Getting Started

with Informix Extended Parallel Server

Version 8.3
December 1999
Part No. 000-6541

Published by Informix® Press

Informix Corporation
4100 Bohannon Drive
Menlo Park, CA 94025-1032

Documentation Team: Linda Briscoe, Erin Cizina, Evelyn Eldridge, Laura Kremers, Barbara Nomiyama, Judith Sherwood

Table of Contents

Chapter 3 Getting Up and Running with the Database Server

Introduction

In This Introduction

This Introduction provides an overview of the information in this manual and describes the conventions it uses.

About This Manual

This manual provides an overview of how to work efficiently with Informix Extended Parallel Server. It outlines Extended Parallel Server architecture, introduces the major features, and discusses associated Informix client and application programming interface (API) products and manuals. It also summarizes the basic tasks that are required for you to get up and running with the database server and contains information to help you use the documentation for Extended Parallel Server, Version 8.3.

Types of Users

This manual is written for the following users:

- Database server administrators
- Database server operators
- Database administrators
- Programmers and client application developers
- Database users

This manual assumes that you have the following background:

- Familiarity with structured query language (SQL)
- A working knowledge of your computer, your operating system, and the utilities that your operating system provides
- Some experience working with relational databases or exposure to database concepts
- Some experience with computer programming
- Some experience with database server administration, operating-system administration, or network administration
- Familiarity with data warehousing and data marts, as well as on-line transaction processing (OLTP) and decision-support system (DSS) operations

If you are unfamiliar with relational database concepts or SQL, review the *Informix Guide to SQL: Tutorial* and the *Informix Guide to Database Design and Implementation*. For additional titles, see Chapter 4, "Using the Documentation."

Software Dependencies

This manual assumes that you are using Informix Extended Parallel Server, Version 8.3, as your database server.

Tip: *This version of the product does not support Windows NT.*

Assumptions About Your Locale

Informix products can support many languages, cultures, and code sets. All culture-specific information is brought together in a single environment, called a Global Language Support (GLS) locale.

This manual assumes that you use the U.S. 8859-1 English locale as the default locale. The default is **en_us.8859-1** (ISO 8859-1) on UNIX platforms. This locale supports U.S. English format conventions for dates, times, and currency, and also supports the ISO 8859-1 code set, which includes the ASCII code set plus many 8-bit characters such as é, è, and ñ.

If you plan to use nondefault characters in your data or your SQL identifiers, or if you want to conform to the nondefault collation rules of character data, you need to specify the appropriate nondefault locale.

For information on how to specify a nondefault locale, additional syntax, and other considerations related to GLS locales, see the *Informix Guide to GLS Functionality*.

Demonstration Databases

The DB-Access utility, which is provided with your Informix database server products, includes one or more of the following demonstration databases:

- The **stores_demo** database illustrates a relational schema with information about a fictitious wholesale sporting-goods distributor. Many examples in Informix manuals are based on the **stores_demo** database.
- The **sales_demo** database illustrates a dimensional schema for data-warehousing applications.

The scripts that you use to install the demonstration databases reside in the **$INFORMIXDIR/bin** directory on UNIX platforms and in the **%INFORMIXDIR%\bin** directory in Windows environments.

For information about how to create and populate the demonstration databases, see the *DB-Access User's Manual*. For descriptions of the databases and their contents, see the *Informix Guide to SQL: Reference*. For conceptual information about dimensional data modeling, see the *Informix Guide to Database Design and Implementation*.

New Features

For a comprehensive list of new database server features, see the release notes. For information on how to access the release notes, see "Documentation Notes, Release Notes, Machine Notes" on page 10 of this Introduction.

Documentation Conventions

This section describes the conventions that this manual uses. These conventions make it easier to gather information from this and other volumes in the documentation set.

The following conventions are discussed:

- Typographical conventions
- Icon conventions

Typographical Conventions

This manual uses the following conventions to introduce new terms, illustrate screen displays, describe command syntax, and so forth.

Convention	Meaning
KEYWORD	All primary elements in a programming language statement (keywords) appear in uppercase letters in a serif font.
italics ***italics*** *`italics`*	Within text, new terms and emphasized words appear in italics. Within syntax and code examples, variable values that you are to specify appear in italics.
boldface ***boldface***	Names of program entities (such as classes, events, and tables), environment variables, file and pathnames, and interface elements (such as icons, menu items, and buttons) appear in boldface.
`monospace` *`monospace`*	Information that the product displays and information that you enter appear in a monospace typeface.
KEYSTROKE	Keys that you are to press appear in uppercase letters in a sans serif font.
♦	This symbol indicates the end of one or more product- or platform-specific paragraphs.
→	This symbol indicates a menu item. For example, "Choose **Tools→Options**" means choose the **Options** item from the **Tools** menu.

