can compile the Java source files.
PATH	This environment variable specifies the location of executable files: ▶ On UNIX: \$INFORMIXDIR/bin ▶ On Windows: %INFORMIXDIR%\bin

7.2.3 Configure connectivity

When an application or a utility needs to connect to the database server, it has to know the name of the database server, the host machine on which it resides, the communication protocol to use, and the port number on which the database server is listening. In this section, we describe how to configure needed connectivity elements for IBM Informix.

The sqlhosts information

The sqlhosts information, in the sqlhosts file on UNIX or the SQLHOSTS registry key on Windows, contains connectivity information for each database server. The sqlhosts information also contains definitions for groups. The database server looks up the connectivity information when the database server is started, when a client application connects to a database server, or when a database server connects to another database server.

The connectivity information for each database server includes four fields of required information and one optional field. The group information contains information in only three of its fields. See Table 7-3 for a sample `sqlhosts` file's entries.

Table 7-3 Sample `sqlhosts` file

Observer name	Nettype	Host name	Service name or port	Options
sales	onipcshm	machine01.ibm.com	salessshm	N/A
payroll	onsoctcp	machine02	payroll_tcp	s=2
asia	group	N/A	N/A	e=asia.g
asia.1	ontlitcp	10.21.150.19	9088	g=asia
asia.2	onsoctcp	machine03	asia2srv	g=asia

On Windows, the `HKEY_LOCAL_MACHINE` registry contains the `sqlhosts` information. The database server installation procedure prepares the registry information. Do *not* edit the `HKEY_LOCAL_MACHINE` registry.

On UNIX, the `sqlhosts` file contains connectivity information. The default location of this file is `$INFORMIXDIR/etc/sqlhosts`. If you store the information in another location, you must set the `INFORMIXSQLHOSTS` environment variable to point to that location.

Table 7-4 on page 329 lists the five fields of connectivity information from one line in the UNIX `sqlhosts` file in the order that they appear in the file. The `sqlhosts.std` template file is provided in the `etc` subdirectory of `INFORMIXDIR`.

Table 7-4 Fields of sqlhosts file

UNIX field name	Windows field name	Description of connectivity information	Description of group information
dbservername	Database server name key or database server group key.	Database server name.	Database server group name.
nettype	PROTOCOL	Connection type.	The word group.
hostname	HOST	Host computer for the database server.	No information. Use a hyphen as a placeholder in this field.
servicename	SERVICE	Alias for the port number.	No information. Use a hyphen as a placeholder in this field.
options	OPTIONS	Options that describe or limit the connection.	Group options.

To manage the `sqlhosts` entries, you can use a text editor on UNIX and the `setnet32` utility on Windows. For full details about the `sqlhosts` file and an expanded discussion of its entries, see *IBM Informix Dynamic Server Administrator's Guide, version 11.10*, G229-6359.

DRDA communication protocol

When you install IBM Informix Version 11.50, the installer enables you to configure a database server alias and a port for clients that use the Distributed Relational Database Architecture (DRDA) protocol. DRDA is for open development of applications that allow access of distributed data. DRDA is interoperable with IBM Data Server clients. If you do not select the DRDA support option, you can still set up the instance to function with the DRDA protocol after installation.

To configure Informix to connect to an IBM Data Server Client, set up a new server alias in the `sqlhosts` file or Windows registry, using either the `drtlitcp` or `drsoctcp` connection protocol. Specify information in one of the following formats:

- ▶ `server_name drtlitcp machine_name service name/portnumber`
- ▶ `server_name drsoctcp machine_name service name/portnumber`

7.2.4 Starting and administering the database server

After you install and configure the database server, you need to perform one or more of the following tasks:

- ▶ Start the database server and initialize disk space.
- ▶ Prepare to connect to applications.
- ▶ Create storage spaces.
- ▶ Set up your backup and restore system.
- ▶ Perform administration tasks.

Starting the database server and initializing disk space

Initialization of the database server refers to two related activities:

- ▶ Shared memory initialization
- ▶ Disk space initialization

Shared memory initialization

Shared memory initialization allocates operating system memory segments to the server. Bringing up or starting the server establishes the contents of database server shared memory in the following ways: internal tables, buffers, and the shared-memory communication area. Shared memory is initialized every time that the database server starts up. You use the `oninit` utility from the command line to initialize database server shared memory and to bring the database server online. Shared-memory initialization also occurs when you restart the database server.

One key difference distinguishes shared memory initialization from disk space initialization. Shared memory initialization has no effect on disk space allocation or layout, and no data is destroyed. Perform these tasks to bring the database server online:

- ▶ On UNIX

To bring the database server to online mode, enter **`oninit`**.

- ▶ On Windows

On Windows, the database server runs as a service. Use the Service control application to bring the database server to online mode.

Another way to initialize the database server on Windows is to use the following command, where *dbservername* is the database server name:

`starts dbservername`

Disk space initialization

Disk space initialization uses the values that are stored in the configuration file to create the initial chunk of the root dbspace on disk. When you initialize disk space, the database server automatically initializes shared memory as part of the process. Disk space is usually initialized only one time the first time that the database server starts. It is only initialized thereafter during a cold restore or at the request of the DBA.

Important: When you initialize disk space, you overwrite what is on that disk space. If you re-initialize disk space for an existing database server, all the data in the earlier database server becomes inaccessible and is destroyed.

The database server must be in offline mode when you begin initialization. If you are starting a database server for the first time, or you want to remove all dbspaces and their associated data, use the methods in Table 7-5 to initialize the disk space and to bring the database server into online mode.

Table 7-5 Commands to start the server with disk space initialization

Operating system	Action to bring database server into online mode
UNIX	You must be logged in as <i>informix</i> or <i>root</i> to initialize the database server. Execute oninit -iy .
Windows	You must be a member of the Administrators or Power Users group to initialize the database server. ▶ The database server runs as a service. In the Services control application, choose the database server service and type -iy in the Startup parameters field. Then, click Start. ▶ On the command line, use starts dbservername -iy

Important: When you execute these commands, all existing data in the database server disk space is destroyed. Use the **-i** flag only when you are starting a new instance of the database server.

You can use the **oninit -s** option to initialize shared memory and to leave the database server in quiescent mode.

Monitor the message log file referenced by the MSGPATH configuration parameter of your *onconfig* file to monitor the state of the database server. Often, the database server provides the exact nature of the problem and the suggested corrective action in the message log. For more information, see the *IBM Informix Dynamic Server Administrator's Guide, version 11.10*, G229-6359.

For Windows only: When you install the database server and choose to initialize a new instance of the database server, or when you use the instance manager program to create a new instance of the database server, the database server is initialized and started for you.

7.2.5 Creating additional storage spaces and chunks

You are responsible for planning and implementing the storage configuration. However, be aware that the way that you distribute the data on disks can affect the performance of the database server. A *chunk* is the same as a logical volume, physical unit, or regular file that has been assigned to the database server. The maximum size of an individual chunk is four terabytes (4 TB), and the number of allowable chunks is 32,766. A logical storage space (*dbspace*) is composed of one or more chunks.

Tip: To take advantage of the large limit of 4 TB per chunk, assign a single chunk per disk drive. This data distribution can increase performance.

After the database server is initialized, you can create storage spaces, such as *dbspaces*, *blobspaces*, or *sbspaces*. Use the *onspaces* utility, *onmonitor*, or OpenAdmin Tool (OAT) to create storage spaces and chunks. The database server can use regular operating system files or raw disk devices to store data. On UNIX, use raw disk devices to store data whenever performance is important. On Windows, use NTFS files to store data for ease of administration. An Informix storage space can reside on a Network File System (NFS)-mounted file system using regular operating system files. Devices or files that are used for chunks need to exist with correct access permissions prior to adding the storage space to the database server, as demonstrated in Example 7-1. We use the *onspaces* command line utility to create a new *dbspace*, which uses a 500 MB chunk of device *c0t3d0s4*, starting at offset 0 of the device.

Example 7-1 Creating a dbspace named dbs1 of size 500 MB

```
onspaces -c -d  dbs1 -p /dev/rdisk/c0t3d0s4 -o 0 -s 500000
```

You must create a *sbspace* if you use the following functions:

- ▶ J/Foundation to hold Java JAR files
- ▶ Enterprise Replication (ER) to hold spooled row data
- ▶ Smart large objects (binary large object (BLOB) and character large object (CLOB) data types)

- ▶ Multirepresentational data types (such as DataBlade modules or HTML data types)

You can also create temporary dbspaces or add chunks to an existing dbspace, as illustrated with the ON-Monitor example in Figure 7-5 on page 349.

For a detailed discussion of the allocation and management of storage spaces, see the *IBM Informix Dynamic Server Administrator's Guide, version 11.10*, G229-6359.

After completing the disk layout and allocating all the needed storage to the database server, the DBAs can create the databases and their associated tables in the desired dbspaces as designed. The following example creates the bookdb database in the data2 dbspace:

```
create database bookdb in data2;
```

If the storage option is not used in the create database statement, the database and its associated system catalog tables are created in the root dbspace.

7.2.6 Configuring client connectivity

When the database server is online, you can connect client applications and begin to create databases. A client application can reside on the same machine as the database server or on a remote machine. Before you can access information in a database, the client application must connect to the database server environment. To connect to and disconnect from a database server, you can issue SQL statements from the following client programs:

- ▶ DB-Access
- ▶ SQL Editor
- ▶ IBM Informix ESQL/C
- ▶ IBM Informix ODBC Driver
- ▶ IBM Informix JDBC Driver

For information about how to use client applications, see these books:

- ▶ *IBM Informix DB-Access User's Guide*, SC23-9430
- ▶ *IBM Informix ESQL/C Programmer's Manual*, SC23-9420
- ▶ *IBM Informix ODBC Driver Programmer's Manual*, SC23-9423
- ▶ *IBM Informix JDBC Driver Programmer's Guide*, SC23-9421

The database administrator specifies the types of connections that the database server supports in the `sqlhosts` file on UNIX or the `PROTOCOL` field in the `SQLHOSTS` registry key on Windows.

The database server supports the following types of connections to communicate between client applications and a database server.

Table 7-6 Connection types supported

Connection type	Microsoft Windows	UNIX	Local	Network
Sockets	X	X	X	X
Transport Layer Interface (TLI) (TCP/IP)	N/A	X	X	X
TLI (IPX/SPX)	N/A	X	X	X
Shared memory	N/A	X	X	N/A
Stream pipe	N/A	X	X	N/A
Named pipe	X	N/A	X	N/A

A client application establishes a connection to a database server with either the CONNECT or DATABASE SQL statement.

For information about the client/server configurations that the database server supports, see the *IBM Informix Dynamic Server Administrator's Guide, version 11.10*, G229-6359.

DRDA

Distributed Relational Database Architecture (DRDA) is a set of protocols that enable communication between applications and database systems on separate platforms and enables relational data to be distributed among multiple platforms. Any combination of relational database management products that use DRDA can be connected to form a distributed relational database management system. DRDA coordinates communication between systems by defining what must be exchanged and how it must be exchanged.

You can configure IBM Informix to use DRDA to respond to requests from a common application programming interface (API), for example, from the IBM Data Server JDBC Driver and the IBM Data Server .NET Provider.

You can also enable DRDA connections between a client API and the Informix Connection Manager (the oncmsm utility), which manages and redirects client connection requests between a High Availability Data Replication (HDR) primary server and HDR secondary servers. The Connection Manager provides

automatic failover and load balancing capabilities. Informix ER can coexist with DRDA.

MaxConnect

IBM Informix MaxConnect enables IBM Informix to support greatly increased numbers of client connections. MaxConnect is a software tier, introduced between the database server and clients, that transparently funnels multiple client connections onto a smaller number of server connections. The database server is freed from managing thousands of client connections, which results in improved response time and decreased CPU cost for the database server.

MaxConnect is best for OLTP data transfers, but it is less efficient for large multimedia data transfers. MaxConnect provides the following performance advantages for medium to large OLTP configurations.

MaxConnect multiplexes connections so that the ratio of client connections to database connections can be 100:1 or higher. A multiplexed connection uses a single network connection between the database server and a client to handle multiple database connections from the client. If you need to manage several hundreds or thousands of client connections, consider ordering IBM Informix MaxConnect.

For more information about installing, configuring, monitoring, and tuning MaxConnect, see the *Guide to Informix MaxConnect, Version 1.1*, G251-0577.

Important: Informix MaxConnect and the *Guide to Informix MaxConnect, Version 1.1*, G251-0577, ship separately from the IBM Informix database server.

7.3 Informix administration utilities

You can perform Informix database server administration either by using a GUI tool or using a command-line administration tool. In this section, we discuss several of the most common administration utilities that are used to monitor, manage, and configure Informix.

Multiple users for administration mode

The new administration mode enhances and replaces single-user mode as a way to temporarily restrict access to the database server to perform administration tasks. The user *informix* or a Database Server Administrator (DBSA) can now dynamically give one or more specific users the ability to connect to the database server in administration mode. You enable administration mode by using a new

onmode command option (**onmode -j -U**), a new oninit command option (**oninit -U**), or the new ADMIN_MODE_USERS configuration parameter.

7.3.1 OpenAdmin Tool for Informix

OpenAdmin Tool (OAT) is a PHP-based web browser administrative tool for managing one or more Informix. The OpenAdmin Tool for Informix provides the ability to monitor and administer multiple Informix server instances from a single location. Because OAT is an open source tool, it is easy to modify, and you can tune it to meet your needs. Several of the distinct features that make OAT special are support for Informix new data replication features, such as compression, easy-to-use GUI, and automated update statistics.

OAT includes the following features:

- ▶ **Compression:** This feature uses the storage optimization functionality in Informix to save disk space by compressing tables and table fragments, consolidating free space in tables and fragments through a repack operation, and returning free space to the dbspace through the shrink operation. OAT also helps you to decide which tables or fragments to compress by graphically showing you the estimated amount of space that compression will save.
- ▶ **Health center:** This feature allows you to analyze the current state of the server with detailed statistics and alert information.
- ▶ **Logs:** You can view previous administration commands, recent messages from the online log, and the latest backup and recovery information from the ON-Bar activity log.
- ▶ **Task Scheduler:** You can view and update tasks that are scheduled to run at the current server.
- ▶ **Space administration:** You can manage server space for your system with tools to administer dbspaces, chunks, checkpoints, and the physical and logical logs.
- ▶ **Server administration:** You can configure and validate your servers, as well as administer high availability clusters (Multi-node Active Cluster for High Availability (MACH) clusters).
- ▶ **Performance analysis:** Get high quality and detailed performance statistics about your databases with the SQL exploration of each table, statement, or transaction. The capability to drill down for queries and session exploration provides detailed information about every aspect of transactions and statement types. View over 20 system performance reports, such as memory pools, the five slowest transactions, and server memory usage.
- ▶ **SQL Toolbox:** Explore your databases, view table statistics, browse schema, run queries, and view query plans.

- ▶ **Help:** View online resources for using OAT for Informix. You can easily plug your own extensions into OpenAdmin to create the functionality that you need. OpenAdmin is an open source program that you can download.

Extend OAT features with additional plug-ins:

- ▶ **Enterprise Replication:** Now, you can use the Enterprise Replication plug-in for the remote graphical setup, administration, and monitoring of replication on the domain level, node level, and replication level. The feature provides complete control over replication servers and replication sets, and it provides failed transaction recovery options.
- ▶ **Schema Manager:** Use the Schema Manager plug-in to view database and table objects. Use it to create, drop, and truncate tables, as well as to load and unload data from external tables.

For details about how to install and configure the OAT, refer to *Informix Dynamic Server 11: Extending Availability and Replication*, SG24-7488.

You can perform many administration tasks with OAT:

- ▶ Add new connections to servers and server groups
- ▶ View server information on a map
- ▶ Customize the help system to add your own content to help topics
- ▶ Configure the display to change the sort order of reports by clicking the column headings
- ▶ Manage dbspaces, chunks, and recovery logs
- ▶ Monitor performance statistics, including recent SQL statements, and combine graphs and reports to create custom reports
- ▶ View the online message log and the ON-Bar activity log
- ▶ Execute ad hoc or saved SQL statements
- ▶ View database, table, and column information
- ▶ Monitor high-availability clusters: HDR servers, shared disk secondary servers, and remote stand-alone secondary servers
- ▶ View information about executed SQL administration API commands

Figure 7-1 on page 338 is an Informix instance dbspaces view in OAT. It lists the dbspace name, type, and status. The lower portion of the window allows you to create a new dbspace.

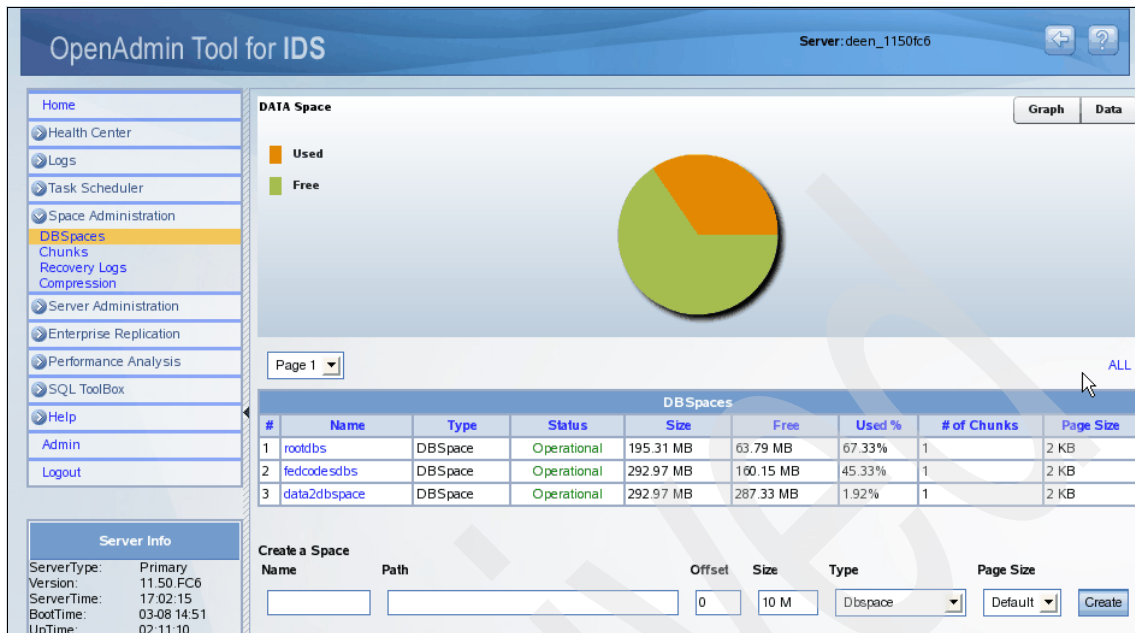


Figure 7-1 OAT dbspaces view

Drilling down on a specific dbspace name displays information about the dbspace that is organized in separate tabs, for example, Summary, Admin, Tables, and Extents. Figure 7-2 illustrates the Summary tab view for the fedcodesdbs dbspace. The Summary tab view includes the dbspace utilization graph.



Figure 7-2 Summary tab view of a dbspace

To download the latest release of OAT, visit this website:

<http://www.openadmintool.org/>

7.3.2 Command-line utilities

Several utilities enable you to monitor and manage Informix from the command line. You can use specific utilities to monitor your instance, such as `onstat` and `onlog`, which display information only. You can use other utilities to perform administration tasks, for example, `oncheck`, `onspaces`, `onmode`, and `onparams`.

In the following section, we present utilities that allow you to perform administration tasks directly from the command line. Table 7-11 on page 346 lists a summary for most of the Informix command-line administration utilities. You can obtain detailed information about these utilities in the relevant publication of the Informix documentation set, as shown in the last column of the table.

onstat

The `onstat` utility provides a way to monitor database server shared memory from the command line. It reads data from shared memory and reports statistics that are accurate for the instant during which the command executes. That is, `onstat` provides information that changes dynamically during processing, including changes in buffers, locks, indexes, and users.

The `onstat` utility displays a wide variety of performance-related and status information that is contained in the system monitoring interface (SMI) tables. You can use the `onstat` utility to check the current status of the database server and to monitor the activities of the database server.

For a complete list of all `onstat` options and their functions, use the `onstat --` command. For a complete display of all the information that `onstat` gathers, use the `onstat -a` command.

Tip: Profile information that is displayed by `onstat` commands, such as `onstat -p`, accumulates from the time that the database server was started. To clear performance profile statistics so that you can create a new profile, run `onstat -z`. If using `onstat -z` to reset statistics for a performance history or appraisal, ensure that other users do not also enter the command at separate intervals.

Table 7-7 on page 340 shows several of the `onstat` options for general information.

Table 7-7 Several common onstat command options

onstat option	Description
onstat -b	Displays information about buffers that are currently in use
onstat -l	Displays information about the physical and logical logs
onstat -R	Displays information about buffer pools, including information about the buffer pool page size
onstat -d	Displays information about all storage spaces and information about the chunks in each storage space
onstat -u	Displays a user activity profile that provides information about user threads, including the thread owner's session ID and login name

Example 7-2 illustrates sample output of the **onstat -p** command.

Example 7-2 Sample onstat -p output

```
$ onstat -p

IBM Informix Dynamic Server Version 11.50.FC6 -- On-Line -- Up 13 days 23:08:13
-- 166912 Kbytes

Profile
dskreads pagreads bufreads %cached dskwrits pagwrits bufwrits %cached
103164 536233 12364719 99.17 255523 413199 1916850 86.67

isamtot open start read write rewrite delete commit rollbk
9990629 271217 400724 6459857 555500 352955 14866 67830 1

gp_read gp_write gp_rewrt gp_del gp_alloc gp_free gp_curs
5 0 0 0 0 0 7

ovlock ovuserthread ovbuff usercpu syscpu numckpts flushes
0 0 0 864.50 493.14 3852 2275

bufwaits lokwaits lockreqs deadlks dltouts ckpwaits compress seqscans
1028 2 5475632 0 0 844 22675 25593

ixda-RA idx-RA da-RA RA-pgsused lchwaits
463 9 33477 33542 4964
```

For the complete list of all onstat options and their explanations, see *IBM Informix Dynamic Server Administrator's Reference*, SC23-7749.

Important: You can use the seconds parameter with the -r option flag to cause all other flags to execute repeatedly after they wait the specified number of seconds between each execution. This parameter and flag are handy options when you use onstat as a monitoring tool (**onstat -mr 3**).

oncheck

The oncheck utility displays information about the database disk configuration and usage, such as the number of pages used for a table, the contents of the reserved pages, and the number of extents in a table. You can display table information by using the **oncheck -pt databasename:tablename** command, as shown in Example 7-3.

Example 7-3 The output of oncheck -pt bookdb:brands command

TBLspace Report for bookdb:informix.brands				
Physical Address	3:190			
Creation date	03/11/2010	15:08:53		
TBLspace Flags	901	Page Locking		
		TBLspace contains VARCHARS		
		TBLspace use 4 bit bit-maps		
Maximum row size	106			
Number of special columns	2			
Number of keys	0			
Number of extents	1			
Current serial value	1			
Current SERIAL8 value	1			
Current BIGSERIAL value	1			
Current REFID value	1			
Pagesize (k)	2			
First extent size	8			
Next extent size	8			
Number of pages allocated	8			
Number of pages used	2			
Number of data pages	1			
Number of rows	56			
Partition partnum	3145937			
Partition lockid	3145937			
Extents				
	Logical Page	Physical Page	Size	Physical Pages
	0	3:1504	8	8

You can obtain the physical layout of the information in a chunk by using the **oncheck -pe** command.

Example 7-4 shows the types of information that are displayed as the result of the **oncheck -pe** command:

- ▶ The name, owner, and creation date of the dbspace
- ▶ The size in pages of the chunk, the number of pages used, and the number of free pages
- ▶ A listing of all the tables in the chunk, with the initial page number and the length of the table in pages

This output lists the tables within a chunk sequentially. This output is useful, for example, for determining the chunk space fragmentation (non-contiguous allocation). If the database server is unable to allocate an extent in a chunk despite an adequate number of free pages, the chunk might be badly fragmented.

Example 7-4 oncheck -pe output

DBspace Usage Report: data2dbspace Owner: informix Created: 02/15/2010

Chunk Pathname	Pagesize(k)	Size(p)	Used(p)	Free(p)
3 /space/CHUNKS/fed2codes2	2	150000	4348	145652
Description			Offset(p)	Size(p)

RESERVED PAGES			0	2
CHUNK FREELIST PAGE			2	1
data2dbspace:'informix'.TBLSpace			3	50
bookdb:'informix'.systables			53	8
bookdb:'informix'.syscolumns			61	24
bookdb:'informix'.sysindices			85	32
.....				
.....				
.....				
bookdb:'informix'.employee			1488	8
bookdb:'informix'.authors			1496	8
bookdb:'informix'.brands			1504	8
FREE			1512	35551
.....				
.....				
Total Used:	4348			
Total Free:	145652			

For the complete list of **oncheck** command options and usage, see *IBM Informix Dynamic Server Administrator's Reference*, SC23-7749.

oninit

You can run the `oninit` utility from the command line to initialize database server shared memory and to bring the database server online. If you use the `oninit -i` option, you can also initialize disk space.

Before initializing the database server, set the `INFORMIXSERVER` environment variable to the database server name that you chose when you set the configuration parameter `DBSERVERNAME`. The `INFORMIXSERVER` environment variable is not required for initialization, but if it is not set, the database server does not build the `sysmaster` tables. In addition, the DB-Access utility requires that the `INFORMIXSERVER` environment variable is set.

Example 7-5 depicts the commands to take the database server offline and bring it back online.

Example 7-5 Database server commands

```
onmode -ky  
oninit
```

The following list shows several of the prerequisites for the `oninit` command:

On UNIX, you must log in as the *root* or *informix* user to run the `oninit` command. Make sure that the *informix* user is the only member of the *informix* group.