Tip: *When you are instructed to "enter" characters or to "execute" a command, immediately press* RETURN *after the entry. When you are instructed to "type" the text or to "press" other keys, no* RETURN *is required.*

Icon Conventions

Throughout the documentation, you will find text that is identified by several different types of icons. This section describes these icons.

Comment Icons

Comment icons identify three types of information, as the following table describes. This information always appears in italics.

Icon	Label	Description
⚠	***Warning:***	Identifies paragraphs that contain vital instructions, cautions, or critical information
⇨	***Important:***	Identifies paragraphs that contain significant information about the feature or operation that is being described
💡	***Tip:***	Identifies paragraphs that offer additional details or shortcuts for the functionality that is being described

Cross-Reference Icons

Cross-reference icons indicate paragraphs that show where you can find more information about a topic.

Icon	Description
	Identifies paragraphs that contain cross-references to other Informix manuals that provide additional information on a topic

Feature, Product, and Platform Icons

Feature, product, and platform icons identify paragraphs that contain feature-specific, product-specific, or platform-specific information.

Icon	Description
GLS	Identifies information that relates to the Informix Global Language Support (GLS) feature

These icons can apply to an entire section or to one or more paragraphs within a section. If an icon appears next to a section heading, the information that applies to the indicated feature, product, or platform ends at the next heading at the same or higher level. A ♦ symbol indicates the end of feature-, product-, or platform-specific information that appears within one or more paragraphs within a section.

Additional Documentation

For additional information, you might want to refer to the following types of documentation:

- On-line manuals
- Printed manuals
- Error message documentation
- Documentation notes, release notes, and machine notes
- Related reading

For an overview of the manuals in the Extended Parallel Server documentation set, see Chapter 4, “Using the Documentation.”

On-Line Manuals

An Answers OnLine CD that contains Informix manuals in electronic format is provided with your Informix products. You can install the documentation or access it directly from the CD. For information about how to install, read, and print on-line manuals, see the installation insert that accompanies Answers OnLine.

Informix on-line manuals are also available on the following Web site:

```
www.informix.com/answers
```

Printed Manuals

To order printed manuals, call 1-800-331-1763 or send email to moreinfo@informix.com. Please provide the following information when you place your order:

- The documentation that you need
- The quantity that you need
- Your name, address, and telephone number

Error Message Documentation

Informix software products provide ASCII files that contain all of the Informix error messages and their corrective actions.

To read error messages and corrective actions, use one of the following utilities.

Utility	Description
finderr	Displays error messages on line
rofferr	Formats error messages for printing

Instructions for using the preceding utilities are available in Answers OnLine. Answers OnLine also provides a listing of error messages and corrective actions in HTML format.

Documentation Notes, Release Notes, Machine Notes

In addition to printed documentation, the following sections describe the on-line files that supplement the information in this manual. Please examine these files before you begin using your database server. They contain vital information about application and performance issues.

The following on-line files appear in the **$INFORMIXDIR/release/en_us/0333** directory.

On-Line File	Purpose
STARTDOC_8.3	The documentation notes file for your version of this manual describes topics that are not covered in the manual or that were modified since publication.
SERVERS_8.3	The release notes file describes feature differences from earlier versions of Informix products and how these differences might affect current products. This file also contains information about any known problems and their workarounds.
XPS_*x.y*	The machine notes file describes any special actions that you must take to configure and use Informix products on your computer. Machine notes are named for the product described.

Related Reading

The following publications provide additional information about the topics that this manual discusses.

To learn more about SQL, consider the following books:

- *A Guide to the SQL Standard* by C. J. Date with H. Darwen (Addison-Wesley Publishing, 1997)
- *Understanding the New SQL: A Complete Guide* by J. Melton and A. Simon (Morgan Kaufmann Publishers, 1993)

For additional technical information on database management, consult the following books:

- *Object-Relational DBMSs: The Next Great Wave* by Michael Stonebraker with Dorothy Moore (Morgan Kaufmann Publishers, Inc., 1996)
- *An Introduction to Database Systems* by C. J. Date (Addison-Wesley Publishing, 1995)
- *Transaction Processing: Concepts and Techniques* by Jim Gray and Andreas Reuter (Morgan Kaufmann Publishers, Inc., 1993)

To learn more about fundamental concepts and approaches to database design, consult the following books:

- *Database Modeling and Design, The Entity-Relationship Approach* by Toby J. Teorey (Morgan Kauffman Publishers, Inc., 1998)
- *Handbook of Relational Database Design* by Candace C. Fleming and Barbara von Halle (Addison-Wesley Publishing Company, 1989)

To learn advanced concepts and techniques of dimensional data modeling, refer to the following books:

- *The Data Warehouse Toolkit* by Ralph Kimball (John Wiley & Sons, Inc., 1998)
- *Building the Data Warehouse* by W.H. Inmon (John Wiley & Sons, Inc., 1996)

This manual assumes that you are familiar with your computer operating system. If you have limited UNIX system experience, consult your operating-system manual or a good introductory text before you read this manual.

The following texts provide a good introduction to UNIX systems:

- *Learning the UNIX Operating System* by G. Todino, J. Strang, and J. Peek (O'Reilly & Associates, 1997)
- *A Practical Guide to the UNIX System* by M. Sobell (Benjamin / Cummings Publishing, 1994)
- *UNIX System V: A Practical Guide* by M. Sobell (Benjamin / Cummings Publishing, 1995)

Compliance with Industry Standards

The American National Standards Institute (ANSI) has established a set of industry standards for SQL. Informix SQL-based products are fully compliant with SQL-92 Entry Level (published as ANSI X3.135-1992), which is identical to ISO 9075:1992. In addition, many features of Informix database servers comply with the SQL-92 Intermediate and Full Level and X / Open SQL CAE (common applications environment) standards.

Informix Welcomes Your Comments

Let us know what you like or dislike about our manuals. To help us with future versions of our manuals, we want to know about any corrections or clarifications that you would find useful. Include the following information:

- The name and version of the manual that you are using
- Any comments that you have about the manual
- Your name, address, and phone number

Write to us at the following address:

Informix Software, Inc.
SCT Technical Publications Department
4100 Bohannon Drive
Menlo Park, CA 94025

If you prefer to send electronic mail, our address is:

doc@informix.com

The **doc** alias is reserved exclusively for reporting errors and omissions in our documentation.

We appreciate your suggestions.

Chapter

1 Architecture of Extended Parallel Server

In This Chapter

This chapter introduces the architecture of Informix Extended Parallel Server. For information about specific features, see Chapter 2, "Features of Extended Parallel Server." For information about supported Informix products and client applications, see "Checking Product Compatibility" on page 3-7.

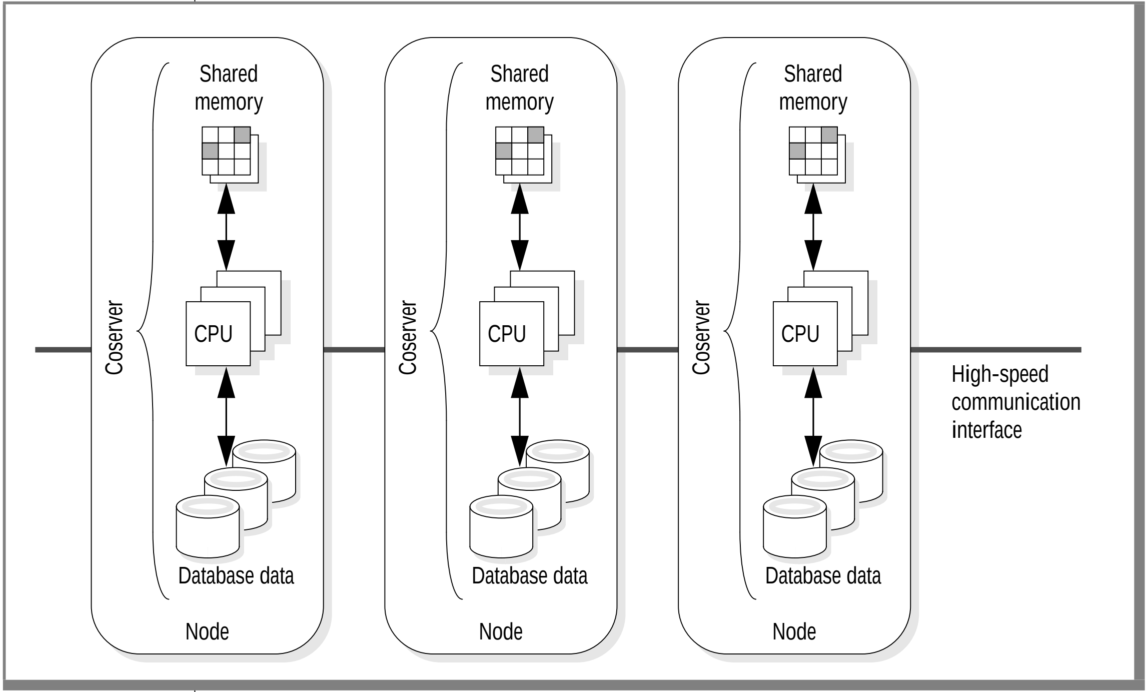
For complete details of the database server architecture, see the *Administrator's Guide* and *Administrator's Reference*.