On Windows, Informix runs as a Windows service. Users with permissions can start and stop Informix through one of the following methods:

- ▶ Control Panel administration tools
- ▶ The `net start` and `net stop` commands

Use the `starts` utility to pass command-line arguments to `oninit`. For example, to start Informix in single-user mode, use the following command:

```
%INFORMIXDIR%\bin\starts %INFORMIXSERVER% -j
```

onparams

Use the `onparams` utility to add or drop a logical log file, change physical log parameters, and add a new buffer pool.

If you do not use any options, the `onparams` utility returns a usage statement. We list several of the `onparams` options in Table 7-8 on page 344.

Be aware that any `onparams` command will fail if a storage space backup is in progress.

Table 7-8 Common onparams options

onparams command	Purpose
onparams -a -d dbspace [-i]	Add a logical log file
onparams -d -l <i>lognum</i>	Drop a logical log file
onparams -p	Change the size or location of the physical log
onparams -b	Add a new buffer pool

We show examples of the **onparam** command in Example 7-6.

Example 7-6 Examples of onparams commands

```
onparams -a -d rootdbs -s 1000 # adds a 1000 KB log file to rootdbs

onparams -a -d rootdbs -i      # inserts the log file after the current log

onparams -d -l 7              # drops log 7

onparams -p -d dbspace1 -s 3000 # resizes and moves physical log to dbspace1
```

onmode

You can use the **onmode** utility for many purposes. For example, you can use the **onmode** utility to change the database server mode, to add shared memory, and to add or remove virtual processors. On UNIX, you must be the *root* or *informix* user to execute the **onmode** utility. On Windows, you must be a member of the Informix-Admin group. Table 7-9 displays several of the most commonly used **onmode** options.

Table 7-9 Commonly used onmode options

onmode options	Purpose
-k	Takes the database server to the offline mode and removes shared memory
-m	Takes the database server from the quiescent or administration mode to the online mode
-s	Shuts down the database server gracefully to the quiescent mode
-u	Shuts down the database server immediately to the quiescent mode
-j	Puts the database server into Single User administration mode
-a	Adds a shared memory segment

onspaces

Use the **onspaces** utility for space management. With it, you can create or modify the dbspaces, blobspaces, or sbspaces. We list several of the command options for the **onspaces** utility in Table 7-10.

You can specify a maximum of 32,766 chunks for a storage space, and you can specify a maximum of 2,047 storage spaces on the database server system. The storage spaces can be any combination of dbspaces, blobspaces, and sbspaces.

On UNIX, you must log in as the *root* or *informix* user to execute the **onspaces** command. On Windows, you must be a member of the Informix-Admin group.

You cannot use the **onspaces** utility on High Availability Data Replication (HDR) secondary servers, remote stand-alone secondary (RSS) servers, or shared disk secondary (SDS) servers.

Table 7-10 *onspaces* command options

onspaces options	Purpose
-c -d	Create a dbspace
-c -b	Create a blobspace
-c -S	Create a sbspace
-a	Add a chunk to a dbspace, blobspace, or sbspace
-c -x	Create an extspace
-d	Drop a blobspace, dbspace, extspace, or sbspace
-d	Drop a chunk in a dbspace, blobspace, or sbspace
-ren	Rename a dbspace, blobspace, sbspace, or extspace
-m	Start Mirroring
-r	Stop Mirroring
-s	Change the status of a mirrored chunk
-f	Specify DATASKIP parameter -f

In Example 7-7 on page 346, we show how to create an sbspace named `vspace1` and a dbspace named `db1`, each of which is 20 MB in size.

Example 7-7 Creating an sbspace and a dbspace

On UNIX:

```
onspaces -c -S vspace1 -p /home/informix/chunk2 -o 0 -s 20000  
onspaces -c -d dbs1 -p /home/informix/chunk3 -o 0 -s 20000
```

On Windows:

```
onspaces -c -S vspace1 2 -p d:\home\informix\chunk2 -o 0 -s 20000  
onspaces -c -d dbs1 -p c:\home\informix\chunk3 -o 0 -s 20000
```

Important: The device or file that is used in the creation of a chunk or dbspace must have the correct permission, such as read, write, or execute, prior to running the **onspaces** command.

Table 7-11 lists the Informix command-line administration utilities.

Table 7-11 Informix utilities

Utility	Description	Where described
archecker	Verifies backups and performs table-level restores.	<i>IBM Informix Backup and Restore Guide, version 11.10, G229-6361</i>
cdr	Controls ER operations.	<i>IBM Informix Dynamic Server Enterprise Replication Guide, v11.50, SC23-7755</i>
dbexport	Unloads a database into text files for later importing into another database and creating a schema file.	<i>IBM Informix Migration Guide, SC23-7758</i>
dbimport	Creates and populates a database from text files. Use the schema file with dbimport to recreate the database schema.	<i>IBM Informix Migration Guide, SC23-7758</i>
dbload	Loads data into databases or tables.	<i>IBM Informix Migration Guide, SC23-7758</i>
dbschema	Creates a file that contains the SQL statements that are needed to replicate a specified table, view, or database, or to view the information schema.	<i>IBM Informix Migration Guide, SC23-7758</i>
ism	Manages IBM Informix Storage Manager, storage devices, and media volumes.	<i>IBM Informix Storage Manager Administrator's Guide, v2.2, G229-6388</i>

Utility	Description	Where described
onaudit	Manages audit masks and auditing configurations.	<i>IBM Informix Security Guide, v11.50, SC23-7754</i>
onbar	Backs up and restores storage spaces and logical logs.	<i>IBM Informix Backup and Restore Guide, version 11.10, G229-6361</i>
oncheck	Checks specified disk structures for inconsistencies, repairs inconsistent index structures, and displays information about disk structures.	<i>IBM Informix Dynamic Server Administrator's Reference, SC23-7749</i>
oncmsm	Starts the Connection Manager, which manages and redirects client connection requests based on service level agreements (SLAs) configured by the system administrator.	<i>IBM Informix Dynamic Server Administrator's Reference, SC23-7749</i>
ondblog	Changes the logging mode.	<i>IBM Informix Dynamic Server Administrator's Reference, SC23-7749</i>
oninit	Brings the database server online.	<i>IBM Informix Dynamic Server Administrator's Reference, SC23-7749</i>
onload	Loads data that was created with onunload into the database server.	<i>IBM Informix Migration Guide, SC23-7758</i>
onlog	Displays the contents of logical log files.	<i>IBM Informix Dynamic Server Administrator's Reference, SC23-7749</i>
onmode	Changes the database server operating mode and performs various other operations on shared memory, sessions, transactions, parameters, and segments.	<i>IBM Informix Dynamic Server Administrator's Reference, SC23-7749</i>
ON-Monitor	Performs administration tasks using the ON-Monitor menus.	<i>IBM Informix Dynamic Server Administrator's Reference, SC23-7749</i>
onparams	Modifies the configuration of logical logs or physical logs.	<i>IBM Informix Dynamic Server Administrator's Reference, SC23-7749</i>

Utility	Description	Where described
onpassword	Encrypts and decrypts password files for ER and Connection Manager.	<i>IBM Informix Dynamic Server Administrator's Reference</i> , SC23-7749
onperf	Monitors database server performance (creates graphs, queries trees, and shows status and metrics).	<i>IBM Informix Dynamic Server Performance Guide</i> , v11.50, SC23-7757
onpladm	Writes scripts and creates files that automate data load and unload jobs.	<i>IBM Informix High-Performance Loader User's Guide</i> , v11.50, SC23-9433
onshowaudit	Extracts information from an audit trail.	<i>IBM Informix Security Guide</i> , v11.50, SC23-7754
onspaces	Modifies dbspaces, blobspaces, sbspaces, or extspaces.	<i>IBM Informix Dynamic Server Administrator's Reference</i> , SC23-7749
onstat	Monitors the operation of the database server.	<i>IBM Informix Dynamic Server Administrator's Reference</i> , SC23-7749
ontape	Logs, backs up, and restores data.	<i>IBM Informix Backup and Restore Guide</i> , version 11.10, G229-6361
onunload	Unloads data from the database server.	<i>IBM Informix Migration Guide</i> , SC23-7758

7.3.3 ON-Monitor utility

The ON-Monitor utility is an additional administration utility that is part of the database server executables and that is used to perform various administration tasks. The utility provides character-based menu driven commands to facilitate monitoring and administering the components of an Informix instance.

In this section, we briefly introduce you to the utility menus by using an add chunk example.

If you are logged in as user *informix* or user *root*, the main menu appears, as illustrated in Figure 7-3 on page 349. All users other than *informix* and *root* have access only to the Status menu.


```
Dynamic Server: S Status Parameters Dbspaces Mode Force-Ckpt Archive Logical-Logs Exit
Status menu to view Dynamic Server.

-----On-Line----- Press CTRL-W for Help. -----
```

Figure 7-3 ON-Monitor main menu

The following example illustrates the addition of a chunk to an existing dbspace using On-Monitor menu commands:

1. From the main menu, choose **Dbspaces**; see Figure 7-4.

```
DBSPACES: C Create BLOBspace Mirror Drop Info Add_chunk datasKip Status Exit
Create a new dbspace.

-----On-Line----- Press CTRL-W for Help. -----
```

Figure 7-4 Dbspaces menu

2. Chose **Add Chunk**. Figure 7-5 shows the resulting window.

```
Press ESC to return to the Dbspaces menu.
Use arrow keys to move the cursor.
Press F3 or CTRL-B to choose the highlighted DBspace/BLOBspace.

LIST OF DBSPACES/BLOBSACES

Dbspace/BLOBspace Name          When Created      Status
lootdbbs                   02/15/2010        N B
fedcodesbbs                   02/15/2010        N B
data2dbspace                  02/15/2010        N B
```

Figure 7-5 On-Monitor add chunk command

3. Highlight the desired dbspace to which you need to add a chunk. Press Ctrl-B so that the ADD CHUNK TO DBSPACE parameters entry window appears. This window allows you to fill in the values, as shown in Figure 7-6 on page 350.

```
Press ESC to add new chunk(s).
Press Interrupt to cancel the option and return to the Dbspaces menu.
Press F2 or CTRL-F for field level help.

                        ADD CHUNK TO DBSPACE

Dbospace Name [data2dbospace  ]      Mirror [N]      Temp [N]
Dbospace Page Size [ 0] Kbytes

PRIMARY CHUNK INFORMATION:
Full Pathname [Dbospace/CHUNKS/redbook_chunk      ]
Offset [ 0] Kbytes      Size [ 20000] Kbytes

MIRROR CHUNK INFORMATION:
Full Pathname [      ]
Offset [ 0] Kbytes
```

Figure 7-6 New chunk values entered

4. After you enter the values, and you press the Esc key, a message appears indicating that the chunk was successfully added.

Although the ON-Monitor utility provides a character-based menu driven administration utility, you can perform all the tasks, which are performed by the ON-Monitor utility, from the command line by using the appropriate command-line utility.

For more details about how to navigate and use the ON-Monitor utility, see the *IBM Informix Dynamic Server Administrator's Reference*, SC23-7749.

7.4 Database server monitoring

Monitoring the database server activities, resources, and the operating system resource utilization is one of the essential tasks of database administration. In this section, we describe common monitoring commands and queries to get you started. For expanded details about server monitoring and performance considerations, see *IBM Informix Dynamic Server Performance Guide*, v11.50, SC23-7757.

IBM Informix initialization creates an internal system database, which is called *sysmaster*, that is used for the purpose of system monitoring. Its tables are referred to as the *system-monitoring interface* (SMI).

The SMI consists of tables and pseudo-tables that the database server maintains automatically. While the SMI tables appear to the user as tables, they are not recorded on disk in the way that typical tables are recorded. Instead, the database server constructs the tables in memory, on demand, based on

information in shared memory at that instant. When you query SMI tables, the database server reads information from these shared-memory structures. The SMI tables provide information about the state of the database server. You can query these tables to identify processing bottlenecks, determine resource usage, track session or database server activity, and so on.

We query the SMI tables in this section when we monitor several of the major resources:

- ▶ Operating system and database server memory utilization
- ▶ System utilization
- ▶ Sessions, active queries, temporary tables, and session memory
- ▶ Cache utilization
- ▶ Disk space utilization

7.4.1 Memory monitoring

One of the major resources on the host operating system is the memory that is used by the database server and the client applications. With a focus on Informix, the monitoring provides information, such as the amount of allocated shared memory segments and their sizes. The database server processes maintain all their data in the shared memory segments. The use of the local data segments in the process is itself low, and so, no special monitoring activity is required. As shown in Example 7-8, you can see the current amount of allocated shared memory segments and their utilization. You can map the onstat output with the `ipcs -m` operating system command output by using the segment ID in the first column.

Example 7-8 Monitoring shared memory segments in Informix

```
$ onstat -g seg
```

IBM Informix Dynamic Server Version 11.50.FC6 -- On-Line -- Up 00:07:24 --
150528 Kbytes

Segment Summary:

id	key	addr	size	ovhd	class	blk used	blk free
402653296	52694801	10a000000	119537664	1834048	R	29133	51
402653297	52694802	111200000	33554432	394720	V	5348	2844
402653298	52694803	113200000	1048576	13552	M	138	118
Total:	-	-	154140672	-	-	34619	3013

(* segment locked in memory)

```
$ ipcs -m | grep informix
m 402653298 0x52694803 --rw-rw-rw- root informix
```

m	402653297	0x52694802	--rw-rw----	root	informix
m	402653296	0x52694801	--rw-rw----	root	informix

Example 7-9 shows similar information that is obtained from SMI.

Example 7-9 Query sysmaster database for shared memory segment information

> **select * from sysseg1st;**

seg_address	4582277120
seg_next	4615831552
seg_prev	4462739456
seg_class	2
seg_size	33554432
seg_osshmid	402653297
seg_osmaxsize	33554432
seg_osshmkey	1382631426
seg_procid	13149
seg_userid	0
seg_shmaddr	4582277120
seg_ovhd	394720
seg_lock	4582277680
seg_nextid	402653298
seg_bmapsz	393216
seg_blkused	5406
seg_blkfree	2786

If you want to drill down more deeply into the memory utilization of the allocated shared memory segments, use the **onstat -g mem** command. Example 7-10 depicts typical output. This output shows you the allocated pools that are in the shared memory segments, the amount of memory that is allocated for each pool, and the distribution of the memory in free and used memory.

Example 7-10 Memory pool summary displayed with onstat -g mem

\$ **onstat -g mem**

IBM Informix Dynamic Server Version 11.50.FC6 -- On-Line -- Up 01:38:12 -- 150528 Kbytes

Pool Summary:

name	class	addr	totalsize	freesize	#allocfrag	#freefrag
afpool	V	111271040	8192	2040	5	3
tpcpool	V	111f87040	20480	4376	13	3
proxy_10	V	11137d040	4096	808	1	1
.....						
.....						
pn1pool	V	111f8a040	24576	3288	18	3
proxy_13	V	111380040	4096	808	1	1
ampool	V	111fb8040	4096	304	7	1

shmcon	M	113204040	548864	4864	2	2
onmode_monV		112493040	28672	3208	13	3
proxy_100	V	1113d7040	4096	808	1	1
2	V	1121b6040	12288	536	16	1
proxy_110	V	1113e1040	4096	808	1	1
rsam	V	1114d4040	598016	1816	607	4
sscpool0	V	111f7c040	12288	1608	2	2
smartblob	V	1113f4040	16384	6504	7	2
glS	V	111f79040	36864	2416	348	4
res-buff0	R	10b01f040	102424576	4864	2	2
polycypoolV		111f8d040	8192	3288	2	2
sb_delq	V	1113fe040	49152	8752	4	3
resident	R	10a1c0040	15069184	4864	2	2
dictpool	V	111f7d040	286720	3576	92	3
mt	V	111272040	2129920	8272	5474	101

Blkpool Summary:

name	class	addr	size	#blks
mt	V	1112749b8	2560000	33
global	V	11126f308	0	0

Example 7-11 shows more specific and similar information that is obtained from SMI.

Example 7-11 Query sysmaster syspoolst table for memory pool information

```
> select * from syspoolst where po_name='res-buff0';
```

```
po_id          2
po_address     4479643712
po_next       4479643728
po_prev       4479643720
po_lock       0
po_name       res-buff0
po_class      1
po_flags      0
po_freeamt    4864
po_usedamt    102419712
po_freelist   4479643856
po_list       4479643816
po_sid       0
```

1 row(s) retrieved.

7.4.2 Process utilization and configuration

A specific Informix database server instance is defined by a specific number of processes. The number of processors that is dedicated to the instance is stable and independent from the number of clients connecting to the database server. If you want to monitor the number of server processes and their workload, you can use the **onstat -g glo** command. Example 7-12 shows the details of a possible output. You can select the sysmaster table named sysvplst for the same information using the SQL interface.

Example 7-12 Individual and cumulative virtual process statistics

#onstat -g glo

Virtual processor summary:

class	vps	usercpu	syscpu	total
cpu	4	1947.94	461.54	2409.48
aio	10	228.86	447.69	676.55
lio	1	20.12	38.17	58.29
pio	1	24.68	56.40	81.08
adm	1	182.66	299.36	482.02
soc	1	130.97	540.31	671.28
msc	1	0.00	0.01	0.01
total	19	2535.23	1843.48	4378.71

Individual virtual processors:

vp	pid	class	usercpu	syscpu	total
1	4337	cpu	343.62	97.09	440.71
2	4338	adm	182.66	299.36	482.02
3	4339	cpu	525.95	112.75	638.70
4	4340	cpu	535.50	114.64	650.14
5	4341	cpu	542.87	137.06	679.93
6	4342	lio	20.12	38.17	58.29
7	4343	pio	24.68	56.40	81.08
8	4344	aio	20.74	42.63	63.37
9	4345	msc	0.00	0.01	0.01
10	4346	aio	24.14	56.59	80.73
11	4347	aio	21.00	40.94	61.94
12	4348	aio	27.68	53.03	80.71
13	4349	aio	19.06	37.33	56.39
14	4350	aio	21.86	39.81	61.67
15	4351	aio	23.37	42.26	65.63
16	4352	aio	29.99	56.47	86.46
17	4353	aio	18.72	39.23	57.95
18	4354	aio	22.30	39.40	61.70
19	4355	soc	130.97	540.31	671.28
		tot	2535.23	1843.48	4378.71

```
dbaccess -e sysmaster << EOF
select * from sysvplst;
EOF
```

```

vpid      1
address   4579319848
pid       4337
usecs_user 343.6400000000
usecs_sys 97.0900000000
scputimep 4579319864
....

```

Monitoring the processes and their system utilization is useful from the operating system monitoring point of view. From the database administration point of view, you can achieve a much better view of the granularity of the workload of the database server by looking at the threads. Informix internally creates its own threads to handle the work for incoming client requests. The threads are executed on all processes of the name type CPU VP. Look at the output of the **onstat -g ath**, **onstat -g act**, and **onstat -g rea** commands. These outputs give you either the complete view of the existing threads in the system, or a particular view of the threads with the same status in the system. Example 7-13 shows a sample output of all the threads that currently exist on the server, their names, current status, and thread ID, which is needed to map to other onstat outputs.

Example 7-13 onstat -g ath command output

```

Threads:
tid  tcb      rstcb      prty  status          vp-class  name
2   1120c84a8  0          1    IO Idle         3lio*    lio vp 0
3   1120e72a8  0          1    IO Idle         4pio*    pio vp 0
4   1121062a8  0          1    IO Idle         5aio*    aio vp 0
5   1121252a8  0          1    IO Idle         6msc*    msc vp 0
6   1121542a8  0          1    IO Idle         7fifo*   fifo vp 0
7   1121742a8  0          1    IO Idle         8aio*    aio vp 1
8   112193548  1115bf028  3    sleeping secs: 1 14cpu    main_loop()
9   112106cc8  0          1    running          9soc*    soctcppoll
10  112125830  0          1    running          1cpu*    sm_poll
14  11224e028  1115bf880  1    sleeping secs: 1 14cpu    flush_sub(0)
....
....
26  1123be698  1115c3b40  2    sleeping secs: 1 14cpu    aslogflush
27  11239fa78  1115c4398  1    sleeping secs: 39 1cpu     btscanner_0
43  1125d5568  1115c5ca0  3    sleeping secs: 1 1cpu*    onmode_mon
44  1125d5850  1115c5448  3    sleeping secs: 1 14cpu    periodic
52  11266f8b0  1115c64f8  1    sleeping secs: 117 1cpu*    dbScheduler
53  1126ddbfb  1115c6d50  1    sleeping forever 1cpu     dbWorker1
56  11259b9d8  1115c7e00  1    cond wait netnorm 1cpu     sqlexec
160 112d0ebe8  1115c8658  1    cond wait netnorm 1cpu     sqlexec

```

7.4.3 Disk space monitoring

In addition to the memory and processor utilization, the disk space is also a critical resource that is provided by the operating system and used by the database server. Informix views a dbspace as a logical organizational unit, combining disks for use in the server instance for objects, such as logs, tables, and indexes. We define chunks as the physical unit that describes the view of the database server to the physical files and raw devices. You can use the **onstat -d** command for monitoring the dbspace and chunk statistics.

Example 7-14 shows the sample output for a system with three dbspaces and one sbspace with their respective chunks. From the administration point of view, the parameter that is attached to the chunk is important. Look particularly at the sizes, the location, and the free parameter. Define high watermarks for the free parameter to react appropriately before the chunk runs out of disk space and your production environment is affected by an out of disk space condition.

Example 7-14 onstat -d sample output

Dbspaces									
address	number	flags	fchunk	nchunks	pgsize	flags	owner	name	
1114d5028	1	0x40001	1	2	2048	N B	informix	rootdbs	
1114d55e0	2	0x40001	2	1	2048	N B	informix	fedcodesdbs	
1114d5778	3	0x40001	3	2	2048	N B	informix	data2dbspace	
1114d5910	4	0x48001	6	1	2048	N SB	informix	vspace1	
4 active, 2047 maximum									

Chunks									
address	chunk/	offset							
	/dbs	size	free	bpages	flags	pathname			
1114d51c0	1 1	0	100000	166	P0-B-	/chunks/root_1150FC6			
1114d5c98	2 2	0	150000	81995	P0-B-	/CHUNKS/fedcodes			
112342028	3 3	0	150000	145652	P0-B-	/CHUNKS/fed2codes2			
112342218	4 3	0	10000	9997	P0-B-	/CHUNKS/books_chunk			
1114d5aa8	5 1	0	50000	49703	P0-B-	/chunks/root2_1150FC6			
112342408	6 4	0	10000	9250	POSB-	/chunks/chunk2			
		Metadata	697	518	697				
6 active, 32766 maximum									

NOTE: The values in the "size" and "free" columns for DBspace chunks are displayed in terms of "pgsize" of the DBspace to which they belong.

Expanded chunk capacity mode: always

You can obtain the similar information from the SMI, as illustrated in Example 7-15.

Example 7-15 Query sysmaster database for dbspace and chunk information

```
$ dbaccess -e sysmaster << EOF
> select * from sysdbspaces;
```



```
> select * from syschunks;  
> EOF
```

Database selected.

```
select * from sysdbspaces;
```

dbnum	1
name	rootdbs
owner	informix
pagesize	2048
fchunk	1
nchunks	2
is_mirrored	0
is_blobspace	0
is_sbspace	0
is_temp	0
flags	262145

.....

.....

```
select * from syschunks;
```

chknun	1
dbnum	1
nxchknun	5
pagesize	2048
chksize	100000
offset	0
nfree	166
mdsize	-1
udsize	-1
udfree	-1
is_offline	0
is_recovering	0
is_blobchunk	0
is_sbchunk	0
is_inconsistent	0
flags	196672
fname	/testing/chunks/root_1150FC6
mfname	
moffset	
mis_offline	0
mis_recovering	0
mflags	

7.4.4 Session monitoring

Informix provides two types of views for session monitoring. You can use a table-oriented global view with the **onstat -g sql** command. Or, you can use the **onstat -g ses** command to show session basics, such as the user name, the current statement type, database and status information, and the accumulated values for used memory, as depicted in Example 7-16.

Example 7-16 output of onstat -g ses and onstat -g sql

onstat -g ses

IBM Informix Dynamic Server Version 11.50.FC6 -- On-Line -- Up 22:05:51 -- 158720 Kbytes

session					#RSAM	total	used	dynamic
id	user	tty	pid	hostname	threads	memory	memory	explain
10164	informix	-	0	-	0	12288	11752	off
10163	informix	27	4429	blazer	1	86016	69024	off
48	informix	27	13893	blazer	1	69632	61128	off
31	informix	27	13771	blazer	1	180224	170472	off
30	informix	27	13764	blazer	1	114688	110536	off
26	root	15	13719	blazer	1	110592	89760	off
24	informix	-	0	-	1	466944	394840	off
23	informix	-	0	-	1	376832	309192	off
22	informix	-	0	-	1	462848	385760	off
8	informix	20	13177	blazer	1	110592	88984	off
7	informix	-	0	-	0	12288	11752	off

\$onstat -g sql

Sess	SQL	Current	Iso	Lock	SQL	ISAM	F.E.	
Id	Stmt type	Database	Lvl	Mode	ERR	ERR	Vers	Explain
10164	UPDATE	sample2	CR	Not Wait	0	0	9.24	Off
10163	INSERT	sample2	CR	Not Wait	0	0	9.24	Off
48	-	sample2	CR	Not Wait	0	0	9.24	Off
31	SELECT	bookdb	NL	Not Wait	0	0	9.24	Off
30	SELECT	codes	CR	Not Wait	0	0	9.24	Off
26	-	bookdb	NL	Not Wait	0	0	9.24	Off
24		sysadmin	DR	Wait 5	0	0	-	Off
23		sysadmin	DR	Wait 5	0	0	-	Off
22		sysadmin	DR	Wait 5	0	0	-	Off
8	-	sysmaster	CR	Not Wait	0	0	9.24	Off

You can obtain the system session information from sysmaster, as shown in Example 7-17 on page 359.