What Is Extended Parallel Server?

Diverse information-management systems require a combination of on-line transaction processing (OLTP), batch operations, and complex decision-support system (DSS) applications. The architecture of Extended Parallel Server makes full use of all available hardware resources—including central processing units (CPUs), disks, memory, and network controllers—to deliver mainframe-caliber scalability, manageability, and performance, while requiring minimal operating-system overhead.

Extended Parallel Server is a *relational database server* that manages access to databases for client applications. It provides scalable performance on all types of hardware. Figure 1-1 illustrates the Extended Parallel Server architecture.

Figure 1-1
Sample Illustration of Extended Parallel Server



Extended Parallel Server is configured on a single computer or a parallel-processing platform as a set of one or more *coservers*. Each coserver performs database operations in parallel with the other coservers that make up the database server. Each coserver independently manages its own resources and activities such as logging, recovery, locking, and buffering.

Coservers communicate directly across a high-speed network, bus, or interconnect to access data and perform database operations in parallel. The architecture coordinates communication between coservers. Extended Parallel Server also coordinates concurrent data requests from multiple clients and enforces physical and logical consistency on the data.

Terminology

To understand the concepts that this manual discusses, you must understand the following terms. For a more detailed description of some of these concepts, see "Database Management Features" on page 2-7.

- As Figure 1-1 shows, the database server architecture consists of coservers that are located on nodes. A *node* is an individual computer with one or more CPUs that runs a single instance of an operating system on a parallel-processing platform. (See "Types of Parallel-Processing Architecture" on page 1-6.) A node can be a uniprocessor, a cluster of stand-alone computers, a symmetric multiprocessing (SMP) computer, or an independent subsystem configured in an SMP computer.
- A *coserver* is the functional equivalent of a database server that operates on a single node. Each coserver can run on a separate node and manages its own dbspaces.
  - A *connection coserver* is the coserver to which a client is directly connected.
  - A *participating coserver* is a coserver that controls one or more fragments of a table that the database server accesses.
- A *cogroup* is a named group of coservers. At initialization, the database server creates a cogroup named **cogroup_all** from all configured coservers.

The following kinds of logical storage units provide the ability to control the location of data, allowing the database administrator to improve performance and ensure the availability of data in case of media failure:

- A *dbspace* provides the link between the logical and physical units of storage.

  A dbspace lets a database server administrator associate physical units (such as chunks) with logical units (such as fragments of tables).
- A *dbslice* is a named set of dbspaces on multiple coservers.

  A dbslice is managed as a single storage object.
  - A *physslice* is a dbslice that contains the physical log.
  - A *rootslice* is the dbslice that contains the root dbspace.

- A *logical log* is an allocation of disk space that contains records of all changes that were performed on a database while the log was active.

  A logical log is managed by the database server and used to recover and restore data.

- A *logslice* is a set of log files that occupy a dbslice and are owned by multiple coservers, one log file per dbspace.

Additional terms:

- In this manual, a *platform* refers to the operating system and its underlying hardware.
- The **xctl** (execute utilities across coservers) utility lets you run coserver-specific utilities on multiple coservers and execute operating-system commands on multiple nodes. You can invoke **xctl** on any node in your platform or from any computer that can initiate commands remotely on your platform.

For other terms that pertain to Informix database servers, see the Glossary in the *Informix Guide to SQL: Reference*.