Example 7-17 SMI used to obtain session information

```
dbaccess -e sysmaster << EOF
select * from sysessions;
EOF
```

If you want to have more details about a particular session, use the **onstat -g ses** or **onstat -g sql** command, followed by the session ID, to obtain session-specific details, as shown in the sample results in Figure 7-7.

```
# onstat -g ses 31
IBM Informix Dynamic Server Version 11.50.FC6 -- On-Line -- Up 22:16:50 -- 158720 Kbytes

session      effective      tty      pid      hostname      #RSAM      total      used      dynamic
id            user            user            pid      hostname      threads    memory    memory    explain
31            informix        -            27       13771         blazer     1         180224    170472    off

tid          name      rstcb      flags      curstk      status      netnorm
164          sqlexec   1115c9708   Y--P---    7839        cond wait   -

Memory pools
name          class addr      totalsize freesize  #allocfrag #freefrag
31            V         112adc040      176128    8944       209        17
31*00         V         1128c2040      4096      808        1          1

name          free      used      name          free      used
overhead      0         6576      scb            0         144
opentable     0         4312      filetable     0         1184
log           0         16536     temprec       0         21664
keys          0         680       ralloc        0         78448
gentcb        0         1584      ostcb         0         2816
sort          0         104       sqscb         0         20872
sql           0         72        rdahead       0         1120
hashfiletab   0         552       osenv         0         2832
sqtcdb        0         8840      fragman       0         792
udr           0         1344

sqscb info
scb           sqscb      optofc      pdqpriority  optcompind  directives
112a3d0c0     11286b028  0           0             2            1

Sess          SQL          Current      Iso Lock      SQL  ISAM F.E.
Id            Stmt type    Database    Lvl Mode     ERR  ERR  Vers  Explain
31            SELECT      redbook     NL Not Wait  0    0    9.24  Off

Current statement name : colcur

Current SQL statement :
select colno, colname, coltype, collength,
informix.syscolumns.extended_id, name from informix.syscolumns,
informix.systables, outer informix.systdypes where
informix.syscolumns.tabid = informix.systables.tabid and
informix.syscolumns.extended_id = informix.systdypes.extended_id and
tablename = ? and informix.systables.owner = ? order by
informix.syscolumns.colno;

Host variables :
address      type      flags value
-----
0x000000001129071b0 CHAR      0x000 authors
0x00000000112907240 CHAR      0x000 informix
```

Figure 7-7 onstat -g ses sid sample output

Additionally, DBAs use the **onstat -u** command for user threads frequently, because it depicts user activities and the number of reads and writes for each user. See Example 7-18 on page 360.

Example 7-18 onstat -u output

Userthreads		ses									
address	flags	id	User	tty	wait	tout	locks	nreads	nwrites		
1115bf028	---P--D	1	root	-	0	0	0	128	1248		
1115bf880	---P--F	0	root	-	0	0	0	0	1519		
1115c00d8	---P--F	0	root	-	0	0	0	0	6503		
1115c0930	---P--F	0	root	-	0	0	0	0	3		
1115c1188	---P--F	0	root	-	0	0	0	0	0		
1115c19e0	---P--F	0	root	-	0	0	0	0	0		
1115c2238	---P--F	0	root	-	0	0	0	0	0		
1115c2a90	---P--F	0	root	-	0	0	0	0	0		
1115c32e8	---P--F	0	root	-	0	0	0	0	0		
1115c3b40	---P---	9	root	-	0	0	0	0	2		
1115c4398	---P--B	10	root	-	0	0	0	26	0		
1115c4bf0	Y--P--D	25	root	-	10a24b8f0	0	0	0	0		
1115c5448	---P--D	14	root	-	0	0	0	2	0		
1115c5ca0	---P--D	13	root	-	0	0	0	0	48		
1115c64f8	---P---	22	informix	-	0	0	1	192	104		
1115c6d50	---P---	23	informix	-	0	0	1	143	473		
1115c75a8	---P---	24	informix	-	0	0	1	144	343		
1115c7e00	Y--P---	8	informix	20	11259b798	0	2	11	0		
1115c8658	Y--P---	26	root	15	112ae5cd8	0	1	48	0		
1115c8eb0	Y--P---	30	informix	27	112ae5028	0	1	74	0		
1115c9708	Y--P---	31	informix	27	112ae5ba0	0	1	62	0		
1115c9f60	Y--P---	48	informix	27	112c84148	0	1	0	0		
1115cb010	Y--P---	10163	informix	27	112d558c0	0	1	1	13441		
23 active, 128 total, 25 maximum concurrent											

For a detailed explanation of the output fields, see *IBM Informix Dynamic Server Administrator's Reference*, SC23-7749.

7.4.5 Cache monitoring

The database server uses separate caches to maintain the information about already evaluated and still valid database objects. The object is evaluated one time by reading the data from the system catalog from disk. All subsequent readers can use the cache to access the information, either exclusively or shared, depending on the current requests. Using the cache to access the information, in return, improves performance by eliminating the need to read this information again from disk. See Figure 7-8 on page 361. After the object is invalidated in cache, it is removed. The database server has various types of caches that can be enabled and monitored:

- ▶ Data dictionary
- ▶ Data distribution

- SQL statement
- User-defined routines (UDR)

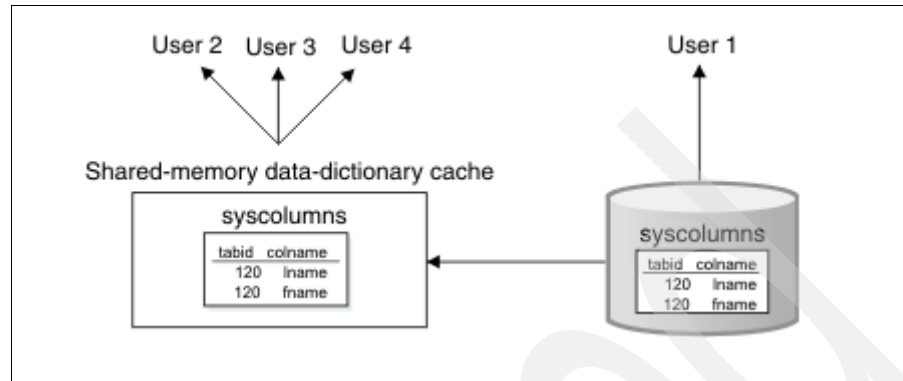


Figure 7-8 Data dictionary cache

You can enable, configure, and monitor the various caches in the database server for objects, such as tables, stored procedures, and user-defined types or statements. See Table 7-12 for a list of caches on the server, the command to monitor each cache, and the configuration parameters that are used to enable or tune the cache.

Table 7-12 IBM Informix caches

Cache	Monitoring command	Enable, configure, and tune
Data dictionary	onstat -g dic	DD_HASHSIZE DD_HASHMAX
Data distribution	onstat -g dsc	DS_POOLSIZE DS_HASHSIZE
SQL statement	onstat -g ssc	STMT_CACHE STMT_CACHE_HITS STMT_CACHE_NOLIMIT STMT_CACHE_NUMPOOL STMT_CACHE_SIZE
UDR	onstat -g prc	PC_POOLSIZE PC_HASHSIZE

Important: You can also run the **onstat -g cac** command to list the contents of other memory caches, such as the aggregate cache, as well as the UDR cache.

Example 7-19 on page 362 illustrates a sample output of the cache monitoring.

Example 7-19 Sample output of onstat commands used to monitor server cache

\$onstat -g dic

Dictionary Cache: Number of lists: 31, Maximum list size: 10

list#	size	refcnt	dirty?	heapptr	table name
0	1	0	no	1123e0038	sysmaster@srv:informix.sysdbstab
1	2	0	no	1123be038	sysmaster@srv:informix.syspollst
		0	no	112373038	sysmaster@srv:informix.systabnames
3	3	0	no	112571838	sysmaster@srv:informix.sysdbspaces
		0	no	11236c838	sysadmin@srv:informix.mon_vps
		0	no	1123ab838	sysmaster@srv:informix.systcblst
4	3	0	no	1122bb038	sysmaster@srv:informix.syscheckpoint
		0	no	1128ba038	sysadmin@srv:informix.systables
		0	no	1128c6838	sysadmin@srv:informix.ph_version
5	1	1	no	112340038	sysadmin@srv:informix.mon_config
6	2	0	no	112388038	sysmaster@srv:informix.sysshmhdr
		0	no	1123f8838	sysmaster@srv:informix.systables
7	1	0	no	112270838	sysmaster@srv:informix.sysseglst
8	1	0	no	11231e038	sysmaster@srv:informix.sysenv
9	1	0	no	1128db838	sysadmin@srv:informix.sysindices
11	4	1	no	1122e5838	stores_demo@srv:informix.systables
		0	no	11238f838	sysmaster@srv:informix.sysvplst
		0	no	1123d5838	sysmaster@srv:informix.sysptnhdr
		6	no	1128e6838	sysadmin@srv:informix.ph_run
14	1	7	no	1112f6568	sysadmin@srv:informix.ph_task

\$onstat -g prc

UDR Cache:

Number of lists : 31
PC_POOLSIZE : 127

UDR Cache Entries:

list#	id	ref_cnt	dropped?	heap_ptr	udr name
0	27	0	0	1123f7038	sysadmin@srv:.destroy
2	380	0	0	11231b438	sysadmin@srv:.check_backup
3	33	0	0	1122ab438	sysadmin@srv:.destroy
3	133	0	0	112319438	sysadmin@srv:.assign
4	28	0	0	112312838	sysadmin@srv:.assign

8	378	0	0	1122a0c38	sysadmin@srv:.onconfig_save_diffs
8	49	0	0	1123e4838	sysadmin@srv:.destroy

7.4.6 Non-Informix system monitoring tools

The system administrator and the DBA are encouraged to use the operating system utilities to observe network and operating system resource allocations and activities. There are several commonly available tools:

- ▶ vmstat
- ▶ sar
- ▶ top
- ▶ netstat
- ▶ ipcs
- ▶ ps
- ▶ Windows Sysinternals

Determining which tool to use depends on the system component that you are trying to diagnose.

7.5 High availability and disaster recovery

Informix provides many rich high availability mechanisms. In this section, we introduce the high availability features, in addition to data replication, that are designed to enhance Informix fault tolerance and high availability. Refer to 7.6, “Data replication to enable business continuity” on page 370 for a detailed description of the data replication, its role in business continuity and high availability, and Multi-node Active Cluster for High Availability (MACH11) features.

Informix provides many rich high availability mechanisms. In this section, we introduce the high availability features that are designed to enhance Informix fault tolerance and high availability.

Informix uses the following logging and recovery mechanisms to protect data integrity and consistency if an operating system or media failure occurs:

- ▶ Backup and restore
- ▶ Fast recovery
- ▶ Autonomic
- ▶ Dynamic configuration
- ▶ Mirroring

7.5.1 Backup and restore

One of the high availability features in Informix is its ability to back up your data while the database server is online and processing without affecting transaction concurrency, performance, or data integrity. Informix design allows for logical transaction log backup, in addition to the physical dbspace backup. Therefore, in the case of a disaster, you can restore your system without any or with a minimal loss of transaction data but with assured consistency and reliable data integrity.

Informix offers the `ontape` and `ON-Bar` utilities to facilitate online database server backup and restore operations. Use the `ON-Bar` or `ontape` utility to back up your database server data and logical logs as insurance against lost or corrupted data. A program error or disk failure can cause data loss or corruption. If a dbspace, an entire disk, or the database server fails, use `ON-Bar` or `ontape` to restore the data from the backup copy. To restore, you must use the same utility that was used for backup. You must use the same utility for both the dbspace and the logical logs' backup to be able to restore them both successfully. See Table 7-13 on page 366 for a comparison between the `ontape` and `ON-Bar` utility capabilities.

ontape utility

The `ontape` utility is the oldest available utility and one of the simplest utilities to use. It does not require a storage manager. It can back up the data from the database server to tape or disk. The `ontape` utility can perform physical and logical log backups and restores. The `ontape` utility performs backups and restores serially. The `ontape` utility has the following major features:

- ▶ Simplicity and ease of use
- ▶ Backup capability to tape, file, or directory
- ▶ Support for table-level restore
- ▶ Support for external backup and restore
- ▶ Support for backup and restore filters
- ▶ Capability to change database logging status
- ▶ Capability to start continuous logical log backups
- ▶ Data replication setup capability
- ▶ Capability to rename chunks to separate path names and offsets

ON-Bar utility

Use the `ON-Bar` utility to perform the backup and restore of both dbspaces and logical logs. The `ON-Bar` utility requires a storage manager. Informix Storage Manager is a limited storage manager that ships with the product. Or, you can use another storage manager, such as Tivoli® Storage Manager. `ON-Bar` communicates with both the database server and the storage manager using an industry standard X/Open Backup Services Application Programmer's Interface

(XBSA). Each storage manager vendor provides the XBSA shared library to be used with their storage manager, which is how ON-Bar is able to work with a variety of storage manager software. If you use ON-Bar as your backup tool, you must set up a storage manager before you can back up and restore data. The ON-Bar utility has the following major features:

- ▶ Parallel backup and restore.
- ▶ Point-in-time restores.
- ▶ Integrated backup verification command.
- ▶ Support for external backup and restore. An external backup and restore allows you to copy and physically restore devices without using ON-Bar. Then, you can use ON-Bar for the logical restore to bring the dbspace to a consistent state.
- ▶ Support for table-level restore.
- ▶ Capability to start continuous logical log backups.
- ▶ Capability to verify a backup with the archecker utility.
- ▶ Ability to rename chunks to separate path names and offsets.

ON-Bar is packaged with IBM Informix Storage Manager, which manages data storage for the Informix database server. IBM Informix Storage Manager resides on the same computer as ON-Bar and the database server. The storage manager handles all media labeling, mount requests, and storage volumes. IBM Informix Storage Manager receives backup and restore requests from ON-Bar and directs data to and from storage volumes that are mounted on storage devices. It also tracks backed-up data through a data life cycle that the database or system administrator determines and also manages storage devices and storage volumes. In addition, it can back up data to as many as four storage devices at a time. IBM Informix Storage Manager stores data on simple tape drives, optical disk devices, and file systems. However, you can purchase a storage manager from another vendor if you want to use more sophisticated storage devices, back up to more than four storage devices at a time, or back up over a network.

When you plan your storage space and logical log backup schedule, make sure that the storage devices and backup operators are available to perform backups. For information about configuring and managing storage devices and media, see *IBM Informix Storage Manager Administrator's Guide, v2.2*, G229-6388, or your vendor-acquired storage manager documentation.

To ensure that you can recover your databases in the event of a failure, make frequent backups of your storage spaces and logical logs. You also can verify ON-Bar backups with the archecker utility.

How often you back up the storage spaces depends on how frequently the data is updated and how critical it is. A backup schedule might include a complete (level-0) backup once a week, incremental (level-1) backups daily, and level-2 backups hourly. You also need to perform a level-0 backup after performing administration tasks, such as adding a dbspace, deleting a logical log file, or enabling mirroring. Back up each logical log file as soon as it fills. You can back up these files manually or automatically.

Table 7-13 summarizes the differences between On-Bar and ontape.

Table 7-13 Comparison between onbar and ontape

Description	On-Bar	ontape
Uses a storage manager to track backups and storage media	Yes	No
Backs up selected storage spaces	Yes	No
Initializes High Availability Data Replication (HDR)	Yes	Yes
Restores data to a specific point in time	Yes	No
Performs separate physical and logical restores	Yes	Yes
Backs up and restores separate storage spaces in parallel	Yes	No
Uses multiple tape drives concurrently for backups and restores	Yes	No
Restarts a restore after an incomplete restore	Yes	No
Changes logging mode for databases	Yes	Yes

Performing a level-0 backup of all storage spaces

To perform a standard, level-0 backup of all online storage spaces and used logical logs, use the command that is shown in Example 7-20.

Example 7-20 Performing a level-0 backup of all storage spaces

```
onbar -b
or
onbar -b -L 0
-----
ontape -s -L 0
```

Informix never backs up offline storage spaces, temporary dbspaces, or temporary sbspaces.

Performing a level-0 backup of specified storage spaces

To perform a level-0 backup of specified storage spaces and all logical logs (for example, two dbspaces named `fin_dbspace1` and `fin_dbspace2`), use the **-b** option, as depicted in Example 7-21. You can also specify the **-L 0** option, but it is not necessary.

Example 7-21 Performing a level-0 backup of specified storage spaces only

```
onbar -b fin_dbspace1 fin_dbspace2
```

```
-----  
ontape  
(N/A)
```

For a short step-by-step example of setting up Informix Storage Manager on UNIX, see the short presentation at this website:

<http://eb140.elearn.ihost.com/ECM/KnowledgeNow/13/player.html>

Performing an incremental backup

An incremental backup saves all changes in the storage spaces since the last level-0 backup and performs a level-0 backup of the used logical logs. To perform a level-1 backup, use the **-L 1** option, as depicted in Example 7-22.

Example 7-22 Command to perform incremental backup

```
onbar -b -L 1
```

```
-----  
ontape -s -L 1
```

Performing a restore

To perform a complete cold or warm restore, use the **onbar -r** command. ON-Bar restores the storage spaces in parallel. To perform a restore, use the **onbar** command, as shown in Example 7-23.

Example 7-23 Command to perform a full system restore

```
onbar -r
```

```
-----  
ontape -r
```

Restoring specific storage spaces

To restore particular storage spaces (for example, dbspaces named `fin_dbspace1` and `fin_dbspace2`), use the `-r` option, as shown in Example 7-24.

Example 7-24 Command to restore specific storage spaces only

```
onbar -r fin_dbspace1 fin_dbspace2
-----
ontape -r -D fin_dbspace1 fin_dbspace2
```

For detailed information about all options of using ON-Bar and ontape, see the *IBM Informix Backup and Restore Guide, v11.50, SC23-7756*.

7.5.2 Fast recovery

Fast recovery is an Informix fault-tolerant mechanism to recover the database server to a consistent state after it goes offline under uncontrolled conditions. Fast recovery rolls forward all committed transactions since the last checkpoint and rolls back any uncommitted transactions, preserving data integrity.

When the database server starts, it checks if fast recovery is needed by checking the physical log, which contains pages that have not yet been written to disk. If the physical log is empty, the database server was shut down in a controlled fashion. If the physical log is not empty, it is an indication that we have modified pages in memory that did not make it to disk so the database server automatically performs fast recovery. For information about fast recovery, see *IBM Informix Dynamic Server Administrator's Guide, v11.50, SC23-7748*.

7.5.3 Autonomic features

The intelligence of the database server in analyzing system performance and the ability to introduce the automatic or autonomic tuning of various server components are integral to high availability.

IBM Informix offers many automatic tuning parameters that a DBA can configure to use Informix smart use, allocation, and tuning of resources, as needed, in effect, increasing server high availability. Informix has many autonomic tuning features:

- ▶ Automatic checkpoints
- ▶ Automatic least recently used (LRU) tuning
- ▶ Automatic statistics update
- ▶ Automatic management of asynchronous I/O virtual processors
- ▶ Automatic re-prepare and optimization of stored procedures

To learn how to configure and use the autonomic features in Informix, see *IBM Informix Dynamic Server Administrator's Reference*, SC23-7749.

7.5.4 Dynamic configuration

Informix continues to enhance its high availability by adding more parameters to its dynamic configuration feature.

IBM Informix offers DBAs the ability to change the configuration of the database server dynamically while the database server is online. The DBA can optionally make these changes permanent or apply them only for a specific period of time. You can use the **onmode** command with option **-wf** or **-wm** to dynamically change certain configuration parameters. The **onmode -wf** command updates the value of the specified configuration parameter in the `onconfig` file. The **onmode -wm** command dynamically sets the value of the specified configuration parameter for the current session.

Example 7-25 illustrates how to use this command.

Example 7-25 Dynamically changing the value of an onconfig file parameter

Current `onconfig` settings for `DUMPSHMEM` is 1. It can be changed to 0 permanently in the `onconfig` file as follow:

```
onmode -wf DUMPSHMEM=0
```

A message in the database server message log (`online.log`) will reflect the success of the command change as follow:

```
18:10:27 Value of DUMPSHMEM has been changed to 0.
```

Important: The **onmode -wf** and **onmode -wm** commands have an equivalent SQL administration API function.

You cannot change all configuration parameters dynamically. For more details about the feature and a list of parameters, see *IBM Informix Dynamic Server Administrator's Reference*, SC23-7749.

7.5.5 Mirroring

A disk failure is possible with any system and can render your database server partially or totally unavailable. Yet, the Informix design allows internal mirroring of the database server chunks, which allows for continuing high availability even in a disk failure situation. Informix mirroring is another high availability feature. Do

not confuse Informix high availability mirroring with any additional hardware mirroring that you might have in place.

When you use Informix disk mirroring, the database server writes each piece of data to two locations. Mirroring is a strategy that pairs a primary chunk of one storage space with an equal-sized mirrored chunk. Every write to the primary chunk is automatically accompanied by an identical write to the mirrored chunk. If a failure occurs on the primary chunk, mirroring lets you read from and write to the mirrored chunk until you can recover the primary chunk, all without interrupting user access to data or affecting performance.

Consider mirroring the following data:

- ▶ Root dbspace
- ▶ Dbspaces that contain the physical log and logical log files
- ▶ Frequently queried data

Figure 7-9 illustrates this concept of mirroring.

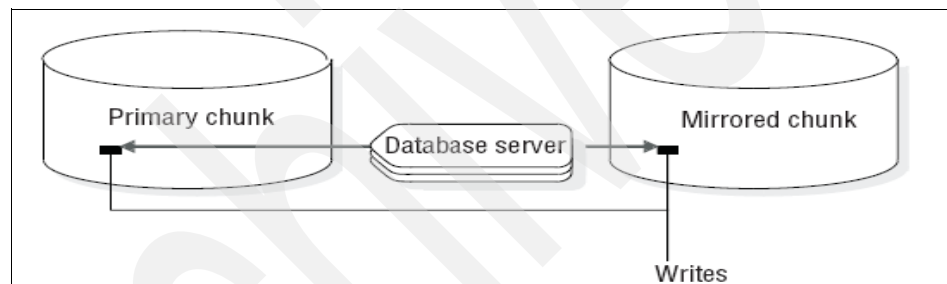


Figure 7-9 Database server writes to both the primary and mirrored chunks

For detailed information about mirroring, see the *IBM Informix Dynamic Server Administrator's Guide, v11.50, SC23-7748*.

7.6 Data replication to enable business continuity

Enterprise business success is extremely dependent on continuous and quick access to the data. Informix has various solutions of data replication to assure high availability and business continuity. *Data replication* refers to the process of representing database objects at more than one distinct site, which allows an enterprise to share corporate data throughout the organization and to better manage workload balancing across multiple servers. Data replication also can provide a backup system in the case of a catastrophic failure of one of the servers.

IBM Informix provides several high availability cluster configuration options:

- ▶ High Availability Data Replication (HDR)
- ▶ Shared disk secondary (SDS)
- ▶ Remote stand-alone secondary (RSS)
- ▶ Enterprise Replication (ER)

Multi-node Active Cluster for High Availability (MACH) configurations consist of a primary server and one or more secondary servers, such as an High Availability Data Replication secondary server, one or more shared disk secondary servers, and one or more remote stand-alone secondary servers. In addition, Informix supports IBM Informix Enterprise Replication. You can combine data replication options on the same database server. For more information, see *IBM Informix Dynamic Server Enterprise Replication Guide*, v11.50, SC23-7755.

In this section, we introduce all the available options.

7.6.1 High Availability Data Replication

High Availability Data Replication (HDR) provides synchronous data replication for Informix. Use an HDR secondary server if you require a hot standby. Configuring an HDR secondary server provides a way to maintain a backup copy of the entire database server that applications can access quickly in the event of a catastrophic failure of the primary server.

In Figure 7-10, the secondary database server is dynamically updated, with changes made to the data that the primary database server manages.

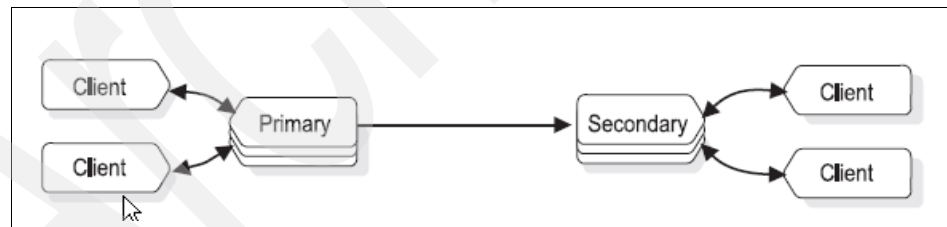


Figure 7-10 With HDR, clients write to the primary are shipped on to the secondary

if one of the database servers fails, as depicted in Figure 7-11, you can redirect the clients that were using that database server to the other database server in the pair, which then becomes the primary server.

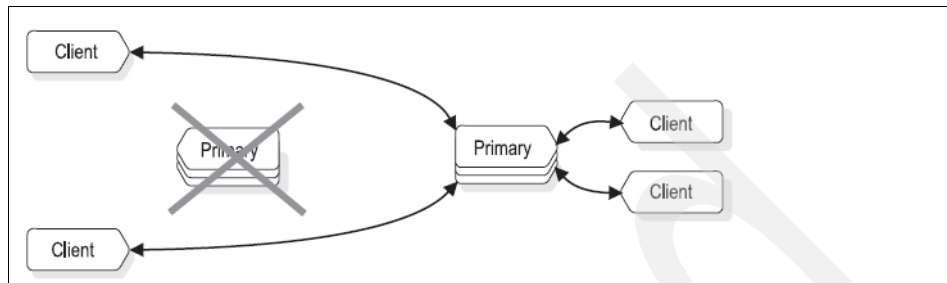


Figure 7-11 Clients redirected to the newly promoted (to become the primary) server

7.6.2 Shared disk secondary server

A shared disk (SD) secondary server shares disk space with a primary server. An SD secondary server does not maintain a copy of the physical database on its own disk space, instead, it shares disks with the primary server.

Figure 7-12 on page 373 depicts an example of shared disk secondary configuration as a primary server with three SD secondary servers. In this case, the role of the primary server can be transferred to any of the other SD secondary servers. This transfer is true whether the primary server needed to be taken out of service because of a planned outage, or because of an unexpected failure of the primary server.

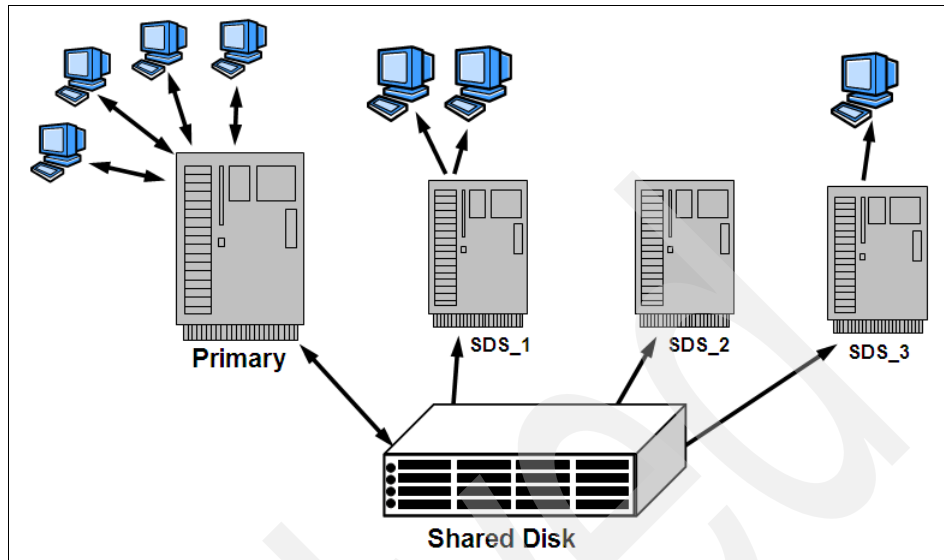


Figure 7-12 Example of SDS setup

Because all the SD secondary servers are reading from the same disk subsystem, they are all equally able to assume the primary server role, as illustrated in Figure 7-13, in which the primary server is offline.

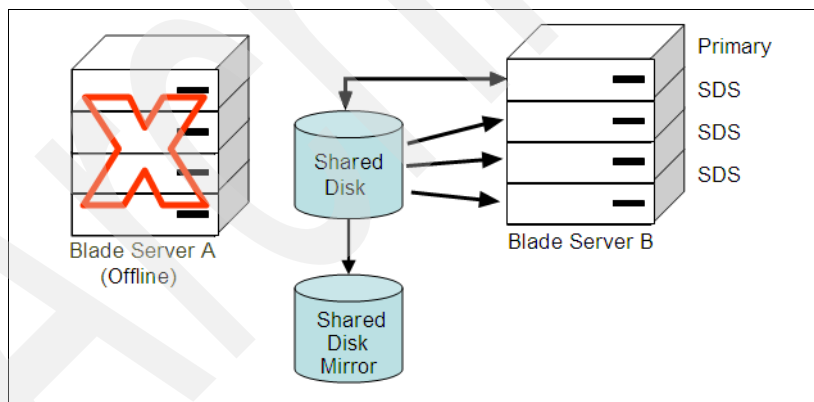


Figure 7-13 Server SDS_1 becomes the primary server when the original primary is offline

You can integrate the SDS functionality easily into existing Informix high availability solutions. RSS, HDR, and ER together in a mixed environment can serve the requirements of database reliability well in a modern IT infrastructure.

An SDS-only setup usually encompasses disk mirroring, as illustrated in Figure 7-14.

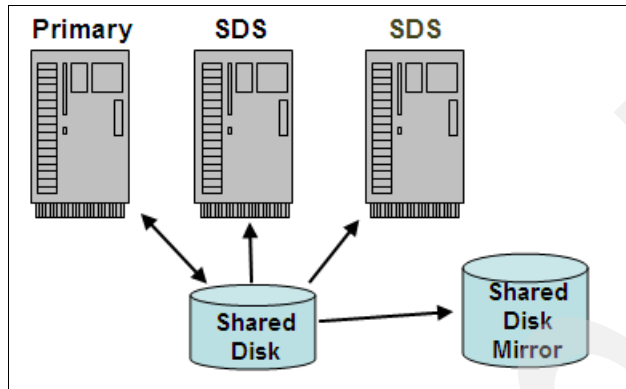


Figure 7-14 SDS setup utilizing disk mirroring

7.6.3 Remote stand-alone secondary server

A *remote stand-alone secondary server* (RSS) is updated asynchronously from the primary server. RSS servers can be geographically distant from the primary server, serving as remote backup servers in disaster recovery scenarios. Each RSS server maintains a complete copy of the database, with updates transmitted asynchronously from the primary server over secure network connections.

Figure 7-15 on page 375 illustrates a configuration that consists of multiple RSS servers. This configuration is useful in a situation where the primary server is located a long distance from the RS secondary servers, or if the network speed between the primary server and the RS secondary server is slow or erratic. Because RS secondary servers use fully duplexed communication protocols, and they do not require checkpoints that are processed in SYNC mode, the additional servers typically do not have a significant effect on the performance of the primary server.

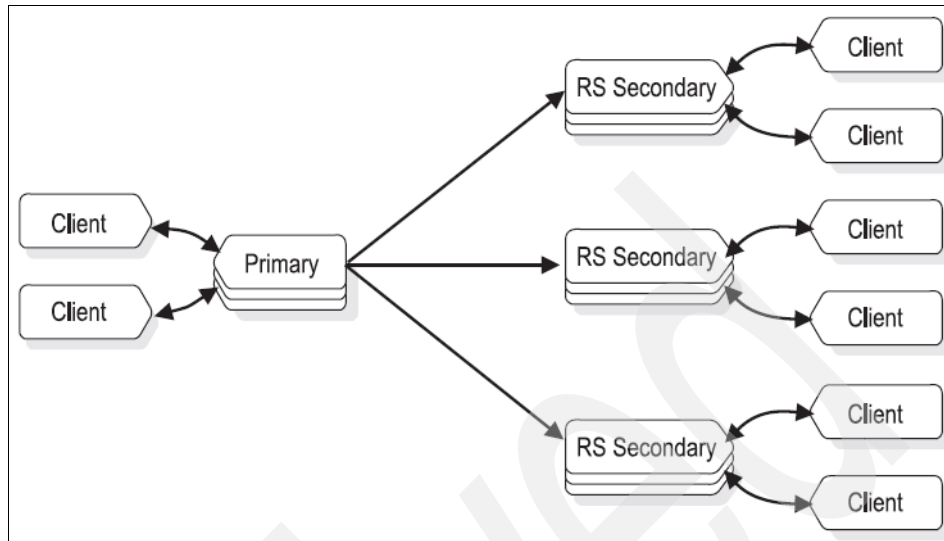


Figure 7-15 Primary server with three RS secondary servers

Figure 7-16 illustrates an example of an RS secondary server and an HDR secondary server configuration. In this example, the HDR secondary server provides high availability while the RS secondary server provides additional disaster recovery if both the primary and HDR secondary servers are lost. The RS secondary server can be geographically remote from the primary and HDR secondary servers so that a regional disruption, such as an earthquake or flood, does not affect the RS secondary server.

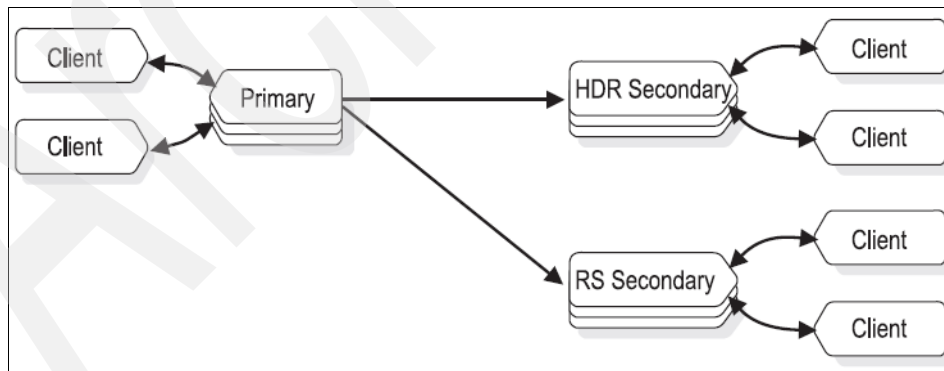


Figure 7-16 Primary server with HDR secondary and RS secondary servers

If a primary database server fails, you can convert the existing HDR secondary server into the primary server, as shown in Figure 7-17 on page 376.

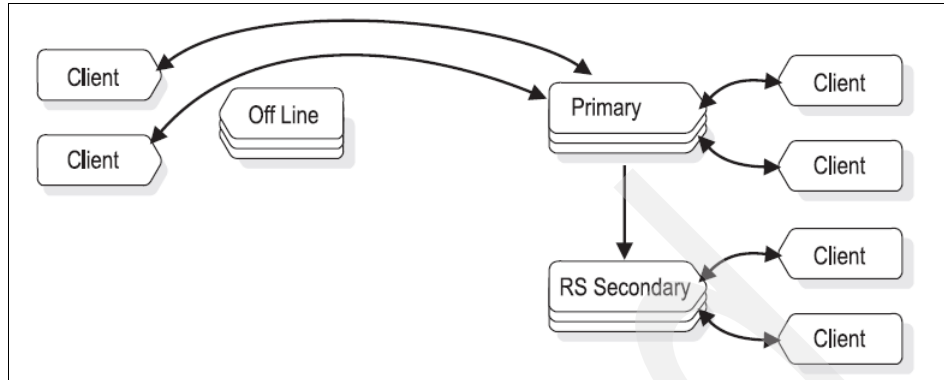


Figure 7-17 Failover of primary server to HDR secondary server

If it appears that the original primary server is going to be offline for an extended period of time, the RS secondary server can be converted to an HDR secondary server. Then, when the original primary server comes back online, it can be configured as an RS secondary server.

In summary, RSS is extremely similar to HDR, especially in setup and behavior. It is a full copy of an instance, and it can help with load balancing. However, this improved functionality allows more than one backup to be available. When comparing the two features, RSS differs from HDR in several distinct ways:

- ▶ The communication protocol is fully duplexed to allow better communication over slower lines.
- ▶ There is no synchronous mode of running RSS servers (including checkpoints).
- ▶ An RSS server cannot be promoted to a primary server.
- ▶ There can be any number of RSS servers.

You can combine any of the previous configurations with ER. For information about HDR secondary servers, RS secondary servers, and SD secondary servers, see the *IBM Informix Dynamic Server Administrator's Guide, v11.50*, SC23-7748.

7.6.4 Enterprise Replication

Enterprise Replication (ER) supports asynchronous data replication over geographically distributed database servers and allows you to replicate both entire databases or subsets of databases and tables. ER captures transactions to be replicated throughout the enterprise. On the source database server, ER reads the logical log and transmits each transaction to the target database

servers. At each target database server, ER receives and applies each transaction to the appropriate databases and tables. You can combine ER with other data replication solutions. The major benefit of this type of data replication is that if a particular database server fails, the replication process can continue and all transactions in the replication system will be committed. In contrast, *synchronous data replication* replicates data immediately when the source data is updated. Synchronous data replication uses the two-phase commit technology to protect data integrity. In a two-phase commit, a transaction is applied only if all interconnected distributed sites agree to accept the transaction. Synchronous data replication is appropriate for applications that require immediate data synchronization. However, synchronous data replication requires that all hardware components and networks in the replication system are available at all times.

The type of replication can be either primary-target or update-anywhere. In primary-target replication, data is replicated from one system to one or more other systems, often for the purpose of data consolidation or distribution. In an update-anywhere replication system, data is replicated among a set of database servers where any of the servers might be updated. This type of replication is often used for workload sharing. An enterprise needs either or both types of replication based on its requirements. Business continuity is always the primary requirement.

ER offers these strengths:

- ▶ ER support for network topologies includes fully connected, hierarchical tree, and a forest of trees.
- ▶ ER supports all built-in IBM Informix data types, as well as extended and user-defined data types.
- ▶ ER operates in local area network (LAN), wide area network (WAN), and combined LAN/WAN configurations across a range of network transport protocols.
- ▶ ER supports the Global Language Support (GLS) feature, which allows IBM Informix products to handle various languages, regional conventions, and code sets.
- ▶ Centralized administration.
- ▶ Ease of implementation.
- ▶ Supports the same network encryption options that you can use with communications between a server and its clients to provide complete data encryption.

For detailed information about how to set up and monitor ER, see *IBM Informix Dynamic Server Enterprise Replication Guide*, v11.50, SC23-7755.

7.6.5 The Connection Manager

Because multi-node high-availability clusters often consist of multiple database servers, client applications can connect to the primary server or to one of many secondary servers. With a large number of servers, it can be difficult to determine which server to connect to. It can be also be difficult to determine which server has sufficient free resources to perform a given task. Additionally, when a server encounters a problem, to which server do you fail over? Connection Manager is a thin, lightweight layer between the client and the database cluster that is designed to address these situations.

Connection Manager dynamically routes client application connection requests to the most appropriate server in a high-availability cluster. Connection Manager connects to each of the servers in the cluster and gathers statistics regarding the type of server, unused workload capacity, and the current state of the server. From this information, the Connection Manager is able to redirect the connection to the appropriate server. In addition, Connection Manager Arbitrator provides automatic failover logic for high-availability clusters. You configure Connection Manager Arbitrator using a configuration file to specify which secondary server will take over the role of the primary server in the event of a failure of the primary server. The Connection Manager can also balance workloads by directing client application connections to the server with the least amount of activity. The Connection Manager performs three roles:

- ▶ Rule-based connection redirection
- ▶ Cluster load balancing
- ▶ Cluster failover

Figure 7-18 on page 379 illustrates client applications issuing connection requests to the Connection Manager. Based on predefined service level agreements (SLAs), the Connection Manager routes the connections to the appropriate server.

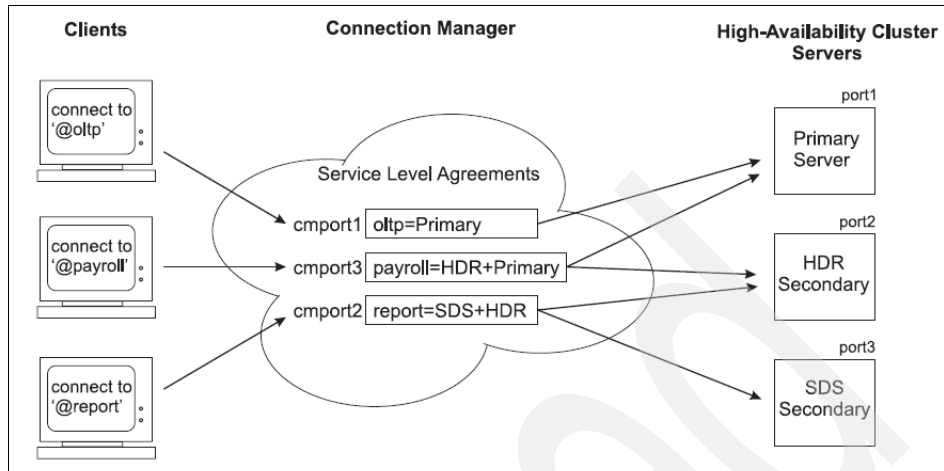


Figure 7-18 Connection Manager setup for routing client connections

You can install and run the Connection Manager on an Informix server instance, or, to isolate the Connection Manager from database server failures, you can install it on a separate, non-Informix machine. In addition, you can configure the Connection Manager to run on a hardware platform other than the machines that make up the cluster. The only requirement is the correct configuration of the `sqlhosts` file to indicate the connection host name and port number.

Important: Configuring multiple Connection Manager instances is especially important to avoid having the Connection Manager become a single point of failure.

For more details about how to set up and monitor the Connection Manager, see the *IBM Informix Dynamic Server Administrator's Guide, v11.50*, SC23-7748.

7.7 Automatic administration and task management

The automation of database administration is a growing area in database technology. Automation allows automatic management, monitoring, and the execution of jobs on a regular basis without manual intervention.

In this section, we provide you with an overview of Informix components that enable you to automate information collection, database maintenance, and the task scheduling of the server in complex systems.

The sysadmin database is automatically created and populated with the objects that are needed to enable automation. You can use the SQL administration API, the Scheduler, and Query Drill-Down functionality to manage automatic maintenance, monitoring, and administration tasks. Refer to the *IBM Informix Dynamic Server Administrator's Guide, v11.50, SC23-7748*, for complete details.

7.7.1 The sysadmin database

The sysadmin database is an internal database that is created automatically for you during the first disk initialization of the Informix instance. The sysadmin database contains the tables that contain and organize the Scheduler tasks and sensors, stores the data that is collected by sensors, and records the results of Scheduler jobs. The SQL administration API functions are also defined in the sysadmin database.

The sysadmin database also contains the following items:

- ▶ The built-in task() function
- ▶ The built-in admin() function
- ▶ The command_history table, which contains information about the commands that the SQL administration API ran

Important: Do not drop or alter the sysadmin database, because it is used by several important database server components.

7.7.2 Administration API

The SQL administration API enables you to perform remote administration using various, specific SQL commands for tasks, such as managing spaces, managing configuration, running routine jobs, and system validation. Because SQL administration API operations occur entirely in SQL, these functions can be used in client tools to administer the database server. The two functions that were designed for that purpose, admin() and task(), are defined in the sysadmin database. You can use EXECUTE FUNCTION statements to invoke the built-in admin() or task() functions to accomplish administration tasks that are equivalent to executing various administration command-line utilities of Informix.

In the EXECUTE FUNCTION statements, which are depicted in Example 7-26 on page 381, items in the argument list specify the utility and its command-line arguments. As in SQL, the SQL statement, which is equivalent to the **oncheck -ce** command, instructs the database server to check the extents.

Example 7-26 Executing admin functions

```
EXECUTE FUNCTION admin('check extents');
```

If you want to increase the virtual memory that the database server can use in an application, the application executes the SQL statement that is depicted in Example 7-27.

Example 7-27 Executing task functions

```
EXECUTE FUNCTION task('add memory', '10 MB');
```

You can view the history of all the SQL administration API functions that were run previously in the `command_history` table in the `sysadmin` database. Example 7-28 shows the sample results in the `command_history` table.

*Example 7-28 A sample result of select * from the command_history table*

```
-----
cmd_number      118
cmd_exec_time   2010-02-24 13:48:57
cmd_user        informix
cmd_hostname     blazer
cmd_executed     scheduler shutdown
cmd_ret_status   0
cmd_ret_msg      Successfully shutdown scheduler

-----
cmd_number      125
cmd_exec_time   2010-03-15 17:16:59
cmd_user        informix
cmd_hostname     NA
cmd_executed     fragment estimate_compression
cmd_ret_status   0
cmd_ret_msg      est  curr  change partnum  table
-----
83.9%  0.0%  +83.9 0x0030015d sample2:"informix".customer

Succeeded: admin_fragment_command('fragment estimate_compressio
n ', '3146077')
-----
cmd_number      127
cmd_exec_time   2010-03-17 14:03:35
cmd_user        informix
cmd_hostname     blazer
cmd_executed     onmode
cmd_ret_status   0
cmd_ret_msg      Checkpoint Completed
```

You can perform the following types of administration tasks with the SQL administration API:

- ▶ Control data compression
- ▶ Update configuration parameters
- ▶ Check data, partition, and extent consistency, control the B-tree scanner, and force and checkpoint
- ▶ Set up and administer ER
- ▶ Set up and administer high-availability clusters
- ▶ Control logging and logical logs
- ▶ Control shared-memory and add buffer pools
- ▶ Control mirroring
- ▶ Control decision-support queries
- ▶ Change the server mode
- ▶ Add, drop, and configure storage spaces
- ▶ Control the SQL statement cache
- ▶ Control and configure SQL tracing
- ▶ Start and stop the listen control threads dynamically
- ▶ Perform other tasks, such as moving the sysadmin database, terminating a session, or adding a virtual processor

For more information about all the SQL administration API commands and the function arguments, see *IBM Informix Dynamic Server Administrator's Reference*, SC23-7749.

7.7.3 The Scheduler

The Scheduler is an administrative tool that enables the database server to execute database functions and procedures at predefined times or as determined internally by the server. The Scheduler manages and executes scheduled tasks for maintenance, monitoring, and administration operations. The set of tasks can be composed of SQL statements. The SQL statements can either collect information or monitor activities and adjust the server.

The Scheduler manages the following items:

- ▶ Tasks, which can run a specific job at a specific time or interval
- ▶ Sensors, which collect and save information
- ▶ Startup tasks, which run only once when the database server starts
- ▶ Startup sensors, which run only once when the database starts

The Scheduler stores information about tasks and sensors in five tables in the sysadmin database (ph_task, ph_run, ph_group, ph_alert, and ph_threshold). You can also add your own tasks and sensors by inserting rows into these tables.

Additionally, you can use the Scheduler to perform these tasks:

- ▶ Associate tasks and sensors into functional groups
- ▶ Track the execution time and return value each time that a task or sensor is run
- ▶ Define alerts with varying severity
- ▶ Define thresholds to control when tasks or sensors are run

Figure 7-19 shows using OpenAdmin Tool (OAT) to view Scheduler tasks.

OpenAdmin Tool for IDS

Server:deen_1150f65@blazer.linux.lbr

Home

Health Center

Alerts

Dashboard

Logs

Admin Command

Online Messages

OnBar Activity

Task Scheduler

Scheduler

Task Details

Task Runtimes

Space Administration

DBSpaces

Chunks

Recovery Logs

Compression

Server Administration

MACH

Configuration

System Validation

Page 1

Task Schedule

515 All

Name	Start Time	Stop Time	Run Frequency	M	T	W	T	F	S	S	Enabled
check_backup	05:00:00	NEVER	1 00:00:00	✓	✓	✓	✓	✓	✓	✓	✓
mon_compression_estimates	02:30:00	NEVER	7 00:00:00	✓	✓	✓	✓	✓	✓	✓	✓
mon_vps		NEVER	0 04:00:00	✓	✓	✓	✓	✓	✓	✓	✓
mon_memory_system		NEVER	0 02:00:00	✓	✓	✓	✓	✓	✓	✓	✓
Auto Update Statistics Evaluation	01:00:00	01:10:00	1 00:00:00	✓	✓	✓	✓	✓	✓	✓	✓
Auto Update Statistics Refresh	01:11:00	05:00:00	1 00:00:00	✗	✗	✗	✗	✗	✓	✓	✓
mon_profile		NEVER	0 04:00:00	✓	✓	✓	✓	✓	✓	✓	✓
mon_users		NEVER	0 04:00:00	✓	✓	✓	✓	✓	✓	✓	✓
Alert Cleanup	02:00:00	NEVER	1 00:00:00	✓	✓	✓	✓	✓	✓	✓	✓
Job Runner		NEVER		✓	✓	✓	✓	✓	✓	✗	✗

Add a New TaskDelete a Task

Figure 7-19 OAT view of the task schedule

You can click a specific task and view or adjust its details, as shown in Figure 7-20 on page 384.

Task Details	
Task Name	check_backup
ID	12
Description	Checks to ensure a backup has been taken of the data server.
Execution Statement	check_backup
Start Time	5 : 0 :00
Stop Time	0 : 0 :00 ☒ NEVER
Frequency	1 Days 0 Hours 0 Minutes
Monday	Enabled
Tuesday	Enabled
Wednesday	Enabled
Thursday	Enabled
Friday	Enabled
Saturday	Enabled
Sunday	Enabled
☒ Enable Task	
Save Cancel	

Figure 7-20 Scheduler task details (OAT view)

7.7.4 Query drill-down

The query drill-down functionality provides statistical information about recently executed SQL statements, enabling you to track the performance of individual SQL statements and analyze statement history.

You can perform query drill-down tasks to gather statistical information about each SQL statement that is executed on the system and to analyze statement history. The query drill-down feature helps you answer these types of questions:

- ▶ How long do SQL statements take?
- ▶ How many resources do individual statements use?
- ▶ How long did statement's execution take?
- ▶ How much time was involved in waiting for each resource?

The statistical information is stored in a circular buffer, which is an in-memory pseudo table called *syssqltrace*, that is stored in the sysmaster database. You can dynamically resize the circular buffer.

By default, this feature is turned off, but you can turn it on for all users or for a specific set of users. When this feature is enabled with its default configuration, the database server tracks the last 1,000 SQL statements that ran, along with the profile statistics for those statements. Because the administration API operations occur entirely in SQL, client tools can use these features to administer the database server.