Types of Parallel-Processing Architecture

Extended Parallel Server provides parallel processing of SQL queries on various types of parallel computer architectures, including:

- *massively parallel processing* (MPP) systems, which are composed of multiple computers that are connected to a single high-speed communication subsystem.
- clusters of stand-alone computers that are connected over a high-speed network.
- *symmetric multiprocessing* (SMP) computers.

You can configure some SMP computers to mimic an MPP system. In a computer of this type, you can segregate resources into smaller, independently addressed subsystems with separate CPUs, memory regions, and I/O channels.

For information on how to configure Extended Parallel Server for your hardware, see "Deciding on Your Hardware Configuration" on page 3-5.

Client/Server Architecture

Informix client applications and database servers conform to a model of software design called *client/server*. Client / server functionality handles all of the connections between the client application and the database server.

A *client* is an application program that requests or modifies information from a database by issuing SQL statements. When the client connects to the database server through an SQL statement, the client transparently accesses **sqlhosts** connectivity information that is contained in a file on UNIX. (See "The sqlhosts Connectivity File" on page 3-15.)

Client programs include the DB-Access utility and programs that you write with an application-programming interface (API) such as ESQL / C or Informix ODBC Driver. Clients are briefly described in "Checking Product Compatibility" on page 3-7.

Extended Parallel Server processes requests for data from client applications, accesses the requested information in relational or dimensional databases, and returns the results to the client. Database-access activities include coordination of concurrent requests from multiple clients, read and write operations, and enforcement of logical and physical data consistency.

For information about client / server connectivity, see the *Administrator's Guide*. For full details about SQL, see the *Informix Guide to SQL* set. For detailed instructions on how to use an SQL API, see the appropriate programmer's manual, as listed in Chapter 4, "Using the Documentation."

Supported Connection Types

A *connection* is a logical association between a client application and a database server or between two database servers. You must establish a connection between the client and database server before data transfer can take place and you must maintain the connection for the duration of the data transfer.

Extended Parallel Server supports the following types of connections to communicate between client applications and database servers on UNIX platforms.

Connection Type	Local	Network
Sockets	✔	✔
TLI (transport layer interface)	✔	✔
Shared memory	✔	
Stream pipe	✔	

For more information about supported connectivity protocols, read the machine notes file. For information on how to set up local and network connections for Informix database servers and client applications, see the *Administrator's Guide*.

Supported Network Connections

To establish a *network connection* between a client on one computer and a database server on another computer, you must use a combination of a network interface and a network protocol.

Extended Parallel Server supports the following types of interface and protocol combinations for network connections.

Interface	Network Protocol
Sockets	TCP/IP
TLI	TCP/IP
TLI	IPX/SPX

For more information about the network connections that the database server supports, see the *Administrator's Guide*.

Supported Client/Server Configurations

Although a typical client / server configuration places a client application on one computer and a database server on another, a client application and database server can also reside on the same computer. The client / server environment is often a multiuser environment, with several clients accessing the same database server.

All coservers in Extended Parallel Server can accept client connections. A coserver that accepts the connection for a particular client is called the *connection coserver* for that client. To identify each connection coserver uniquely, the database server uses a *coserver name* that takes the following form:

```
dbservername.coserver-number
```

For more information about the client / server configurations that the database server supports, see the *Administrator's Guide*.

Supported Coserver Configurations

Extended Parallel Server supports the four types of coserver configurations that are discussed in this section. Each configuration is best suited to a particular platform architecture.

For more information, see "Deciding on Your Hardware Configuration" on page 3-5, and the *Administrator's Guide*.

Single Coserver on a Single-Node Platform

A single coserver on a single node is easy to administer. This configuration is best suited to a uniprocessor or an SMP computer that:

- has approximately 10 or fewer CPUs (for an SMP computer).
- has an amount of physical memory that is comparable in size to the address space of a single process.
- has enough local disk space to support the data that is required for the intended application.

Multiple Coservers on a Single-Node Platform

The use of multiple coservers can help to balance the processing load across the available memory and CPU resources. Multiple coservers are also useful for logging-intensive applications or situations in which extremely rapid recovery from any failure is an urgent requirement. This configuration is best suited to an SMP computer that:

- has between 8 and 24 CPUs or an amount of physical memory that is at least double the size of the address space of a single process.
- cannot be partitioned into subsystems to create separate nodes.
- has enough local disk space to support the data that is required for the intended application.