Example 7-29 illustrates a sample output of an SQL statement history and statistics after the feature was turned on using the Administration task() function:

```
EXECUTE FUNCTION task("set sql tracing on", 1000, 1,"low","global");
```

Example 7-29 SQL statement trace from onstat command

```
# onstat -g his

IBM Informix Dynamic Server Version 11.50.FC6 -- On-Line -- Up 5 days 23:08:08 -- 166912
Kbytes

Statement history:

Trace Level          Low
Trace Mode           Global
Number of traces     1000
Current Stmt ID      3
Trace Buffer size     984
Duration of buffer   228 Seconds
Trace Flags          0x00001611
Control Block        112796028

Statement # 3:       @ 112796060

Database:            0x3000E0
Statement text:
    insert into customer(fname,lname) values ('Ah','Bh')

Iterator/Explain
=====
ID   Left  Right  Est Cost  Est Rows  Num Rows  Partnum Type
1    0     0      1         1         1    3146015 Insert

Statement information:
Sess_id  User_id  Stmt Type  Finish Time  Run Time  TX Stamp  PDQ
10279    101    INSERT    17:52:26    0.0096    69e0c5    0

Statement Statistics:
Page     Buffer   Read     Buffer   Page     Buffer   Write
Read     Read    % Cache  IDX Read  Write   Write   % Cache
```

2	5	60.00	0	0	6	100.00
Lock Requests	Lock Waits	LK Wait Time (S)	Log Space	Num Sorts	Disk Sorts	Memory Sorts
4	0	0.0000	0.000 B	0	0	0
Total Executions	Total Time (S)	Avg Time (S)	Max Time (S)	Avg IO Wait	I/O Wait Time (S)	Avg Rows Per Sec
1	0.0096	0.0096	0.0096	0.0045	0.0090	104.0864
Estimated Cost	Estimated Rows	Actual Rows	SQL Error	ISAM Error	Isolation Level	SQL Memory
3	28	1	0	0	NL	9864

You can also use a Hypertext Preprocessor (PHP)-based web browser administrative tool, the OpenAdmin Tool for Informix, to administer multiple database server instances from a single location. For detailed information, refer to the *IBM Informix Dynamic Server Administrator's Guide, v11.50, SC23-7748*.

7.8 Informix database server security

IBM Informix has many security features that are enabled at various layers of the database server system. In this section, we give a high-level description of the security features that Informix supports and implements at the network, communication, and operating system layers. The database security features include the following types of tasks:

- ▶ Securing server directories
- ▶ Encrypting data across the network
- ▶ Supporting communication protocols
- ▶ Authenticating users
- ▶ Auditing

7.8.1 Server utility and directory security

Informix utilities and product directories are secure, by default. The database server utilities make security checks before the database server starts. When you install a new version of your database server, follow the installation instructions to ensure that the permissions of all key files and directories are set appropriately.

To provide increased security, key server utilities check to determine if your environment is secure. For example, key server utilities check these conditions:

- ▶ The permissions on \$INFORMIXDIR and the directories under it.

- ▶ The permissions on the `onconfig` file.
- ▶ The permissions on the `sqlhosts` file.
- ▶ The length of both the `$INFORMIXDIR/etc/onconfig.std` and `$INFORMIXDIR/etc/$ONCONFIG` filenames must be fewer than 256 characters.

If the tests for any of these conditions fail, the utilities exit with an error message.

You can also use the `DB_LIBRARY_PATH` configuration parameter to control the location from which shared objects, such as external modules, can be loaded.

The `DB_LIBRARY_PATH` configuration parameter allows you to control the location from which shared objects can be loaded, and it allows you to enforce policies and standards on the formats of the `EXTERNAL NAME` clause of the `CREATE FUNCTION`, `CREATE PROCEDURE`, and `CREATE ROUTINE` statements. For more information about the `DB_LIBRARY_PATH` configuration parameter, see *IBM Informix Dynamic Server Administrator's Reference*, SC23-7749.

onsecurity utility

The `onsecurity` utility checks the security of a file, directory, or path. It also troubleshoots the security problems, if any security problems are detected.

Use the `onsecurity` command for one or more of the following purposes:

- ▶ Check whether or not a path leading to a directory or a file is secure
- ▶ Generate diagnostic output that explains the nature of the security problem
- ▶ Generate a script that can be run by the `root` user to remedy the security problems

7.8.2 Network data encryption

Use network encryption to encrypt data that is transmitted between the server and the client, as well as between the server and another server. *Encryption* is the process of transforming data into an unintelligible form to prevent the unauthorized use of the data.

The encryption in communication support modules (CSMs) (ENCCSM) provides network transmission encryption. You can use the CSMs to encrypt data transmissions, including distributed queries, over the network.

You can also implement Secure Sockets Layer (SSL) communications, which encrypt data end-to-end. You can use SSL for encrypted communication with both Distributed Relational Database Architecture (DRDA) and Structured Query Language Interface (SQLI) clients.

Column-level encryption

The granularity of encryption in Informix can be at the column level. All values in a specific column of a database table are encrypted with the same password (word or phrase), the same encryption algorithm, and the same cipher mode. For column-level encryption, you can store the hint outside the encrypted column, rather than repeating it in every row.

You can use column-level encryption to store sensitive data in an encrypted format. After encrypting sensitive data, such as credit card numbers, only users that can provide a secret password can decrypt the data.

Use the built-in `ENCRYPT_AES()` and `ENCRYPT_TDES()` encryption functions to encrypt data in columns containing the following character data types or smart large object data types:

- ▶ CHAR
- ▶ NCHAR
- ▶ VARCHAR
- ▶ NVARCHAR
- ▶ LVARCHAR
- ▶ BLOB (binary large object)
- ▶ CLOB (character large object)

You can also use the `SET ENCRYPTION PASSWORD` statement to set an encryption password for a session. If you use the `SET ENCRYPTION PASSWORD` statement, only users that can provide a secret password can view, copy, or modify encrypted data. Refer to *IBM Informix Dynamic Server Administrator's Reference*, SC23-7749.

7.8.3 User authentication

A user must be authenticated with a security facility by providing a valid user ID and an authentication token (often a password) that match the credentials of a user account on the host computer operating system (OS).

Authenticated users must specify a database to which to connect. A user can perform certain database actions or access certain database objects only if the user has been authorized to perform those tasks by the DBA. Refer to database-specific-level privileges in 7.9.1, “Database-level privileges” on page 392.

Informix provides support for various communication protocols with these authentication modules:

- ▶ Pluggable Authentication Module (PAM) for Informix systems running on UNIX or Linux
- ▶ Lightweight Directory Access Protocol (LDAP) Authentication Support for Windows
- ▶ Informix-specific encryption Communication Support Module (CSM) and simple password CSM for Informix Client Development Software Kit (CSDK) and Generic Security Services Communications Support Module (GSSCSM)

SSL

Secure Sockets Layer (SSL) protocol is another supported communication protocol. SSL uses encryption to provide privacy and integrity for data communication through a reliable end-to-end secure connection between two points over a network.

The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery.

You can use SSL for the following connections:

- ▶ IBM Data Server Driver for Java Database Connectivity (JDBC) and Structured Query Language for Java (SQLJ) connections with an Informix database server
- ▶ IBM Informix ESQL/C connections with an Informix database server
- ▶ IBM Informix ODBC Driver connections with an Informix database server
- ▶ DB-Access connections
- ▶ ER connections
- ▶ High Availability Data Replication (HDR) connections
- ▶ Distributed transaction connections, which span multiple database servers
- ▶ The dbexport, dbimport, dbschema, and dbload utility connections
- ▶ Connection Manager connections between servers in a cluster

You can use the Secure Sockets Layer (SSL) protocol to encrypt data in communications between Informix and DRDA clients.

Network security files

Informix can use these network security files, depending your system configuration and setup:

- ▶ `hosts.equiv`
- ▶ `netrc`
- ▶ `rhosts`
- ▶ `sqlhosts` file (UNIX) and SQLHOSTS registry key (Windows)

The option field of the `sqlhosts` file can contain an 's' option for security value settings. The security options let you control the way that a client (user) gains access to a database server. Table 7-14 shows the possible security option settings in the `sqlhosts` file.

Table 7-14 Security options in the `sqlhosts` file

Setting	Results
s=0	Disables both <code>hosts.equiv</code> and <code>rhosts</code> lookup from the database server side (only incoming connections with passwords are accepted)
s=1	Enables only the <code>hosts.equiv</code> lookup from the database server side
s=2	Enables only the <code>rhosts</code> lookup from the database server side
s=3	Enables both <code>hosts.equiv</code> and <code>rhosts</code> lookup on the database server side (default setting for the database server side)
s=4	Configures a database server to use Pluggable Authentication Modules (PAM)
s=6	Configures ER and high availability connection security
s=7	Configures a database server to support single sign-on (SSO)
r=0	Disables <code>netrc</code> lookup from the client side (no password can be supplied)
r=1	Enables <code>netrc</code> lookup from the client side (default setting for the client side)

7.8.4 Single sign-on authentication

Single sign-on (SSO) is a property of access control of multiple and related but independent software systems. With this property, a user logs in one time and gains access to all systems without being prompted to log in again at each system.

Informix supports SSO security. Informix delivers support for SSO in the Generic Security Services Communications Support Module (GSSCSM) and uses the Kerberos 5 security protocol.

7.8.5 Auditing

Auditing creates a record of selected activities that users perform. Informix conforms to all regulatory compliances and ensures that all necessary details for auditing purposes are provided. Informix provides two major utilities, *onaudit* and *onshowaudit*, for auditing purposes. The *onaudit* utility is used to set the masks that specify the activities to be logged in an audit trail. You can set masks for a particular user, as well. The audit trail report that is generated by Informix is in simple text format. It contains the audit trails of all defined masks for all users. The *onshowaudit* utility is used to read this audit trail report. To make reading the audit trail report easy, you can use several of the options with the *onshowaudit* utility to filter out unnecessary information from the audit trail report.

The audit administrator that analyzes the audit trail can use these records for the following purposes:

- ▶ To detect unusual or suspicious user actions and identify specific users who performed those actions
- ▶ To detect unauthorized access attempts
- ▶ To assess potential security damage
- ▶ To provide evidence in investigations, when necessary
- ▶ To provide a passive deterrent against unwanted activities, as long as users know that their actions might be audited

Refer to the *IBM Informix Security Guide, v11.50*, SC23-7754, for the complete details about all the security aspects of Informix.

7.9 Database-specific security

IBM Informix security extends to database objects, such as tables, routines, roles, and so on. Because database technology comprises interdependent components, Informix allows security to exist on each of these components at each layer in the environment to make a truly secure system. In this section, we discuss the following database-specific components:

- ▶ Database-level privileges
- ▶ Table-level privileges
- ▶ Role privileges
- ▶ Label-based access control (LBAC)
- ▶ Data encryption
- ▶ Other privileges

7.9.1 Database-level privileges

You can use database privileges to control who can access a database. The three levels of database privileges provide the needed access control to the database. Only individual users, not roles, can hold database-level privileges:

- ▶ Connect privilege
- ▶ Resource privilege
- ▶ Database administrator privilege

Connect privilege

Before users can access a database, they must have the Connect privilege. The least of the privilege levels is Connect. Users with the Connect privilege can perform the following functions:

- ▶ Execute the SELECT, INSERT, UPDATE, and DELETE statements, provided that they have the necessary table-level privileges
- ▶ Execute a Stored Procedure Language (SPL) routine, provided that they have the necessary table-level privileges
- ▶ Create views, provided that they are permitted to query the tables on which the views are based
- ▶ Create temporary tables and create indexes on the temporary tables

Resource privilege

The Resource privilege carries the same authorization as the Connect privilege. In addition, users with the Resource privilege are allowed to extend the structure of the database.

Users can create new, permanent tables, indexes, SPL routines, and data types, thus, permanently allocating disk space.

Database administrator privilege

The highest level of database privilege is Database administrator (DBA). When you create a database, you are automatically the DBA of the newly created database. A user with the Database administrator privilege has all the capabilities of the Resource privilege. In addition, the user can perform the following operations:

- ▶ Grant any database-level privilege, including the Database administrator privilege, to another user
- ▶ Grant any table-level privilege to another user or to a role
- ▶ Grant a role to a user or to another role
- ▶ Revoke a privilege

- ▶ Restrict the Execute privilege to DBAs when registering a user-defined routine (UDR)
- ▶ Create any database object
- ▶ Create tables, views, and indexes
- ▶ Designate another user as the owner of any database object
- ▶ Alter, drop, or rename database objects, regardless of who owns them
- ▶ Execute the DROP DISTRIBUTIONS option of the UPDATE STATISTICS statement
- ▶ Execute DROP DATABASE and RENAME DATABASE statements

For expanded details about database-level privileges and the SQL syntax, see the *IBM Informix Guide to SQL: Syntax, v11.50, SC23-7751*.

Important: The sysusers system catalog table describes each set of privileges that are granted on the database. It contains one row for each user or role that has privileges on the database.

7.9.2 Table-level privileges

Table privileges specify which operations a user or role can perform on a table or a view in the database. You can use a synonym to specify the table or the view on which you grant or revoke table privileges. Table 7-15 on page 394 lists the table-level privileges.

Table 7-15 Table-level privileges

Privilege	Effect
INSERT	Lets you insert rows.
DELETE	Lets you delete rows.
SELECT	Lets you access any column in SELECT statements. You can restrict the Select privilege to one or more columns by listing the columns.
UPDATE	Lets you access any column in UPDATE statements. You can restrict the Update privilege to one or more columns by listing the columns.
REFERENCES	Lets you define referential constraints on columns. You must have the Resource privilege to take advantage of the References privilege.
INDEX	Lets you create permanent indexes. You must have the Resource privilege to use the Index privilege. (Any user with the Connect privilege can create an index on temporary tables.)
ALTER	Lets you add or delete columns, modify column data types, add or delete constraints, change the locking mode of the table from PAGE to ROW, or add or drop a corresponding ROW data type for your table.
UNDER	Lets you create subtables under a typed table.

When you create a table in a database that is not American National Standards Institute (ANSI)-compliant, all users have access privileges to the table until you, as the owner of the table, revoke table-level privileges from specific users.

Important: When database-level privileges conflict with table-level privileges, the more restrictive privileges take precedence.

ANSI-compliant databases and non-ANSI-compliant databases differ as to which users are granted table-level privileges by default when a table is created. ANSI standards specify that the database server grants only the table owner (as well as the DBA if the table owner and the DBA are not the same user) any table-level privileges. In a database that is not ANSI compliant, however, privileges are granted to PUBLIC. In addition, the database server provides two table-level privileges, Alter and Index, that are not included in the ANSI standards.

For expanded details about table-level privileges and the SQL syntax for table-level privileges, see the *IBM Informix Guide to SQL: Syntax, v11.50*, SC23-7751.

7.9.3 Other privileges

Informix has additional privileges in various areas:

- ▶ Routine-level privileges, including external user-defined routines (UDRs) and libraries that exist outside the database server.
- ▶ Type-level privileges (*Usage* and *Under*). To control who can use an opaque type, distinct type, or named row type, specify the Usage privilege on the data type. The Under privilege controls whether users can use a typed table as a supertable in an inheritance hierarchy.
- ▶ Sequence-level privileges.
- ▶ Language-level privileges. Informix also supports language-level privileges, which specify the programming languages of UDRs that users who have been granted Usage privileges for a given language can register in the database.
- ▶ Fragment-level privileges. If you use the REVOKE statement to withdraw existing table-level privileges, you can use the GRANT FRAGMENT statement to restore specified table-level privileges to users, roles, or PUBLIC on a subset of the fragments.

7.9.4 Role privileges

A *role* is a database feature that lets the DBA standardize and change the privileges of many users by treating them as members of a class or a group. Privileges that you grant to that role are thereby granted to all users who are currently associated with that role. Think of a role as a work-task classification, such as payroll or a payroll manager. Each defined role has privileges on the database object granted to the role. You use the CREATE ROLE statement to define a role. For example, you can create a role called `news_mes` that grants connect, insert, and delete privileges for the databases that handle company news and messages. When a new employee arrives, you only need to add that person to the role `news_mes`. The new employee acquires the privileges of the role `news_mes`. This process also works in reverse. To change the privileges of all the members of `news_mes`, change the privileges of the role.

Important: Use the GRANT statement to grant privileges. Use the REVOKE statement to revoke privileges.

7.9.5 Label-based access control (LBAC)

Label-based access control (LBAC) is an implementation of multilevel security (MLS) that enables you to control who has read access and who has write

access to individual rows and columns of data. MLS systems process information with separate security levels, permit simultaneous access by users with separate security clearances, and allow users access only to the information for which they have authorization. MLS is a well-known implementation of mandatory access control (MAC). Label-based access control (LBAC) uses a set of security “labels” to control the ability of any user to read, write, delete, or update data in a database. You use these labels to enforce security “policies” that are defined in the database governing data access to the selected tables. These policies are enforced regardless of the method that is used to access the data. For example, with LBAC security, you can prevent users from seeing individual columns or complete rows. For the full details about LBAC and setting it up, see *IBM Informix Security Guide, v11.50*, SC23-7754.

7.10 Informix performance enhancing features

Informix has many features that distinguish it from other RDBM servers. We include a short list in this section to highlight performance enhancing features. For a more comprehensive list of Informix features, go to this website:

http://publib.boulder.ibm.com/infocenter/idshelp/v115/index.jsp?topic=/com.ibm.gsg.doc/ids_gsg_213.htm&resultof=%22new%22%20%22features%22%20%22featur%22

7.10.1 Compression and storage optimization

With IBM Informix, you can compress data and optimize storage in Informix databases. Informix provides full online support for enabling storage optimization and compressing existing table data, while applications continue to use the table. The data compression technology in Informix can help produce up to 80% savings on disk space and I/O improvements of up to 20%. The data compression technology in Informix provides less volume to move around, faster search times, more efficient use of memory, and reduced backup and recovery time.

Compression

Compression is the ability to store data rows in a compressed format on the disk. Compression offers the following advantages:

- ▶ Savings of up to 80% of row storage space, allowing you to fit more data onto a page
- ▶ Ability to estimate possible compression ratio
- ▶ Greater amount of data can be fit into a buffer pool
- ▶ Reduction in logical log usage

- ▶ Fewer I/O operations needed, resulting in faster data scans
- ▶ Fewer I/O in the database backup, resulting in faster backups and restores

Informix offers a way to estimate a compression ratio without performing the compressions. This way, the DBA can identify tables or fragments that are candidates for compression. In Example 7-30, we use the `task()` argument `table estimate_compression` to estimate the compression ratio on the `fedcodes` table in the `sampledb` database.

Example 7-30 Compression estimate indicates 77.7% of space saving

```
> execute function sysadmin:task("table estimate_compression","fedcodes");
```

est	curr	change	partnum	table
77.7%	0.0%	+77.7	0x00500002	sampledb:informix.fedcodes

```
Succeeded: table estimate_compression sampledby:informix.fedcodes
```

```
1 row(s) retrieved.
```

Example 7-30 indicates that the table is a good compression candidate. In Example 7-31, we compress the identified table using the `Admin()` function argument `table compress`.

Example 7-31 Execution of table compress command

```
>execute function sysadmin:task("table compress","fedcodes");
```

```
Succeeded: table compress sampledby:informix.fedcodes
```

```
1 row(s) retrieved.
```

The Informix instance message log also indicates the success of the compression operation, as shown in Example 7-32.

Example 7-32 Message log (online.log) indicating the result of the compress command

```
18:21:02 SCHAPI table compress sampledby:informix.fedcodes succeeded
```

You can monitor the progress of the compression or storage optimization operations with the `onstat -g dsk` command.

Additionally, uncompressing or extracting a compressed table or fragment deactivates compression for new insert and update operations, uncompresses all

compressed rows, deactivates the compression dictionary, and allocates new pages for rows that no longer fit on their original pages.

Storage optimization

Storage optimization refers to the ability to consolidate free space in a table or fragment. Storage optimization offers the following advantages:

- ▶ Consolidated data means better clustering and less I/O.
- ▶ Storage optimization offers the ability to return this free space to the dbspace.
- ▶ Storage optimization offers better space utilization.
- ▶ Any table in the dbspace can use the newly freed space and returned space.
- ▶ Storage optimization allows the space that is saved by compression to be reclaimed from tables and table fragments

Informix provides two storage optimization functions in the Admin API:

- ▶ **repack:**
 - This storage optimization function coalesces all the rows to the front of the partition.
 - This function moves data rows to the available space on data pages in logical page order.
 - This function does not move attached index and partition BLOB pages.
 - A table or fragment is fully accessible.
 - You do not have to compress the table or fragment.
 - The Admin() argument to repack is `table repack`.
- ▶ **shrink:**
 - This storage optimization function returns unused space at the end of the table or fragment back to the dbspace.
 - You cannot shrink the first extent smaller than the initial first extent size that was specified at the table creation.
 - You typically use this function after a repack.
 - You do not have to compress the table or fragment.
 - The Admin() argument to shrink is `table shrink`.

Informix allows you to run compression, repack, and shrink all in one function call. See Example 7-33 on page 399.

```
execute function sysadmin:task("table compress repack shrink","fedcodes");
```

For more details about compression and storage optimization, see the *IBM Informix Dynamic Server Administrator's Guide, v11.50, SC23-7748*.

OAT support for compression and storage optimization

OAT provides the DBA with an easy interface for both compression and storage optimization operations. The DBA can use OAT to view all the tables within a selected database. After the database is selected, OAT selects the tables within that database that might benefit from being compressed and calculates the approximate space savings. If you hold the cursor over the Usage column for a specific table, OAT displays a compression estimate.

7.10.2 Fragmentation

One of the most frequent causes of poor performance in relational database systems is contention for data that resides on a single I/O device. The proper fragmentation of high-use tables can significantly reduce I/O contention.

Fragmentation is a database server feature that allows you to control where data is stored at the table level using a fragmentation strategy. You can logically divide the table and index data into fragments, or partitions. Using one or more fragmentation schemes, you can improve the ability to access several data elements within the table or index in parallel. Fragmentation increases and manages data availability and concurrency. For example, if a sequential read of a fragmented table is required, the sequential read completes more quickly. The sequential read scans the fragments simultaneously rather than reading each disk section separately and serially.

Informix has two major fragmentation schemes that define how data is distributed across the disks fragments. When fragmenting a table, you can specify either scheme:

- ▶ Round robin: Data is evenly distributed across each partition with each new row distributed to the next partition sequentially.
- ▶ Expression-based: Data is distributed into the partitions based on one or more sets of logical rules applied to values within the data. Rules can be *range-based*, using operators, such as "=", ">", "<", "<=", MATCHES, IN, and their inverses, or *hash-based* where the SQL MOD operator is used in an algorithm to distribute data.

Additionally, depending on the data types that are used in the table, You can store individual data columns in a separate kind of data storage space, which is

called a *smart BLOBspace*. The rest of the table's regular data is stored in dbspaces. You can partition indexes using an expression-based partitioning scheme. A table's index partitioning scheme does not need to be the same as the table's data partition scheme. You can place partitioned indexes on a separate physical disk than the data, resulting in optimum parallel processing performance.

7.10.3 Parallelism in Informix

Parallelism in Informix allows multiple threads to process a task simultaneously, taking advantage of multiple resources, such as processors and memory, while managing and maintaining concurrency. The database server can allocate multiple threads to work in parallel on a single query. This feature is known as *parallel database query* (PDQ). PDQ is a database server feature that can improve performance dramatically when the server processes queries that are initiated by decision-support applications. PDQ enables Informix to distribute the work for one aspect of a query among several processors. Informix provides DBAs with PDQ tools for resource management. Informix parallel operations use the virtual processors of the CPU class to run multiple session threads, working in parallel, for a single client. You can apply PDQ to these types of operations:

- ▶ Index building
- ▶ Sorting
- ▶ Recovery
- ▶ Scanning
- ▶ Joining
- ▶ Aggregation
- ▶ Grouping
- ▶ User-defined-routine (UDR) execution
- ▶ Updating statistics
- ▶ Parallel deletes
- ▶ Parallel inserts
- ▶ Parallel backup
- ▶ Parallel restore

For more details about PDQ and how it can help improve your system performance, see the *IBM Informix Dynamic Server Performance Guide, v11.50, SC23-7757*.



Terminology mapping

This appendix compares the terminology that is used in Microsoft SQL Server and Informix database servers.

A.1 SQL Server to Informix terminology comparison

Table A-2 on page 403 shows the existing editions for SQL Server and Informix as of the writing of this book. We cannot map the editions of both database management systems (DBMSs) directly because of the functionality differences. However, the editions in the table match in terms of similar user limitations, memory, and CPU sizing.

For further detailed information about the Informix editions, refer to this website:

<http://www.ibm.com/developerworks/data/library/techarticle/dm-0801doe/index.html>

Table A-1 Available database server editions

SQL Server	Informix
SQL Server 2008 Developer Edition	IBM Informix Dynamic Server Developer Edition
SQL Server 2008 Express Edition	IBM Informix Dynamic Server Express Edition
SQL Server 2008 Workgroup Edition	IBM Informix Dynamic Server Workgroup Edition
SQL Server 2008 Standard Edition	IBM Informix Dynamic Server Workgroup Edition
SQL Server 2008 Enterprise Edition	IBM Informix Dynamic Server Enterprise Edition

Table A-2 on page 403 shows the terminology mapping of physical objects between SQL Server and Informix.

Table A-2 Physical objects

SQL Server	Informix
Filegroup	Dbospace: ▶ Belongs to a database server ▶ Can be shared by objects, such as tables or indexes, that belong to separate databases
Database file with FILEGROW>0	Chunk: ▶ Cooked file or raw device with a static size that is defined at creation time ▶ Belongs logically to a dbospace
Database file with SIZE>0, FILEGROW>0	Chunk: ▶ Cooked file or raw device with a static size that is defined at creation time ▶ Belongs logically to a dbospace
Global allocation map	Chunk free list page.
Page free space pages	Bitmap page for a table or a fragment (which is also named <i>partition</i>).
Shared global allocation map (SGAM)	N/A: ▶ Pages are not shared between separate objects. ▶ Extends and pages are uniquely assigned to one database object.
Block	Extend
Page (8 KB)	Page: ▶ The page size depends on the dbospace definition that is defined at dbospace creation time.

Table A-3 on page 404 shows the terminology mapping of the memory and process level between SQL Server and Informix.

Table A-3 Memory, threads, and processes

SQL Server	Informix
Buffer pool contains pages with 8 KB	Buffer pool: ▶ Buffer size is variable (2, 4, 8, or 16 KB), depending on the buffer pool definition. ▶ The buffer pool maintains data pages from dbspaces (chunks) that are defined with the same page size.
Threads	Threads: ▶ Each database client application is assigned exclusively to a thread in the database server that is serving the incoming tasks based on submitted SQL statements. ▶ There are internally used server threads for I/O, page cleanup, index maintenance, communication, and authentication in the database server.
sqlservr.exe	oninit (.exe): ▶ The database server logically defines a set of processes. Processes are internally referred to as <i>virtual processors</i> (VPs). VPs belong to logical task groups, depending on their duties, such as I/O, authentication, or client application task processing. The number of VPs for certain task groups can be reduced or increased dynamically, depending on workload requirements. ▶ On Windows, the VPs are created as threads, only one oninit process runs for each configured instance. ▶ On UNIX, each VP runs on its own process.

SQL Server	Informix
Process address space	Process private memory on Windows. Shared memory on UNIX systems: ► Memory is divided into three shared segments: resident, virtual, and message. The resident segment contains the buffer pools, log buffers, and internally used structures. The virtual segment contains the session-related memory, such as caches for tables, procedures, and compression. The message segment is an area for client and server communication.

Table A-4 shows the terminology mapping of logical objects between SQL Server and Informix.

Table A-4 Logical objects

SQL Server	Informix
Server	Server
Instance	Instance
Database	Database
Windows registry	Onconfig file
System tables (sys.* tables for database objects)	System catalog in each database (systables)
System tables (sys.* tables for instance monitoring)	Sysmaster database

Table A-5 on page 406 shows the terminology mapping of database objects between SQL Server and Informix.

Table A-5 Database objects

SQL Server	Informix
Schema	Schema
Table	Table
Rule and table constraint	Table constraint
View	View
Index	Index
Transaction log	Logical log
Transaction log dump	logical log backup
Users, groups, and roles	Users and groups (operating system) Roles (database server-maintained)
AdventureWorks sample databases	Stores_demo and superstores_demo demonstration databases that are generated by dbaccessdemo and dbaccessdemo_ud

Table A-6 shows the terminology mapping of administration and usage between SQL Server and Informix.

Table A-6 Administration and usage

SQL Server	Informix
SQL Server Management Studio	Informix OpenAdmin Tool.
SQL Server Configuration Manager	Setnet32 Utility on Windows. The sql hosts file on UNIX.
Tables assigned to a filegroup	Tables or table fragments are assigned to a dbspace.
Database files assigned to filegroups	Chunks are assigned to dbspaces.
N/A	Command-line utilities, such as onstat, onmode, and onspaces, are available on all platforms to maintain and monitor the database by platform-independent scripting.

SQL Server	Informix
System stored procedures	Administration and monitoring functions in the sysadmin and sysmaster databases. Monitoring and administration command-line utilities, such as onmode, onstat, onspaces, and onparams.
Backup database	Database server backup, full, and incremental (level-based), stand-alone or to a storage manager.
Backup log	Backup of the logical log files, stand-alone or to a storage manager.
Restore database	Restore of the database server with all storage objects (full) or on the dbspace level (warm).
Restore log	Restore of the logical logs.
Automatic recovery	Fast recovery.
CHECKPOINT (database-based SQL Server Transact-SQL (T-SQL) statement)	Checkpoint (instance-based) issued by the onmode -c command. The database server writes modified pages for all objects (for example, indexes and tables) from the buffer pool to disk. Modified tables' statistics and status are updated on disk. Physical log (before image area) is reset. Checkpoints are used for starting point for disaster recovery and for high availability synchronization.
Update statistics. Create statistics	Update statistics, low, medium, high (medium and high are sample-based)
BCP (bulk copy program) BULK INSERT	External tables, High-Performance Loader, dbimport/dbexport, dbload, or load/unload statement
SQL Server Query analyzer	"Set explain on" SQL statement in combination with the onstat -g pqs command and Visual Explain
OSQL, SQLCMD	dbaccess

Table A-7 on page 408 shows the terminology mapping of replication topologies between SQL Server and Informix.

Table A-7 Replication concepts

SQL Server	Informix
Transaction replication	Business continuation solutions, such as High Availability Data Replication (HDR), shared disk secondary (SDS), and remote stand-alone secondary (RSS), with one or more read only or updatable hot standby Informix database servers sharing the same or maintaining their own disk space. Various replication types can be combined in a heterogeneous infrastructure in combination with connection management for workload partitioning.
Merge replication	Enterprise Replication (ER), table-based log shipping in an update anywhere or primary target infrastructure. Changes are subject to definition in terms of shipping time. Additional select criteria for selection and projection can be applied to the replicated table.

Data types

In this appendix, we explain the data types in various development environments:

- ▶ Supported SQL data types in C/C++
- ▶ Supported SQL data types in Java
- ▶ Supported SQL data types in ADO.NET
- ▶ Mapping Microsoft SQL Server data types to Informix data types

B.1 Supported SQL data types in C/C++

Table B-1 provides a complete list of SQL data types, C and C/C++ data type mapping, and a quick description of each data type.

For more information about mapping between SQL data types and C and C++ data types, refer to *IBM Informix ESQL/C Programmer's Manual, v3.50*, SC23-9420, which is available for download from this website:

<http://publib.boulder.ibm.com/infocenter/idshe1p/v115/index.jsp>

Table B-1 SQL to C/C++ data type mapping

	SQL data type sqltype	C/C++ type	sqllen	Description
integer	SMALLINT	short	2	▶ 16-bit signed integer ▶ Range between -32,768 and 32,767 ▶ Precision of five digits
	INTEGER INT SERIAL	long	4	▶ 32-bit signed integer ▶ Range between -2,147,483,648 and 2,147,483,647 ▶ Precision of 10 digits
	SERIAL	long	4	▶ 32-bit unsigned integer ▶ Range between 1 and 2,147,483,647 ▶ Precision of 10 digits
	BIGSERIAL SERIAL8	ifx_int8_t	12	▶ Unsigned 64-bit integer ▶ Range between 1 and 9,223,372,036,854,775,807
	BIGINT INT8	long long long __int64 sqlint64	8	▶ 64-bit signed integer ▶ Range between -9,223,372,036,854,775,807 and 9,223,372,036,854,775,807
floating point	REAL FLOAT	float	4	▶ Single precision floating point ▶ 32-bit approximation of a real number ▶ FLOAT(<i>n</i>) can be synonym for REAL if $0 < n < 25$
	DOUBLE DOUBLE PRECISION	double	8	▶ Double precision floating point ▶ 64-bit approximation of a real number ▶ Range in 0, -1.79769E+308 to -2.225E-307, 2.225E-307 to 1.79769E+308 ▶ FLOAT(<i>n</i>) can be synonym for DOUBLE if $24 < n < 54$

	SQL data type sqltype	C/C++ type	sqllen	Description
Decimal	DECIMAL(p,s) DEC(p,s) NUMERIC(p,s) NUM(p,s) MONEY	dec_t	p/2+1	▶ Packed decimal ▶ If precision/scale not specified, default is (5,0) ▶ Maximum precision is 32 digits, and maximum range between -10E31+1 ... 10E31-1 ▶ Consider using char/decimal functions to manipulate packed decimal fields as char data
Date/ Time	DATE	int	4	Integer value similar to UNIX time
	DATETIME	dtime_t	22	N/A
	INTERVAL	intrvl_t	24	
character	CHAR	char	<i>n</i>	▶ Fixed-length character string consisting of <i>n</i> bytes ▶ Use char[<i>n</i>+1] where 1 <= <i>n</i> <= 254 ▶ If length not specified, defaults to 1
	VARCHAR	char	<i>n</i>	▶ Null-terminated variable length character string ▶ Use char[<i>n</i>+1] where 1 <= <i>n</i> <= 254
	LVARCHAR	char	len	▶ Non null-terminated varying character string with 2-byte string length indicator ▶ Use char[<i>n</i>] in struct form where 1 <= <i>n</i> <= 32739
Binary	CLOB(<i>n</i>)	ifx_lo_t		▶ Informix provides a defined interface handling character large objects (CLOBs), such as ifx_lo_open, ifx_lo_read or ifx_lo_write, and ifx_lo_close
	BLOB (TEXT/BYTE)	Locator structure loc_t		▶ Defines the attributes of the binary large object (BLOB), such as file location, file name, file size, and NULL value specifications

B.2 Supported SQL data types in Java

Table B-2 shows the Java equivalent of each SQL data type, based on the Java Database Connectivity (JDBC) specification for data type mappings. The JDBC driver converts the data exchanged between the application and the database using the following mapping schema. Use these mappings in your Java applications and your parameter style Java procedures and user-defined functions (UDFs).

For more information about mapping between SQL data types and Java types, refer to the *Embedded SQLJ User's Guide, Version 2.90*, G251-2278, which is available for download at this website:

<http://www-01.ibm.com/software/data/informix/pubs/library>

Table B-2 SQL data types mapped to Java declarations

	SQL data type sqltype	Java type	sqllen	Description
Integer	SMALLINT	short	2	16-bit, signed integer
	INTEGER	int	4	32-bit, signed integer
	INT8 SERIAL8	long	4	32-bit, signed integer
	BIGINT BIGSERIAL	bigint	8	64-bit, signed integer
floating point	SMALLFLOAT	float	4	Single precision floating point
	FLOAT DOUBLE PRECISION	double	4	Double precision floating point
Decimal	DECIMAL(p,s)	java.math. BigDecimal	n/2	Packed decimal
	MONEY	java.math. BigDecimal	n/2	Packed decimal
Date/ Time	DATE	java.sql.Date	10	10-byte character string
	DATETIME	java.sql.Time	8	8-byte character string
	DATETIME	java.sql. Timestamp	26	26-byte character string
	INTERVAL	lfxIntervalDF	N/A	N/A

	SQL data type sqltype	Java type	sqlen	Description
character	CHAR	String	<i>n</i>	Fixed-length character string of length <i>n</i> , where <i>n</i> is from 1 to 254
	CHAR FOR BIT DATA	byte[]	N/A	Fixed-length character string of length <i>n</i> , where <i>n</i> is from 1 to 254
	VARCHAR LVARCHAR	java.lang.String	<i>n</i>	Variable-length character string, <i>n</i> <= 32,739
	NVARCHAR	java.lang.String	<i>n</i>	Variable-length character string, <i>n</i> <= 32,739
	NCHAR	java.lang.String	<i>n</i>	Variable-length character string, <i>n</i> <= 254
	VARCHAR FOR BIT DATA	byte[]		Variable-length character string
Binary	CLOB(<i>n</i>)	byte[]	<i>n</i>	Large object variable-length character string
	BLOB(<i>n</i>)	byte[]	<i>n</i>	Large object variable-length binary string

B.3 Supported SQL data types in Informix ADO.NET

Table B-3 on page 415 shows the mapping between the Informix SQL data types, .NET Framework, and namespace data types.

Table B-3 Mapping SQL Informix namespace and .NET data types

	lfxType data type	Informix data type	.NET Framework data type	lfxTypes namespace classes and structures
Char	Char	Char	String	lfxString
	Varchar	Varchar	String	lfxString
	LongVarChar	Lvarchar	String	lfxString
Date/Time	Date	Datetime (date prec)	DateTime	lfxDate
	Time	Datetime (time prec)	TimeSpan	lfxTime
	Datetime	Datetime (date and time prec)	DateTime	lfxTimeStamp
LOB data	Blob	BLOB BYTE	Byte[]	lfxBlob
	Clob	CLOB TEXT	String	lfxClob
NUMERIC	Smallint	BOOLEAN, SMALLINT	int16	lfxInt16
	Integer	INT, INTEGER, SERIAL	int32	lfxInt32
	BigInt BigSerial	BIGINT, BIGSERIAL, INT8, SERIAL8	int64	lfxInt64
	Real	REAL, SMALLFLOAT	Single	lfxReal
	Double	DECIMAL (< 29), DOUBLE PRECISION	Double	lfxDouble
	Float	DECIMAL (32), FLOAT	DOUBLE	lfxDouble
	Decimal	DECIMAL	Decimal	lfxDecimal
	Money	MONEY	Decimal	lfxDecimal
	Numeric	DECIMAL(< 29), NUMERIC	Decimal	lfxDecimal

B.4 Mapping SQL Server data types to Informix

Table B-4 summarizes the mapping from SQL Server data types to the corresponding Informix data types. The mapping is one to many and depends on the actual usage of the data.

The table represents the mapping for the built-in data types. Be aware that the Informix database server provides you with the ability to create additional data types (UDT) easily on your own. In addition, the Informix built-in types are enhanced by the data types that are provided with the DataBlade modules that ship with the database server.

Table B-4 Mapping SQL Server data types to Informix data types

	SQL Server data type	Informix data type	Value range
INTEGER	TINYINT	SMALLINT	-32768 to 32767
	SMALLINT	SMALLINT	-32768 to 32767
	INTEGER	INTEGER SERIAL	-2,147,483,648 to 2,147,483,647
	BIGINT	BIGINT BIGSERIAL INT8	-(263 -1) to 263 -1
	TIMESTAMP ROWVERSION IDENTITY	SERIAL BIGSERIAL	N/A
	BIT	BOOLEAN	0,1

	SQL Server data type	Informix data type	Value range
Floating Point	NUMERIC DECIMAL	NUMERIC DECIMAL DECIMAL(p,s) DEC	N/A
	FLOAT REAL	FLOAT DOUBLE PRECISION SMALLFLOAT REAL	N/A
	MONEY SMALLMONEY	MONEY(p,s)	N/A
Date/Time	DATE	DATE	N/A
	DATETIME DATETIME2 SMALLDATETIME	DATETIME year to fraction(<i>n</i>)	N/A
	DATETIMEOFFSE T	DATETIME year to fraction (5)	N/A
	TIME	DATETIME hour to fraction(<i>n</i>)	N/A
Character	CHAR(<i>m</i>)	CHAR(<i>n</i>)	<i>m</i> <= 8,000 <i>n</i> <= 32,768
	VARCHAR(<i>m</i>)	VARCHAR(<i>n</i>) LVARCHAR(<i>l</i>)	<i>m</i> <= 8,000 <i>n</i> <= 255 <i>l</i> <= 32,739
	NCHAR(<i>m</i>)	NCHAR(<i>n</i>)	<i>m</i> <= 4,000 <i>n</i> <= 32,767
	NVARCHAR(<i>m</i>)	NVARCHAR(<i>n</i>)	<i>m</i> <= 4,000 <i>n</i> <= 254
Large (binary) data	BINARY	LVARCHAR(<i>n</i>) BYTE (<i>m</i>)	<i>n</i> <= 32,739 bytes <i>m</i> < 2 GB
	VARBINARY	CLOB(<i>n</i>) BYTE(<i>m</i>)	<i>n</i> < 4 TB <i>m</i> < 2 GB
	IMAGE TEXT	BLOB CLOB	<i>n</i> < 4 TB <i>m</i> < 2 GB

Function mapping

This appendix provides a mapping of functions from SQL Server to Informix. We describe the following functions:

- ▶ Mathematical functions
- ▶ Character and string functions
- ▶ Boolean functions
- ▶ Date and time functions
- ▶ Metadata functions
- ▶ Aggregate functions
- ▶ System functions
- ▶ Security functions
- ▶ Miscellaneous functions

There are functions with the same name in these two database management systems (DBMSs) that behave differently. There are also SQL Server functions that do not have equivalent functions in Informix. You can implement these functions with the various application development languages that are provided by Informix. At the end of this appendix, we provide a few examples.

C.1 Mathematical functions

Table C-1 provides the mapping of mathematical functions from SQL Server to Informix.

Table C-1 Mathematical function mapping for SQL Server to Informix

SQL Server	Informix	Notes
%	MOD	Modulo
ABS	ABS	Absolute Value: Returns the absolute, positive value of the given numeric expression
ACOS	ACOS	Arccosine: Returns the angle in radians whose cosine is the given float expression
ASIN	ASIN	Arcsine: Returns the angle in radians whose sine is the given float expression
ATAN	ATAN	Arctangent of n : Returns the angle in radians whose tangent is the given float expression
ATN2	ATAN2	Arctangent of n and m : Returns the angle in radians whose tangent is between the two given float expressions
CEILING	CEIL	Returns the smallest integer greater than or equal to the given numeric expression
COS	COS	Returns the trigonometric cosine of the given angle (in radians)
COT	N/A	Returns the trigonometric cotangent of the specified angle (in radians)
DEGREES	N/A	Given an angle in radians, returns the corresponding angle in degrees
EXP	EXP	Returns the exponential value of the given float expression
FLOOR	FLOOR	Returns the largest integer less than or equal to the given numeric expression
LOG	LOG	Returns the natural logarithm of the given float expression
LOG10	LOG10	Returns the base-10 logarithm of the given float expression

SQL Server	Informix	Notes
PI	N/A Refer to the example at the end of the appendix	Returns the constant value of PI
POWER	POWER	Returns the value of the given expression to the specified power
RADIANS	N/A	Returns radians given a numeric expression in degrees
RAND	N/A	Returns a random float value between 0 and 1
ROUND(x,y)	ROUND(x,y)	Returns a numeric expression, rounded to the specified length or precision
ROUND(x,y,z)	TRUNC(x,y)	Returns x truncated to y places to the right of the decimal point if y is positive, or to the left of the decimal point if y is zero or negative
SIGN	N/A	Returns the positive (+1), zero (0), or negative (-1) sign of the given expression
SIN	SIN	Returns the trigonometric sine of the given angle (in radians)
SQUARE	N/A	Returns the square of the given expression
SQRT	SQRT	Returns the square root of the given expression
TAN	TAN	Returns the tangent of the input expression

C.2 Character and string functions

Table C-2 on page 422 provides the mapping of character and string functions from SQL Server to Informix.

Table C-2 Character and string function mapping for SQL Server to Informix

SQL Server	Informix	Notes
ASCII	ASCII. Refer to the example at the end of the appendix.	Returns the ASCII code value of the leftmost character of a character expression.
CHAR	N/A Refer to the example at the end of this appendix.	Converts an INT ASCII code to a character.
CHARINDEX	N/A	Returns the starting position of a pattern within a string. Wild cards are not allowed.
DIFFERENCE	N/A	Returns the difference between the SOUNDEX values of two character expressions as an integer.
LEFT	N/A	Returns the part of a character string starting at a specified number of characters from the left.
LEN	LENGTH	Returns the number of characters, rather than the number of bytes, of the given string expression, excluding trailing blanks.
LOWER	LOWER	Returns a character expression after converting uppercase character data to lowercase.
LTRIM	LTRIM	Returns a character expression after removing leading blanks.
NCHAR	N/A	Returns the Unicode character with the given integer code, as defined by the Unicode standard.
PATINDEX	N/A	Returns the starting position of the first occurrence of a pattern in a specified expression, or zeros if the pattern is not found, on all valid text and character data types. Wild cards are allowed.
QUOTENAME	N/A	Returns a Unicode string with the delimiter added to make the input string a valid SQL Server delimited identifier.

SQL Server	Informix	Notes
REPLACE	REPLACE	Replaces all occurrences of the second given string expression in the first string expression with a third expression.
REPLICATE	N/A	Repeats a character expression a specified number of times.
REVERSE	N/A	Returns the reverse of a character expression.
RIGHT	RIGHT	Returns the part of a character string starting from a specified number of characters from the right.
RTRIM	RTRIM	Returns a character string after removing all trailing blanks.
SOUNDEX	SOUNDEX	Returns a four-character (SOUNDEX) code to evaluate the similarity of two strings.
SPACE	N/A	Returns a string of repeated spaces.
STR	N/A	Returns character data converted from numeric data.
STUFF	N/A	Deletes a specified length of characters and inserts another set of characters at a specified starting point.
SUBSTRING	SUBSTRING	Returns part of a character, binary, text, or image expression.
TEXTPTR	N/A	Returns the text-pointer value that corresponds to a text, ntext, or image column in VARBINARY format.
TEXTVALID	N/A	Returns the text-pointer value that corresponds to a text, ntext, or image column in varbinary format.
UNICODE	N/A	Returns the integer value, as defined by the Unicode standard, for the first character of the input expression.
UPPER	UPPER	Returns a character expression with lowercase character data converted to uppercase.

C.3 Boolean functions

Table C-3 provides the mapping of boolean functions from SQL Server to Informix.

Table C-3 Boolean function mapping from SQL Server to Informix

SQL Server	Informix	Notes
ALL	ALL	Compares a scalar value with a single-column set of values.
AND	AND	Combines two Boolean expressions and returns TRUE when both of the expressions are TRUE.
ANY	ANY	Compares a scalar value with a single-column set of values.
BETWEEN	BETWEEN	Specifies an inclusive range to test.
EXISTS	EXISTS	Specifies a subquery to test for the existence of rows.
IN	IN	Determines if a given value matches any value in a subquery or a list.
LIKE	LIKE	Determines whether or not a given character string matches a specified pattern.
NOT	NOT	Negates a Boolean input.
SOME	SOME	Compares a scalar value with a single-column set of values.

C.4 Date and time functions

Table C-4 on page 425 provides the mapping of date and time functions from SQL Server to Informix. You can rewrite the SQL Server built-in functions (such as DATEDIFF, DATEADD, DATENAME, and DATEPART) mostly by the date and date time expressions based on the SQL syntax. For more information about how to handle the date-based and time-based expressions, refer to the *IBM Informix Guide to SQL: Reference, v11.50*, SC23-7750, which is available for download at this website:

<http://www-05.ibm.com/e-business/linkweb/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SC23-7750-04>

Table C-4 Date and time function mapping from SQL Server to Informix

SQL Server	Informix	Notes
DATEADD	Use SQL date expressions	Returns a new date time value based on adding an interval to the specified date
DATEDIFF	Use SQL date expressions	Returns the number of date and time boundaries crossed between two specified dates
DATENAME	Use SQL date expressions	Returns a character string representing the specified date part of the specified date
DATEPART	Use SQL date expressions	Returns an integer representing the specified date part of the specified date
DAY	DAY	Returns an integer representing the day part of the specified date
GETDATE	CURRENT	Returns the current system date and time in the SQL Server standard internal format for date and time values
MONTH	MONTH	Returns an integer that represents the month part of a specified date
YEAR	YEAR	Returns an integer that represents the year part of a specified date

C.5 Metadata functions

Table C-5 on page 426 lists the metadata functions of SQL Server. Informix does not provide such functions; however, you can get this data by querying the Informix system catalog tables. We present the corresponding Informix catalog table and column in the table. *IBM Informix Guide to SQL: Reference, v11.50, SC23-7750*, describes the complete reference about the Informix system catalogs for the databases. The *reference* contains the name of the tables in the catalog set, their columns, their meanings and their data types, and additionally, a description of the commonly used values.

You can obtain the metadata about the Informix database server, current configuration details, memory, threads and sessions, and storage objects, such as dbspaces and chunks, by querying the sysmaster database. This internal database provides the administrator a set of virtual tables in order to access all instance that is information based on SQL statements.

Table C-5 Metadata function mapping from SQL Server to Informix

SQL Server	Informix	Notes
COL_LENGTH	syscolumns table	Returns the defined length of a column.
COL_NAME	syscolumns table	Returns the name of a database column given the corresponding table identification number and column identification number.
COLUMNPROPERTY	syscolumns table	Returns information about a column or procedure parameter.
DATABASEPROPERTY	systables table	Returns the named database property value for the given database and property name.
DB_ID	N/A	Returns the database identification number.
DB_NAME	N/A	Returns the database name.
FILE_ID	N/A	Returns the file identification number (file ID) for the given logical file name in the current database. Use onstat -d from the command line or sysmaster:syschunks for storage object queries.
FILE_NAME	N/A	Returns the logical file name for the given file identification number (ID).
FILEGROUP_ID	N/A	Returns the filegroup identification number (ID) for the given filegroup name.
FILEGROUP_NAME	N/A	Returns the filegroup name for the given filegroup identification number (ID).
FILEGROUPPROPERTY	N/A	Returns the specified filegroup property value when given a filegroup and property name.
FILEPROPERTY	N/A	Returns the specified file name property value when given a file name and property name.
FULLTEXTCATALOG PROPERTY	N/A	Returns information about full-text catalog properties.

SQL Server	Informix	Notes
FULLTEXTSERVICE PROPERTY	N/A	Returns information about full-text service-level properties.
INDEX_COL	sysindexes table, systables table, and syscolumns table	Returns the indexed column name.
INDEXPROPERTY	sysindexes table	Returns the named index property value given a table identification number, index name, and property name.
OBJECT_ID	sysindexes table and systables table	Returns the database object identification number.
OBJECT_NAME	systables, catalog table, depending on the object type, for example sysviews, systriggers, and sysprocedures	Returns the database object name.
OBJECTPROPERTY	systables, catalog table depending on the object type, for example, sysviews, systriggers, and sysprocedures	Returns information about objects in the current database.
TYPEPROPERTY	syscolumns	Returns information about a data type.

C.6 Aggregate functions

Table C-6 on page 428 provides a mapping of aggregate functions from SQL Server to Informix.

Table C-6 Aggregate function mapping from SQL Server to Informix

SQL Server	Informix	Notes
AVG	AVG	Returns the average of the values in a group.
COUNT	COUNT	Returns the number of items in a group.
GROUPING (OLAP)	GROUPING	Causes an additional column to be output with a value of 1 when the row is added by either the CUBE or ROLLUP operator, or 0 when the row is not the result of CUBE or ROLLUP.
MAX	MAX	Returns the maximum value in the expression.
MIN	MIN	Returns the minimum value in the expression.
STDEV	$\text{SQRT}(\text{VAR}(x) * \text{COUNT}(x) / (\text{COUNT}(x) - 1))$	Returns the statistical standard deviation of all values in the given expression.
STDEVP (population)	STDEV	Returns the statistical standard deviation for the population for all values in the given expression.
SUM	SUM	Returns the sum of all the values, or only the DISTINCT values, in the expression.
VAR	$\text{VAR}(x) * \text{COUNT}(x) / (\text{COUNT}(x) - 1)$	Returns the statistical variance of all values in the given expression.
VARP (population)	VARIANCE	Returns the statistical variance for the population for all values in the given expression.