Single Coservers on Each Node of a Multiple-Node Platform

The use of multiple nodes allows the workload to be distributed across a high-speed interconnect, network, or bus (in the case of a partitioned SMP computer). Communication is not limited to client connections. This configuration is best suited to parallel-processing platforms composed of nodes that:

- each have 10 or fewer CPUs.
- each have an amount of physical memory that is comparable in size to the address space of a single process.
- each have enough local disk space to support an equal portion of the data that is required for the intended application, plus a suitable amount of space for temporary tables and sort files.

Multiple Coservers on Each Node of a Multiple-Node Platform

The use of multiple coservers on each node can help to balance the processing load across the available memory and CPU resources. This configuration is best suited to parallel-processing platforms composed of nodes that:

- each have between 8 and 24 CPUs or an amount of physical memory that is at least double the size of the address space of a single process.
- cannot be partitioned to form separate nodes.
- each have enough local disk space to support an equal portion of the data that is required for the intended application, plus a suitable amount of space for temporary tables and sort files.

Advantages of Extended Parallel Server Architecture

The most effective and efficient solution to operation problems is a database server architecture that delivers scalable performance. Extended Parallel Server lets you adjust performance dynamically to accommodate larger databases and a greater number of concurrent users, without degrading processing response time.

Your database applications also need to be portable from departmental environments to distributed, corporation-wide environments across a range of hardware platforms, from uniprocessor to symmetric multiprocessors and to clusters of SMP and MPP computers.

This section describes the following major advantages of Extended Parallel Server:

- Scalability of performance
- High availability and fault tolerance
- Single point of administration

Scalability of Performance

Extended Parallel Server provides a high degree of scalability for growing workloads. Scalability has the following two aspects:

- *Speed-up* is the ability to add computing hardware to achieve correspondingly faster performance for a DSS query or OLTP operation of a given volume.
- *Scale-up* is the ability to process a larger workload with a correspondingly larger amount of computing resources in a similar amount of time.

The following two key features make possible the linear scalability of the database server across a wide variety of computing platforms and workloads:

- A *shared-nothing* database server architecture that partitions data, control, and execution on various types of hardware
- The ability to execute SQL statements in parallel through efficient communication in a node as well as across multiple nodes

Shared-Nothing Database Server Architecture

Extended Parallel Server provides the following features to support both large and small hardware architectures:

- You can add nodes or processors without resource entanglements.
- The failure or substitution of a node or processor has little impact on the integrity of your system.
- Scalability issues are virtually eliminated.

The Extended Parallel Server approach to managing data minimizes operating-system overhead and reduces network I/O. Each node runs one or more instances of the database server as coservers, which perform their own logging, recovery, locking, and buffers.

Each coserver manages a set of disks and the fragments of the database that reside on these disks. Each coserver accesses only the disks that it owns.

To optimize performance, information about how the data is distributed can be cached across nodes in the system catalog. In addition, smaller, frequently accessed tables can be stored on only one coserver, then shipped across the nodes and cached in memory to further enhance performance.

All of the coservers in a shared-nothing infrastructure cooperate to provide the image of a single, large database server both to general database users and to system and database administrators.

The database server attains true parallel processing through partitioning and achieves its incremental performance improvements by partitioning data, control, and execution.

Data Partitioning

Dividing tables into multiple fragments permits operations such as scans, joins, and sorts to be distributed and executed in parallel on multiple CPUs and disks on separate nodes.

If the data is partitioned by expression or hash the database server can eliminate some fragments altogether for some queries, which reduces overall resource use.

Control Partitioning

Buffer management, database logging, checkpoints, and recovery are all partitioned across coservers so that each coserver manages its own data. Each coserver has its own locking, which eliminates the need for a central lock manager and its associated bottlenecks.

Execution Partitioning

Queries are broken down into smaller operations that can be executed locally on multiple coservers. Efficiencies across coservers are created in the following two major ways:

- Function shipping

  Function shipping is the process of sending execution requests to other coservers. Actual function execution is performed locally on appropriate coservers. Function shipping recruits only the coservers needed to execute a query. The coserver filters data and returns a subset of data to the connection coserver. This process reduces the need to ship data for processing.

- Transaction management

  The database server sends commit messages among coservers instead of sending data blocks across the network. Sending commit messages uses fewer resources than two-phase commit protocols, and its efficiency increases as more coservers are added to the database server.