C.7 System functions

Table C-7 provides a mapping of system functions from SQL Server to Informix.

Table C-7 System function mapping from SQL Server to Informix

SQL Server	Informix	Notes
APP_NAME	N/A	Returns the application name for the current session if an application name has been set by the application.
CASE	CASE	Evaluates a list of conditions and returns one of multiple possible result expressions.

SQL Server	Informix	Notes
COALESCE	N/A	Returns the first non-null expression among its arguments.
CURRENT_TIMESTAMP	CURRENT	Returns the current date and time. Equivalent to GETDATE() function.
CURRENT_USER	USER	Returns the current user. Equivalent to USER_NAME() function.
DATALength	LENGTH	Returns the number of bytes used to represent any expression.
FORMATMESSAGE	N/A	Constructs a message from an existing message in sysmessages.
GETANSINULL	N/A	Returns the default nullability for the database for this session.
HOST_ID	N/A	Returns the workstation identification number.
HOST_NAME	select dbinfo ("dbhostname")	Returns the workstation identification number.
IDENT_INCR	SERIAL, BIGSERIAL data type Sequences	Returns the increment value specified during the creation of an identity column in a table or view that has an identity column.
IDENT_SEED	N/A	Returns the seed value specified during the creation of an identity column in a table or a view.
IDENTITY	N/A	Used only in a SELECT statement with an INTO table clause to insert an identity column into a new table. <i>Not</i> the same as the IDENTITY property.
ISDATE	N/A	Determines whether an input expression is a valid date.
ISNULL	NVL	Replaces NULL with the specified replacement value.
ISNUMERIC	N/A	Determines whether an expression is a valid numeric type.

SQL Server	Informix	Notes
NEWID	SERIAL, BIGSERIAL data type Sequences	Creates a unique value of type unique identifier.
NULLIF	NULLIF	Returns a null value if the two specified expressions are equivalent.
PARSENAME	systables table, sysviews table, or other tables depending on the object type	Returns the specified part of an object name (such as object name, owner name, database name, or server name).
PERMISSIONS	systabauth table, syscolauth table, sysusers table, depending on the type of permissions asked for	Returns a value containing a bitmap that indicates the statement, object, or column permissions for the current user.
SESSION_USER	USER	Allows a system-supplied value for the current session's user name to be inserted into a table when no default value is specified.
STATS_DATE	sysdistrib table	Returns the date that the statistics for the specified index were last updated.
SYSTEM_USER	USER	Allows a system-supplied value for the current system user name to be inserted into a table when no default value is specified.
USER_NAME	N/A	Returns a user database user name from a given identification number.

C.8 Security functions

Table C-8 on page 431 provides a mapping of security functions from SQL Server to Informix.

Table C-8 Security function mapping from SQL Server to Informix

SQL Server	Informix	Notes
IS_MEMBER	N/A	Indicates whether the current user is a member of the specified Microsoft Windows NT® group or Microsoft SQL Server role.
IS_SRVROLEMEMBER	N/A	Indicates whether the current user login is a member of the specified server role.
SUSER_ID	N/A	Returns the user's login identification number. Equivalent to SUSER_SID.
SUSER_NAME	USER	Returns the user's login identification name. Equivalent to SUSER_SNAME.
SUSER_SID	N/A	Returns the security identification number (SID) for the user's login name.
SUSER_SNAME	N/A	Returns the login identification name from a user's security identification number (SID).
USER_ID	N/A	Returns a user's database identification number.
USER	sysusers table	Allows a system-supplied value for the current user's database user name to be inserted into a table when no default value is specified.

C.9 Miscellaneous functions

Table C-9 on page 432 provides a mapping of miscellaneous functions from SQL Server to Informix.

Table C-9 Miscellaneous function mapping from SQL Server to Informix

SQL Server	Informix	Notes
CAST	:: Operator	Explicitly converts an expression of one data type to another.
CONTAINSTABLE	N/A	Returns a table of zero, one, or more rows for those columns containing character-based data types for precise or “fuzzy” (less precise) matches to single words and phrases, the proximity of words within a certain distance of one another, or weighted matches.
CONVERT	:: Operator	Explicitly converts an expression of one data type to another.
CURSOR_STATUS	SQL	A scalar function that allows the caller of a stored procedure to determine whether the procedure has returned a cursor and result set for a given parameter.
FREETEXTTABLE	N/A	Returns a table of zero, one, or more rows for those columns containing character-based data types for values that match the meaning but not the exact wording of the text in the specified freetext_string.
OPENQUERY	N/A	Executes the specified pass-through query on the given linked server, which is an Object Linking and Embedding (OLE) DB data source.
OPENROWSET	N/A	Includes all connection information that is necessary to access remote data from an OLE DB data source.

C.10 Implementation of new C-based functions in Informix

If your application requires a function that does not exist in the standard built-in function set of Informix, you can create your own implementation of this function following these steps:

1. Create a C file with your own C code.
2. Compile a relocatable object file with an available C compiler on your system.
3. Create a shared library with the ld utility.

4. Give the library the appropriate permissions and copy the library to a directory according to your definitions.
5. Create an appropriate SQL function definition in your database where you need to use the new function.

We provide an example that shows all these steps based on a 32-bit library built on Sun Solaris. This library provides the new functions `ascii()`, `chr()`, and `pi()` as sample implementations. Based on this sample, you can implement all other missing libraries that are required by your applications.

Example C-1 shows the sample C code for the functions. We used a naming convention of `ifmx<function>` for function names. There is a mapping to the SQL interface name later on in the creation of the function.

Example C-1 Create your own UDF in C programming language within Informix

```
#include <mi.h>
#include <math.h>
#include <stdio.h>

mi_lvarchar *ifmxpi(mi_lvarchar *input, MI_FPARAM *fparam)
{
    mi_lvarchar *RetVal;          /* The return value. */
    mi_char      buffer[20];

    sprintf(buffer, "3.14159265358979");
    RetVal = mi_string_to_lvarchar(buffer);

    /* Return the function's return value. */
    return RetVal;
}

mi_lvarchar *ifmxchr(mi_integer input, MI_FPARAM *fparam)
{
    mi_lvarchar *RetVal;          /* The return value. */
    mi_char      buffer[20];
    sprintf(buffer, "%c", input);
    RetVal = mi_string_to_lvarchar(buffer);

    /* Return the function's return value. */
    return RetVal;
}

mi_lvarchar *ifmxascii(mi_lvarchar *input, MI_FPARAM *fparam)
{
    mi_integer    ret;
    mi_lvarchar *RetVal;          /* The return value. */
```

```

mi_char    buffer[20];
mi_char    *buffer1;

buffer1=mi_lvarchar_to_string(input);
if ( buffer1 )
sprintf(buffer, "%d", buffer1[0]);
RetVal = mi_string_to_lvarchar(buffer);

/* Return the function's return value. */
return RetVal;
}

```

After the implementation of the function, you must compile and link the code into a shared library, as shown in Example C-2. These calls depend on the base operating system and the memory layout. If you use a 64-bit operating system, the calls to the compiler and linker most likely will require separate options for the build. We build the library and copy the library into the `extend` subdirectory of the Informix distribution, which is the default location where all DataBlade module objects reside. Make sure that the library has only read and execute permissions. Informix does not accept the specification of a file with write permissions enabled.

Example C-2 Compile and link a shared library on Sun Solaris 32-bit

```

#Build example for the Library -- Solaris -- 32 Bit !
#We used an Informix Client SDK Version 3.5
#Using former versions change the Include path from dmi to public
cc -xs -I$INFORMIXDIR/incl/dmi -c -o functions.o functions.c
ld -G -o $INFORMIXDIR/extend/mssql/functions.bld functions.o
chmod a+x $INFORMIXDIR/extend/mssql/functions.bld

```

After creating the library, you have to register the library functions in the Informix server. Use a `create function` statement and the specification of the input and output parameters. The location of the new library and the programming language that is used are also required. Example C-3 shows the SQL statements for the registration. You can see that this registration is where the mapping to the final name is made. At the end of the example, we show how to use the functions in a simple SQL statement.

Example C-3 Register user-defined C: UDR with SQL statements in Informix

```

$dbaccess -e stores_demo << EOF
CREATE FUNCTION "informix".ascii(vvarchar(1))
RETURNING varchar(10)
WITH (NOT VARIANT, PARALLELIZABLE)
EXTERNAL NAME "/sqldists/11.50.UC6/extend/mssql/functions.bld(ifmxascii)"
LANGUAGE C

```

```

END FUNCTION;

CREATE FUNCTION "informix".chr(integer)
RETURNING varchar(10)
WITH (NOT VARIANT, PARALLELIZABLE)
EXTERNAL NAME "/sqldists/11.50.UC6/extend/mssql/functions.bld(ifmxchr)"
LANGUAGE C
END FUNCTION;

CREATE FUNCTION "informix".pi()
RETURNING varchar(20)
WITH (NOT VARIANT, PARALLELIZABLE)
EXTERNAL NAME "/sqldists/11.50.UC6/extend/mssql/functions.bld(ifmxpi)"
LANGUAGE C
END FUNCTION;
EOF

#-----Use the functions in SQL statements

Database selected.

select pi( ) from systables where tabid=1;

(constant)
3.14159265358979
1 row(s) retrieved.

select ascii('A' ) from systables where tabid=1;
(constant)
65
1 row(s) retrieved.

select chr(100) from systables where tabid=1;
(constant)
1 row(s) retrieved.
Database closed.

```

Operator mapping

This appendix provides a mapping of operators from SQL Server to Informix. We describe the following operator types:

- ▶ Arithmetic operators
- ▶ Assignment operators
- ▶ String concatenation operators
- ▶ Comparison operators
- ▶ Logical operators
- ▶ Bitwise operators

We highlight the differences between SQL Server operators and Informix operators in **bold**.

D.1 Arithmetic operators

Table D-1 shows a mapping of the arithmetic operators for numeric and date time data types from SQL Server to Informix.

Table D-1 Arithmetic operator mapping from SQL Server to Informix

SQL Server	Informix	Notes
+	+	Addition
-	-	Subtraction
*	*	Multiplication
/	/	Division
%	MOD	Modulo

D.2 Variable assignment and declaration operators

You can use SQL Server Transact-SQL (T-SQL) on the SQL Server for the assignments of values to a variable. Informix does not provide the functionality to use assignments in SQL batches. However, variable declarations and value assignments are part of the Stored Procedure Language (SPL). You must convert T-SQL batches using assignments into a stored procedure. Table D-2 shows a mapping of assignment and declaration operators for variables from SQL Server to Informix SPL.

Table D-2 Assignment operator mapping from SQL Server to Informix

SQL Server	Informix (SPL)	Notes
DECLARE	DEFINE	Defines variables.
SET	LET	Assigns values to variables or a parameter.

D.3 String concatenation operators

Table D-3 on page 439 shows a mapping of the string concatenation operators for string data types from SQL Server to Informix.

Table D-3 String concatenation operator mapping from SQL Server to Informix

SQL Server	Informix	Notes
+	CONCAT or Operator	Returns the concatenation of two string arguments.

D.4 Comparison operators

Table D-4 shows a mapping of the comparison operators for numeric, date time, and string data types from SQL Server to Informix.

Table D-4 Comparison operator mapping from SQL Server to Informix

SQL Server	Informix	Notes
=	=	Is equal to
>	>	Is greater than
>=	>=	Is greater than or equal to
<	<	Is less than
<=	<=	Is less than or equal to
<>	<>	Is not equal to
!=	!=	Is not equal to
!<	>=	Is not less than
!>	<=	Is not greater than

D.5 Logical operators

Table D-5 on page 440 shows a mapping of the logical operators for the results of expressions from SQL Server to Informix.

Table D-5 Logical operator mapping from SQL Server to Informix

SQL Server	Informix	Notes
ALL	ALL	Returns TRUE if all of a set of comparisons are TRUE.
AND	AND	Returns TRUE if both Boolean expressions are TRUE.
ANY	ANY	Returns TRUE if any one of a set of comparisons are TRUE.
BETWEEN	BETWEEN	Returns TRUE if the operator is within a range.
EXISTS	EXISTS	Returns TRUE if a subquery contains any rows.
IN	IN	Returns TRUE if the operator is equal to one of a list of expressions.
LIKE	LIKE	Returns TRUE if the operator matches a pattern.
NOT	NOT	Returns an opposite value of a Boolean expression evaluated.
OR	OR	Returns TRUE if either Boolean expression is TRUE.
SOME	SOME	Returns TRUE if part of a set of comparisons are TRUE.

D.6 Bitwise operators

The SQL Server bitwise operators are implemented in Informix as built-in functions. Table D-6 lists the mappings between the operator and the function.

Table D-6 Bitwise operators mapping from SQL Server to Informix

SQL Server	Informix	Notes
&	bitand()	Bitwise AND
	bitor()	Bitwise OR
^	bitxor()	Bitwise exclusive OR
~	bitnot()	Bitwise NOT

Archived

Administration and monitoring task mapping

Database management systems (DBMSs) provide various administrative facilities to help make the job of the database administrator (DBA) easier. Each database server provides its own solution for server instance maintenance and monitoring. SQL ServerManagementStudio and Informix OpenAdminTool are examples of the easy-to-use GUI tools. To meet the need to embed crucial administration task functionality in scripting and scheduling and for remote execution, database servers provide internal SQL-based monitoring and administration facilities.

In this appendix, we show the mapping of the methods to extract monitoring information and to apply administration tasks, based on SQL Server Transact-SQL (T-SQL) and command-line statements, between SQL Server and Informix 11.50.

E.1 Database administration

Table E-1 shows a mapping by the core database administration duties. We compare the existing SQL command-line facilities and database utilities, focusing on remote execution and task embedding, such as scripting and scheduling.

Table E-1 SQL Server and Informix comparable tasks

Task		SQL Server	Informix
Instance management	Instance start	sqlservr.exe <parameter> net start mssqlserver	oninit(.exe) <parameter> net start <instance_name>
	Instance stop	net stop mssqlserver Shutdown	net stop <instance_name> sysadmin database: execute function task ("onmode", "ky"); Command line: onmode -ky
	Change Instance options automatically without restart	sp_configure	onmode -wf/onmode -wm Change options only in memory or in memory and in the configuration file for persistence.
	Change Instance options manually		Edit the \$ONCONFIG or \$INFORMIXSQLHOSTS file and restart the instance.
	Access the instance	osql sqlcmd	dbaccess

Task		SQL Server	Informix
Database and Space management	Database management	Create database Alter database modify	Create database Rename database
	Space management	Create database Alter database DBCC SHRINKFILE	sysadmin database onspaces Check the various option to add dbspaces and chunks to the instance. Drop an empty dbspace or chunk with onspaces -d Use the sysadmin database for shrinking certain tables: Execute function task ("table shrink")
	Transaction log management	create database alter database DBCC SHRINKFILE	sysadmin database onparams Check the various options to add and drop log files to the instance and modify the physical log space. Dropping an unused log file with onparams -d or using the sysadmin database.
	Access database	use <i>dbname</i>	database <name> connect to <name>
Backup and Restore	Backup	BACKUP DATABASE <i>db_name</i> TO device	onbar -b <backuplevel> ontape -s
	Restore	RESTORE DATABASE <i>db_name</i> FROM device	onbar -r <object> ontape -r
	Data and schema management	N/A	N/A

Task		SQL Server	Informix
Data and schema management	Export a text file from a table	BCP <i>table_name</i> OUT <i>filename</i> ...	dbexport, unload statement in dbaccess High-Performance Loader utility external tables in SQL
	Load a text file into a table	BCP <i>table_name</i> IN <i>filename</i> ...	dbimport or dbload. Load statement in dbaccess external tables in SQL High-Performance Loader
	Generate Data Definition Language (DDL)	sp_help sp_helptext	dbschema Check various options for generating table, views, triggers, and stored procedure schema and generate the list of applied security statements, such as grant and revoke, and role management
Data consistency	Consistency checks	DBCC CHECKCATALOG DBCC CHECKALLOC DBCC CHECKDB DBCC CHECKTABLE	oncheck -cc oncheck -c (DI) oncheck -pe Check oncheck utility for the complete list of check options.
Security	Administer security	grant revoke sp_helpuser sp_addlogin sp_adduser sp_addalias sp_dropalias sp_dropuser sp_droplogin sp_addgroup sp_helpgroup sp_changegroup sp_password	GRANT REVOKE An external security mechanism, such as the operating system, performs all the user authentication setup.
	Kill connected users	KILL <i>spid_number</i>	onmode -z <userid>

E.2 Database monitoring

Continuous monitoring of the database server subsystems, such as disk space, memory, user sessions, and transactions, to set up corrective actions for any detected performance bottlenecks is the second essential task of a DBA. Table E-2 shows the mapping between the SQL Server system views and stored procedures and the Informix command line and SQL interface that generate similar information.

Table E-2 Monitoring interface mapping between SQL Server and Informix

Task	SQL Server 2008	Informix 11.50
Show Instance settings	sp_configure	sysmaster:syscftab onstat -g env
Show database options	sp_helpdb sp_databases sys.databases	sysmaster:sysdatabases database size monitoring with oncheck -pe
Catalog objects	sys.objects sys.columns sys.indexes	<dbname>:systables <dbname>:sysindexes <dbname>:syscolumns <dbname>:sysprocedures and so on catalog tables in each database
Disk space	sys.masterfiles sys.sysfiles sys.database_files DBCC SHOWFILESTATS DBCC EXTENTINFO	sysmaster:syschunks sysmaster:sysdbspaces onstat -d/onstat -D onstat -g iof oncheck -pe
Disk space for tables	sp_spaceused	sysmaster:sysptnhdr oncheck -pt / oncheck -pT
Disk space I/O statistics	sys.dm_io_virtual_file_stats	onstat -g iof onstat -g ioq onstat -D
Display log entries	DBCC LOG	onlog

Task	SQL Server 2008	Informix 11.50
Session information	sys.dm_exec_sessions sys.dm_exec_connections sp_who	sysmaster:syssessions onstat -g sql onstat -g sql <sessid> onstat -g ses onstat -g ses <sessid> onstat -g ath onstat -g sql
Statistics	sp_monitor	sysmaster:sysprofile onstat -p onstat -g cpu / onstat -g glo onstat -g ntt
Locks	sys.dm_tran_locks	sysmaster:syslocks onstat -k
Transactions	sys.dm_tran_active_transactions	sysmaster:systrans onstat -x
Memory	sys.dm_os_memory_nodes sys.dm_os_memory_objects sys.dm_os_memory_pools	sysmaster:syssegments sysmaster:syspoolst onstat -g seg onstat -g afr /onstat -g ffr onstat -g mem
Buffer pool	sys.dm_os_buffer_descriptors	sysmaster:sysbufhdr onstat -B onstat -P
Libraries and symbols	sys.dm_os_loaded_modules	sysmaster:sysssymtab onstat -g dll onstat -g sym
Replication	syspublications syssubscriptions sysarticles	Enterprise Replication (ER) table-based replication onstat -g cdr log shipping-based replication onstat -g dri onstat -g sds onstat -g rss
Threads	sys.dm_os_threads	sysmaster:systhreads onstat -g ath onstat -g act

Database server utilities

In this appendix, we provide a list of the Informix utilities that can help you perform administration tasks and capture information about configuration and performance.

F.1 Informix utilities

Table F-1 lists the database management and configuration utilities that Informix provides. The relevant publication of the Informix documentation set, as shown in the last column of the table, describe these utilities in detail.

Table F-1 Informix utilities

Utility	Description	Where described
archecker	Verifies backups and performs table-level restores.	<i>IBM Informix Backup and Restore Guide, v11.50, SC23-7756</i>
cdr	Controls Enterprise Replication (ER) operations.	<i>IBM Informix Dynamic Server Enterprise Replication Guide, v11.50, SC23-7755</i>
dbexport	Unloads a database into text files for later import into another database and creates a schema file.	<i>IBM Informix Migration Guide, SC23-7758</i>
dbimport	Creates and populates a database from text files. Use the schema file with dbimport to recreate the database schema.	<i>IBM Informix Migration Guide, SC23-7758</i>
dbload	Loads data into databases or tables.	<i>IBM Informix Migration Guide, SC23-7758</i>
dbschema	Creates a file that contains the SQL statements that are needed to replicate a specified table, view, or database, or views the information schema.	<i>IBM Informix Migration Guide, SC23-7758</i>
imcadmin	Starts or stops Informix MaxConnect, or gathers statistics on it.	<i>Guide to Informix MaxConnect, Version 1.1, G251-0577</i>
ism	Manages IBM Informix Storage Manager, storage devices, and media volumes.	<i>IBM Informix Storage Manager Administrator's Guide, v2.2, G229-6388</i>
onaudit	Manages audit masks and auditing configurations.	<i>IBM Informix Security Guide, v11.50, SC23-7754</i>
onbar	Backs up and restores storage spaces and logical logs.	<i>IBM Informix Backup and Restore Guide, v11.50, SC23-7756</i>
oncheck	Checks specified disk structures for inconsistencies, repairs inconsistent index structures, and displays information about disk structures.	<i>IBM Informix Dynamic Server Administrator's Reference, SC23-7749</i>

Utility	Description	Where described
oncmsm	Starts the Connection Manager, which manages and redirects client connection requests based on service level agreements (SLAs) that are configured by the system administrator.	<i>IBM Informix Dynamic Server Administrator's Reference, SC23-7749</i>
ondblog	Changes the logging mode.	<i>IBM Informix Dynamic Server Administrator's Reference, SC23-7749</i>
oninit	Brings the database server online.	<i>IBM Informix Dynamic Server Administrator's Reference, SC23-7749</i>
onload	Loads data that was created with onunload into the database server.	<i>IBM Informix Migration Guide, SC23-7758</i>
onlog	Displays the contents of logical log files.	<i>IBM Informix Dynamic Server Administrator's Reference, SC23-7749</i>
onmode	Changes the database server operating mode and performs various other operations on shared memory, sessions, transactions, parameters, and segments.	<i>IBM Informix Dynamic Server Administrator's Reference, SC23-7749</i>
ON-Monitor	Performs administration tasks using the ON-Monitor menus.	<i>IBM Informix Dynamic Server Administrator's Reference, SC23-7749</i>
onparams	Modifies the configuration of logical logs or physical logs.	<i>IBM Informix Dynamic Server Administrator's Reference, SC23-7749</i>
onpassword	Encrypts and decrypts password files for Enterprise Replication (ER) and Connection Manager.	<i>IBM Informix Dynamic Server Administrator's Reference, SC23-7749</i>
onperf	Monitors database server performance (creates graphs and query trees, and shows status and metrics).	<i>IBM Informix Dynamic Server Performance Guide, v11.50, SC23-7757</i>
onpladm	Writes scripts and creates files that automate data load and unload jobs.	<i>IBM Informix High-Performance Loader User's Guide, v11.50, SC23-9433</i>
onshowaudit	Extracts information from an audit trail.	<i>IBM Informix Security Guide, v11.50, SC23-7754</i>
onspaces	Modifies dbspaces, blobspaces, sbspaces, or extspaces.	<i>IBM Informix Dynamic Server Administrator's Reference, SC23-7749</i>
onstat	Monitors the operation of the database server.	<i>IBM Informix Dynamic Server Administrator's Reference, SC23-7749</i>

Utility	Description	Where described
ontape	Logs, backs up, and restores data.	<i>IBM Informix Backup and Restore Guide, v11.50, SC23-7756</i>
onunload	Unloads data from the database server.	<i>IBM Informix Migration Guide, SC23-7758</i>

SQL limits

This appendix shows the relevant limits of SQL Server 2008 and Informix 11.50. You can find a detailed overview of all Informix limits in the information center at this website:

http://publib.boulder.ibm.com/infocenter/idshep/v115/index.jsp?topic=/com.ibm.adref.doc/ids_adr_0718.htm

G.1 Identifier length limits

Table G-1 lists the identifier limits of SQL Server 2008 and Informix 11.50.

Table G-1 SQL Server 2008 and Informix identifier limits

Item	SQL Server 2008	Informix 11.50
Table name	128	128
Column name	128	128
View name	128	128
Index name	128	128
Constraint name	128	128
Cursor name	128	128
Password for data source access	128	OS-level restricted
SQL variable	128	128
User name	128	32

G.2 Database limits

Table G-2 lists the database limits of SQL Server 2008 and Informix 11.50.

Table G-2 SQL Server 2008 and Informix 11.50 database limits

Item	SQL Server 2008	Informix 11.50
Database size	520 TB	8 PB
Instances per machine	50	255
File groups per database	32,767	2,047 dbspaces
Files per database	32,767	32,767 chunks
Locks	Dynamic (64 bit)	Dynamic
Columns per table	1,024 (small) 30,000 (large)	column type dependent maximum 32,767
Table row size in bytes	8,036	32,767
Columns per index	16	16

Item	SQL Server 2008	Informix 11.50
Indexes per table	Depending on the table definition	Depending on the table definition
Index key [byte]	900	Depending on the page size of the dbspace where the index is stored
Max char() size	8,000	32,767
Max varchar() size	8,000	32,767
Longest SQL statement [byte]	16,777,216	64 KB
Columns per SELECT statement	4,096	Depending on the table definition and maximum statement length
Columns per INSERT statement	4,096	Depending on the table definition and maximum statement length
Nested stored procedure levels	32	Dynamic

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks publications

For information about ordering these publications, see “How to get IBM Redbooks publications” on page 461. Note that several of the documents referenced here might be available in softcopy only.

- ▶ *Migrating from Oracle . . . to IBM Informix Dynamic Server on Linux, Unix, and Windows*, SG24-7730
- ▶ *Microsoft SQL Server to IBM DB2 UDB Conversion Guide*, SG24-6672
- ▶ *Informix Dynamic Server 11: Advanced Functionality for Modern Business*, SG24-7465
- ▶ *Security and Compliance Solutions for IBM Informix Dynamic Server*, SG24-7556
- ▶ *IBM InfoSphere DataStage Data Flow and Job Design*, SG24-7576
- ▶ *Developing PHP Applications for IBM Data Servers*, SG24-7218
- ▶ *Informix Dynamic Server 11: Extending Availability and Replication*, SG24-7488

Other publications

These publications are also relevant as further information sources:

- ▶ *IBM Informix Dynamic Server Administrator's Guide, v11.50*, SC23-7748
- ▶ *IBM Informix Guide to SQL: Reference, v11.50*, SC23-7750
- ▶ *Embedded SQLJ User's Guide, Version 2.90*, G251-2278
- ▶ *IBM Informix Storage Manager Administrator's Guide, v2.2*, G229-6388
- ▶ *IBM Informix Security Guide, v11.50*, SC23-7754
- ▶ *IBM Informix GLS User's Guide*, G229-6373
- ▶ *IBM Informix Dynamic Server Administrator's Reference*, SC23-7749

- ▶ *IBM Informix Guide to SQL: Syntax, v11.50, SC23-7751*
- ▶ *IBM Informix Guide to SQL: Tutorial, G229-6427*
- ▶ *IBM Informix Dynamic Server Performance Guide, v11.50, SC23-7757*
- ▶ *IBM Informix High-Performance Loader User's Guide, v11.50, SC23-9433*
- ▶ *IBM Informix Migration Guide, SC23-7758*
- ▶ *IBM Informix JDBC Driver Programmer's Guide, SC23-9421*
- ▶ *IBM Informix ESQL/C Programmer's Manual, v3.50, SC23-9420*
- ▶ *IBM Data Server Provider for .NET Programmer's Guide, SC23-9848*
- ▶ *IBM Informix Dynamic Server Installation Guide for UNIX, Linux, and Mac OS X, GC23-7752*
- ▶ *IBM Informix DB-Access User's Guide, SC23-9430*
- ▶ *IBM Informix ODBC Driver Programmer's Manual, SC23-9423*
- ▶ *IBM Informix Dynamic Server Installation Guide for Windows, GC23-7753*
- ▶ *Guide to Informix MaxConnect, Version 1.1, G251-0577*
- ▶ *IBM Informix Backup and Restore Guide, v11.50, SC23-7756*
- ▶ *IBM Informix Dynamic Server Enterprise Replication Guide, v11.50, SC23-7755*
- ▶ "Expand transaction capabilities with savepoints in Informix Dynamic Server" by Uday B. Kale in IBM developerWorks, 26 March 2009:
<http://www.ibm.com/developerworks/data/library/techarticle/dm-0903idssavepoints/index.html>

Online resources

These websites are also relevant as further information sources:

- ▶ *Oracle to IBM Informix Dynamic Server Porting Guide:*
<http://www.ibm.com/developerworks/data/library/long/dm-0608marino/>
- ▶ IBM database migration:
<http://www.ibm.com/software/solutions/softwaremigration/dbmigteam.html>
- ▶ IBM Migration Toolkit:
<http://www.ibm.com/software/data/db2/migration/mtk/>
- ▶ IBM Informix Dynamic Server v11.50 Information Center:
<http://publib.boulder.ibm.com/infocenter/idshep/v115/index.jsp>

Educational support

Available from IBM training, we list the newest offerings to support your training needs, enhance your skills, and boost your success with IBM software.

IBM offers a range of training options from traditional classroom to Instructor-Led Online to meet your demanding schedule.

Instructor-Led Online is an innovative learning format where students get the benefit of being in a classroom with the convenience and cost savings of online training.

IBM On-site training is available for groups as small as three or as large as fourteen. Choose from the same quality training that is delivered in classrooms, or customize a course or a selection of courses to best suit your business needs.

Enjoy further savings when you purchase training at a discount with an IBM Education Pack. The IBM Education Pack is an online account, which is a flexible and convenient way to pay, track, and manage your education expenses online.

Check your local Information Management Training and Education website, or check with your training representative for the most recent training schedule.

Table 16 on page 460 lists related educational offerings.

Table 16 Educational offerings

Course code	Course title	Course type
IX13	Informix Structured Query Language	Classroom
3X13	Informix Structured Query Language	Instructor- Led Online
IX22	Informix Dynamic Server Database Administration: Managing and Optimizing Data	Classroom
3X22	Informix Dynamic Server Database Administration: Managing and Optimizing Data	Instructor-Led Online
IX40	Informix Dynamic Server Performance Tuning	Classroom
IX42	Informix Dynamic Server Replication	Classroom
3X42	Informix Dynamic Server Replication	Instructor-Led Online
IX81	Informix Dynamic Server Systems Administration	Classroom
3X81	Informix Dynamic Server Systems Administration	Instructor-Led Online

Descriptions of courses for IT professionals and managers are available at the following web page:

http://www.ibm.com/services/learning/ites.wss/tp/en?pageType=tp_search

Call IBM training at 800-IBM-TEACH (426-8322) for scheduling and enrollment, or visit this website:

<http://www.ibm.com/training>

IBM Professional certification

IBM Information Management Professional Certification is a business solution for skilled IT professionals to demonstrate their expertise to the world. Certification validates skills and demonstrates proficiency with the most recent IBM technology and solutions. Table 17 lists the related certification exam.

Table 17 Professional certification

Exam number	Exam name	Certification title
918	System Administration for IBM Informix Dynamic Server V11	IBM Certified System Administrator - Informix Dynamic Server V11

For additional information about this exam, see this website:

<http://www-03.ibm.com/certify/certs/30001104.shtml>

How to get IBM Redbooks publications

You can search for, view, or download IBM Redbooks publications, IBM Redpapers, web docs, draft publications, and additional materials, as well as order hardcopy IBM Redbooks publications, at this website:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Symbols

.NET application 298
.NET Data Provider 299
.NET Framework application 298
.NET key component 260
87
87
% 106

A

access control 170
access method 15
access privilege 326
Active Directory Helper 20
Active X Data Object 262
admin() 380
administration 3, 5, 13, 15
administration free zone 13
administrative task 326, 330, 339, 347, 366, 380, 449, 451
ADO 145
aggregate 15
aggregate cache 361
AIX 8
ANSI read committed 181
Apache Web Server 13
API (application programming interface) 13, 106, 286
applet 266, 302
application 1, 135, 428
application migration 285
application migration test 287
application programming interface 258, 265
application testing 60
application tuning 287
archchecker 346
archchecker utility 365
architecture 3–4, 260
argument 107
arithmetic expression 131
array 4
asynchronous 8
asynchronous I/O virtual processor 30

attach command 444, 447
attached index 93–94
auditing function 23
auditing utility 391
authentication 404, 446
authorization 38
availability 4–5

B

backup 7, 13, 20, 407, 445
backup and restore 363
backup database 407, 445
backup history 69
BCP 407, 446
big buffer 28
Binary DataBlade 15
BLOB (binary large object) 37, 332, 411
blobspace 37, 345
blobspaces 345
boolean function 424
buffer cache 24
buffer pool 26, 340, 344
built-in cast 73
built-in function 15
bulk copy program 407

C

C 106, 290
C++ 290
cache 360
cache monitoring 361
calibration 54
cartridge 318
catalog table 38, 43, 88, 105
cataloging 16
cdr 346
change management 50
character large object 37, 332, 411
check constraint 80
checkpoint 26, 368
chunk 32, 331–332, 341, 345, 356, 370
circular file 32
C-ISAM® DataBlade 16

- CLI driver 265
- client 105
- client application 21
- Client SDK 43, 261
- CLR 9
- column 110, 123, 127, 432, 454
- column permission 174
- column-level encryption 388
- command line processor 407
- committed read 181
- common table expression 140
- communication architecture 42
- communication protocol 327
- composite index 92
- compression ratio 397
- computed column 71, 82
- concurrency access 29
- concurrency control 177
- concurrent thread 20
- configurable page size 93
- configuration 4, 121
- configuration cost 3
- configuration file 32
- configuration parameter 32, 325
 - 26
 - bufferpool 26
 - db_library_path 387
 - dbservername 343
 - dd_hashmax 361
 - dd_hashsize 361
 - ds_hashsize 361
 - ds_poolsize 361
 - locks 27
 - msgpath 331
 - pc_hashsize 361
 - pc_poolsize 361
 - physbuff 27
 - physfile 33
 - rootpath 71
 - shmvirtsize parameter 28
 - uselascommitted 180–181
 - vpclass 30
- conflict resolution 8
- connect privilege 392
- connecting to the database 288
- connection context 24
- Connection Manager 378
- connectivity configuration 327
- consolidation 8

- constant 131
- constraint 71, 101
- container 403
- continuous log restore 9
- control block 27
- control file 244
- conversion 1, 125, 128
- convert 126, 128–129, 432
- cooked disk space 23
- cooked file 30, 403
- cost-based optimizer 11
- creating table 71
- cursor stability 182
- cutover strategy 50

D

- data compression 11
- data dictionary 360
- data distribution 360
- data encryption 391
- data file 34
- data fragment 94
- data infrastructure 4
- data integration 4
- data integrity 15
- data migration 56
- data migration strategy 50
- data source 269
- data source name 319
- data structure 25
- data type 3, 15
- data type migration 59
- database client environment 258
- database creation 70
- database extraction layer 267
- database migration 53
- database object 38, 49–51, 54, 60, 262, 370
- database physical design 50
- database server alias 329
- database-administrator privilege 392
- database-level privilege 391
- DataBlade 15–16
- DataBlade Developer's Kit 16
- DataBlade module 15–16
- DataBlades
 - Basic Text Search DataBlade 15
 - Binary DataBlade 15
 - C-ISAM DataBlade 16

- Excalibur Text DataBlade 16
- Geodetic DataBlade 16
- Image Foundation DataBlade 16
- Spatial DataBlade 16
- TimeSeries DataBlade 16
- TimeSeries Real-Time Loader 16
- Video Foundation DataBlade 16
- data-distribution cache 28
- DBA 13
- DB-Access utility 343
- dbexport 346
- dbimport 346
- dbload 346
- DBMS (database management system) 1–2
- dbschema 346
- dbspace 36
- DDL (data definition language) 1, 50, 316
- DDL statement 68
- decision support operation 2
- decision support system 325
- decryption 23, 176
- def_table_lockmode 186
- default privilege 70
- degree of parallelism 14
- delimiter 230
- dictionary cache 28
- direct I/O 31
- dirty read 180
- disaster 9–10
- disaster recovery 375
- disk mirroring 370
- disk scan thread 14
- disk space initialization 330
- distinct type 395
- distributed access 251
- distributed relational database architecture 329
- distributed transaction coordinator 20
- DML 74
- drsoctcp 329
- drtlitcp 329
- dynamic scalable architecture 4–5, 7
- dynamic statement 155

E

- ease of use 5
- embedded static SQL 266
- emergency boot file 69
- encrypt_aes 176

- encrypt_tdes 176
- encryption 23, 176, 387
- Enterprise Edition 7
- enterprise javabeans 302
- Enterprise replication 376
- environment variable
 - ansiowner 70
 - classpath 327
 - client_locale 235
 - db_locale 198, 235
 - dbdelimiter 246
 - ifx_def_table_lockmode 186
 - informixdir 327
 - informixserver 327
 - informixsqlhosts 44
 - java_home 327
 - onconfig 32, 327
 - path 327
 - server_locale 235
- ER (enterprise replication) 8, 332, 335, 346, 371, 450
- error log 31
- ESQL/C 259
- esql/c application 294
- Excalibur 16
- Excalibur Text DataBlade 15
- exclusive lock 184
- executable code 24
- explain 407
- export 407, 446
- extensibility 15
- extensible markup language 3
- extent 35–36, 341, 380
- external dbspace 37
- external directive 144
- extshmadd 28

F

- failover 9
- failover arbitrator 10
- fan-out parallelism 11
- fast recovery 363, 368
- fault-tolerant mechanism 368
- file 403, 426
- file system 365
- filegroup 29, 403–404, 406, 426
- force application 446
- foreign key 78, 207

- forest 8
- fragmentation 4
- fragment-level privilege 395
- framework 268
- function 419–421
- functional indexe 92
- functional testing 58

G

- general file descriptor 30
- generator 286
- geodetic 3
- Geodetic DataBlade 16
- global pool 28

H

- HA (high availability) 9, 51
- HDR (high availability data replication) 6, 8–9, 334, 337, 345, 371, 375
- header file 268
- heaps 27
- hierarchical 8
- high availability 9
- high availability cluster configuration 371
- host variable 290
- HPL (high performance loader) 11
- Hypertext Preprocessor 263

I

- I/O contention 399
- IBM classroom training 53
- IBM Data Management Services Center 48
- IBM Informix .NET Provider 261
- IBM Informix Client Software Development Kit 258
- IBM Informix ESQL/COBOL 259
- IBM Informix OLE DB Provider 262
- IBM migration team 51
- IBM Rational Functional Tester 58
- IBM Rational Performance Tester 58
- identity columns 71
- IDS Client SDK 43
- Image Foundation DataBlade 16
- immediate 129
- import 407
- incremental backup 367
- index build 11
- index fragment 94

- index fragmentation 93
- index scan 12
- Indexe 455
- Informix xiv, 4, 15–16
 - Dynamic Server 4
- Informix C-ISAM 16
- Informix CLUSTER option 95
- Informix Connect 43, 258, 287, 294, 319
- Informix DataBlade Developer's Kit 16
- Informix Dynamic Server 2, 5, 7, 67
- Informix Dynamic Server Workgroup Edition 7
- Informix ESQL/C 289
- Informix Geodetic DataBlade 16
- informix group 324
- Informix Image Foundation DataBlade 16
- Informix JDBC 306
- Informix JDBC driver 302, 304, 320
- Informix SDK 319
- Informix Spatial DataBlade 16
- Informix SPL 67, 105
- Informix syntax 77
- Informix TimeSeries DataBlade 16
- Informix TimeSeries Real-Time Loader 16
- informix user 324
- INFORMIXSERVER environment variable 343
- initial configuration 325
- inline directive 145
- inline optimizer 145
- installation method 324
- instance 3, 5, 7–8, 12–13, 20, 24, 134, 138, 405, 444
- integrity constraint 38
- inter process communication 28
- Internal tables 28
- interval checkpoint 12
- ipcs 363
- ism 346
- isolation level 177, 179

J

- J2SE 301
- Java application 301
- Java environment 301
- Java ServerPages 302
- Java virtual machine 260
- JBDC application 320
- JDBC application 301
- JDBC driver 260

job chain 253
join method 143

K

KAIO 23
kernel asynchronous I/O 23, 30

L

language 1, 106
language syntax change 286
latency 9
LBAC (abel-based access control) 12
LBAC (label-based access control) 391
level-0 backup 366–367
level-1 backup 367
library function call 259
lightweight deployment solution 267
Linux 4
load 7, 11, 13, 16, 407, 446
load balancing 376
load stress testing 59
local partitioned index 93
local variable 112
lock mode 71
lock table 27
locking processes 177
log cache 24
log management 172
logging and recovery 363
logging status 39, 364
logical consistency 12
logical database structure 35
logical log 26
logical log backup 32, 365
logical log buffer 26
logical log file 23, 32, 366
logical volume 332

M

mandatory access control 396
master 68
memory 8, 14
memory address space 24
memory grant manager (MGM) 14
memory pool 24
memory segment 330
memory utilization 352

message buffer 27–28
message log file 33, 331
metadata 16, 154, 419, 425
metrics collected 52
Microsoft SQL Server 19, 431
migrated view 60
migration assessment 49
migration preparation 49
migration refresh 56
mirrored chunk 370
mirroring 363
mixed extent 34
model 7, 16, 68
monitoring functionality 12
msdb 68
msdtc.exe 20
msgpath 33
MTK (IBM Migration Toolkit) 50–51, 61, 458
multi-level security 395
multiple concurrent transaction 289
multiprocessor architecture 4

N

named pipe 44, 269
named row type 395
naming convention 50
near-line 9
nested transaction 178
nesting level 106
netrc 390
netstat 363
network communication 23
network encryption 387
node failover 10
non-logging temporary table 89
non-trusted connection 288
NTFS files 332
number sign 87

O

object definition 51
object interface for C++ 262
object-relational 15
ODBC (Open Database Connectivity) 145, 261, 265, 295, 318, 333
OLAP (Online Analytical Processing) 2, 14
OLE DB 145, 265, 432
OLE DB data provider 262

- OLTP 2
- onaudit 347
- ON-Bar 7, 9, 364
- onbar 347, 367
- oncheck 339, 341, 347
- oncmsm 347
- onconfig file 26, 32, 325
- ondblog 347
- ongoing application development 50
- oninit 330, 343, 347
- online analytical processing 2
- online transaction processing system 325
- onload 347
- onlog 347
- onmode 339, 344, 347, 406
- ON-Monitor 347
- onmonitor 332
- onparams 71, 339, 343, 347
- onpassword 348
- onperf 348
- onpladm 348
- onshowaudit 348
- onspaces 332, 339, 345, 348, 406
- onstat 339, 348
- ontape 9, 348, 364
- onunload 348
- opaque 142, 395
- open cursor 160
- open data services 20
- open source 265
- OpenAdmin Tool 13
- optical disk 23, 365
- optimization 15
- optimizer 11–12
- optimizer directive 142
- optional selection 324
- owner naming 70

P

- page 36, 93, 98, 249, 340, 342
- page lock 185
- parallel backup 365
- parallel data query 11
- parallel database query 400
- parallel scan feature 11
- parallelism 11
- parameter 107
- partition definition 210

- partition function 84
- partition scheme 84
- partitioning 8
- pdate statistics 407
- PDQ (Parallel Data Query) 11
- PDQ priority level 11
- performance 3, 5, 24
- performance feature 11
- performance requirement 50
- PERL database interface 264
- permanent table 88
- persisted keyword 82
- PHP 13, 308, 386
- PHP cursor handling 314
- PHP database interface 309–311
- PHP non-persistent connection 312
- PHP persistent connection 312
- physical log 33
- physical log file 23
- pluggable authentication module 69, 261
- port number 327
- post-migration test 59
- precompiled file 268
- primary 9
- primary chunk 370
- primary key 78
- primary-key constraint 78
- privilege 38, 154
- procedure cache 24
- procedure permission 174
- process private memory 405
- promotable lock 182, 184
- proxy server 10
- ps 363
- pseudo-table 39

Q

- quality assurance 54
- query drill-down 384

R

- raw device 30, 403
- raw disk 332
- raw table 87
- read concurrency 180
- real-time 9, 16
- recovery 3, 9–10
- Redbooks Web site 461

- Contact us xvi
- referential constraint 207
- referential integrity 11
- registry key 390
- regulatory requirement 12
- reliability 5
- remote standalone secondary server 371
- repeatable read 182
- replicate 8
- replication 9–10
- repository 286
- resource privilege 392
- restore 9, 13
- restore database 407, 445
- restore space 368
- rich data type 3
- role 431
- role tree 222
- roles privilege 391
- rollback 41
- roll-out 287
- root dbspace 33–34, 71, 331
- row end marker 230
- row lock 185
- RSS (remote standalone secondary) 6, 9, 337, 374
- runtime deployment component 258
- runtime environment 268
- runtime library 259

S

- sar 363
- savepoint 41, 178
- sbspace 37
- sbspaces 345
- scalability 4–5
- scalable architecture 8
- scalar function 110, 432
- scan thread 11
- scheduler 382
- schema 219
- schema information 38
- scroll cursor 188
- SDS (shared disk secondary) 9, 337, 371–372
- search condition 80
- secondary 6, 9
- secondary data file 31
- secondary instance 10
- secondary server 10

- section object 28
- security 13, 44, 60, 312, 325, 386, 391
- security feature 12, 386
- security planning 50
- semantic error 143
- sequence 101
- server 4
- server_locale 198
- servlet 302
- Session data 28
- session monitoring 358
- session pools 27
- session thread 21
- setnet32 utility 329
- shared library 365
- shared lock 182, 184
- shared memory 24, 39, 44, 330–331, 339, 343–344, 347, 351–352, 405, 451
 - message portion 25
 - virtual extension portion 25
- shared memory communication 23
- shared memory segment 352
- shared memory structure 39
- shared-memory header 25
- shmadd 28
- simple blob space 37
- single poll thread 23
- single sign-on security 390
- singleton connect 288
- smartblob space 37
- SMP 11
- snowflake 8
- sort pools 27
- sorting pool 28
- sp_addalias 446
- sp_addgroup 446
- sp_addlogin 446
- sp_adduser 446
- sp_changegroup 446
- sp_dropalias 446
- sp_droplogin 446
- sp_dropuser 446
- sp_helpgroup 446
- sp_helpuser 446
- sp_password 446
- space management 172
- spatial 3
- Spatial DataBlade 16
- SPL routine cache 28

- SPL routine caches 27
- SQL 1, 7–8, 11–13, 19–20, 68, 74, 401–402, 406–407, 419–421, 437–439, 453–454
- SQL administration API 380
- SQL agent 20
- SQL function 301
- SQL query change 287
- SQL Server 20
- SQL statement cache 27–28, 39
- sqladhelp.exe 20
- sqlagent.exe 20
- SQLCA (SQL communication area) 292–293
- SQLCODE 146, 189
- sqlexec thread 21
- sqlhosts file 327
- SQLHOSTS registry key 327
- sqlhosts registry key 44
- sqlhosts.std 328
- SQLj 301
- stack 27
- standard table 87
- standards compliance 54
- startserver 444, 447
- statistical information 11
- status control 155
- stmt_cache 361
- stmt_cache_hits 361
- stmt_cache_nolimit 361
- stmt_cache_numpool 361
- stmt_cache_size 361
- storage configuration 332
- storage device 30
- storage manager 365
- storage volumes 365
- storage-space backup 32, 343
- stored procedure 60, 106, 266, 292, 301, 432, 455
- stored procedure call 305
- stored procedure result set 314
- stream pipe 44
- structure 154
- structured query language 68
- synchronous 9
- synchronous data replication 371
- synonym 101, 105
- syntactic error 143
- sysadmin database 69, 380
- syscolauth 174
- sysmaster 350
- sysmaster database 68, 385

- sysprocauth 174
- systabauth 174
- system catalog 27, 38, 144, 360
- system catalog table 38
- System data structure 24
- system monitoring interface 38, 339
- system-monitoring interface 68, 350
- sysuser database 69
- sysutils database 69
- sysxdttypeauth 174

T

- table fragmentation 209
- table function 110
- table partition 209
- table partitioning 11
- table permission 174
- table reorganization 184
- table space 36
- table-level privilege 391
- table-level restores 346
- tables 16
- tablespace 35
- task() 380
- Tcl/Tk programming environment 264
- TCP/IP 269
- tempdb 68
- temporary dbspace 37
- temporary table 87
- test phase 56
- testing 57
- testing consideration 59
- testing methodology 57
- testing process 57
- text 422–423, 426, 432, 446
- text search 15
- Text Search DataBlade 15
- thread 11, 304, 340
- Thread data 28
- threads 20
- time-series 3
- TimeSeries DataBlade 16
- TimeSeries Real-Time Loader 16
- TLI 23
- tool 1
- tool selection 50
- top 363
- total cost of ownership 3

transaction 40
transaction log 20, 406
transaction logging 88
transactions 447
transport layer interface 23
trigger 60, 121, 126, 129, 155
T-SQL 106, 145, 155
type 4 266

U

UDF (user-defined function) 60, 292, 301
UDR (user-defined routine) 12, 105, 288, 362
unbuffered logging 70
uniform extent 34
unique constraint 79
unique keyword 42
unit of work 266
units of work 26
UNIX 4
unqualified column wildcard 131
update 12
update cursor 182
update statistic 407
use command 445
user authentication 288, 386
user defined type 15
user education 287
user mode scheduler 20
user session 447
user thread 21
user-built applications 284
user-defined C structure 290
user-defined data 3
user-defined data type 260
user-defined function 266

V

variable 131, 134, 290, 438
video 3, 16
Video Foundation DataBlade 16
view 127–128, 134, 154, 218
virtual portion 27
virtual process 404
virtual processor 11, 14, 20, 344, 354–355
vmstat 363
volume group 210
volume testing 59

W

warm restore 69
watermarks 356
whenever statement 292
wildcard parameter 106
Windows 4, 8
write operation 10

X

XML 3
XML index 96



Migrating from Microsoft SQL Server to IBM Informix

(0.5" spine)
0.475" <-> 0.875"
250 <-> 459 pages



Migrating from Microsoft SQL Server to IBM Informix

Develops a data and applications migration methodology

Migrates step-by-step from SQL Server to IBM Informix

Provides a variety of migration examples

In this IBM Redbooks publication, we discuss considerations, and describe a methodology, for transitioning from Microsoft SQL Server 2008 to IBM Informix. We focus on the topic areas of data, applications, and administration, providing information about the differences in features and functionality, including the data types, data manipulation language, data definition language, and stored procedures. Understanding the features and functionality of the two products assists you in developing a migration plan.

We provide a conversion methodology and discuss the processes for migrating the database objects and data from SQL Server to Informix using various methods. We show the SQL differences between SQL Server and Informix and illustrate, with examples, how to convert tables, views, stored procedures, functions, and triggers. We provide script conversion samples for data loading. We describe application programming and conversion considerations. In addition, we discuss the Informix configuration, as well as the administration features and functions Informix provides to help DBAs manage the Informix database server after it is migrated.

With this information, you can develop your required transition methodology, and you can plan and execute the conversion activities in an orderly and cost-effective manner.

**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