For information on how partitioning affects the performance of Extended Parallel Server, see your *Performance Guide*.

Parallel Execution of SQL Requests

With Extended Parallel Server, coservers process SQL requests and run command-line utilities in parallel. The database server takes advantage of the underlying hardware parallelism to execute SQL operations in parallel. This ability to execute tasks in parallel promotes scalability.

For example, if you fragment the data across multiple coservers, the database server uses the disk space, memory, and CPUs of those multiple coservers to process different parts of the data in parallel. The processing gains are directly proportional to the number of CPUs. For example, Extended Parallel Server on an MPP system with 10 nodes can execute a complex query and access approximately 10 times the amount of data in the same amount of time as a single CPU.

High Availability and Fault Tolerance

Extended Parallel Server has fault-tolerant features that provide a highly available database environment. The database server uses the following mechanisms to protect data integrity and consistency if the operating system or media fails:

- Dbspace and logical-log backups of transaction records
- Fast recovery
- Mirroring
- Specific-time recovery

Dbspace and Logical-Log Backups of Transaction Records

Extended Parallel Server keeps a history of database and database server changes since the last dbspace backup. It has the ability to back up the data that it manages and also to store changes to the database server and data since the backup was performed. The changes are stored in *logical-log files*. The logical log contains records of logical transactions.

When the database server initializes disk space, it places the logical-log files and the physical log in the root dbspace. Before the database server modifies a dbspace data page, it stores a copy of the unchanged page in the physical-log page buffer. The database server eventually flushes the physical-log page buffer that contains this *before-image* to disk.

Each coserver can have multiple log files if multiple dbspaces exist. You can add multiple logslices to a dbslice as long as all the dbspaces have enough free space to accommodate the new log files. Each logslice is a distinct set of log files; log files are not shared across logslices.

You can create incremental or full backups of data while users are accessing the database server. Incremental backups enable you to back up only data that has changed since the last backup, which reduces the amount of time that a backup otherwise requires.

After a media failure, if critical data was not damaged and the database server remains on-line, you can restore only the data that was on the failed media, leaving other data available during the restore.

For more information about backing up data, see the *Backup and Restore Guide*.

Fast Recovery

When a coserver starts up, it checks the *physical log*, which contains pages that have not yet been written to disk. If the physical log is empty, that implies that the coserver was shut down in a controlled fashion. If the physical log is *not* empty, the coserver automatically performs an operation called *fast recovery*.

Fast recovery automatically restores tables to a state of physical and logical consistency after a system failure that might have left one or more transactions uncommitted. During fast recovery, the database server spawns multiple threads to work in parallel and uses its physical log and logical log to perform the following operations:

- Restore the databases to their state at the last checkpoint.
- Roll forward all committed transactions since the last checkpoint.
- Roll back any uncommitted transactions.

For more information about fast recovery, see the *Administrator's Guide* and *Administrator's Reference*.

Mirroring at the Database Server Level

When you use disk mirroring, the database server writes data to two locations. Mirroring eliminates data losses that result from media (hardware) failures. If mirrored data becomes unavailable for any reason, the mirror of the data is still available to users.

Mirroring pairs a *primary chunk* of one defined dbspace 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 either the primary chunk or the mirrored chunk, you can read from and write to the undamaged chunk while you recover the damaged media without interrupting user access to data.

Mirroring does use extra processing time to make each write to both the primary and secondary chunks. However, mirroring improves processing time for reading data because the database server reads from both the primary and secondary chunks to increase efficiency.

Ideally, you should mirror all of your data. However, if cost is a problem, you should select certain critical chunks to mirror. The logical-log files contain critical information and should be mirrored for maximum data protection. If you do not mirror your dbspaces, you must restore from a dbspace backup in the event of a media failure.

The operating system or hardware that you use might provide an alternative form of mirroring to the kind that the database server provides. If you consider a mirroring option that your operating system provides instead of one that the database server provides, compare the implementation of both options before you decide which to use. Operating-system mirroring options that do not use parallel mirror writes and split reads might provide inferior performance to database server mirroring.

Database server mirroring and operating-system mirroring run independently and can run at the same time. For example, you might have both database server data and nondatabase server data on a single disk drive. You could use operating-system mirroring to mirror nondatabase server data and database server mirroring to mirror database server data.

Logical-volume managers and hardware mirroring are other alternative mirroring solutions. Saving data to more than two disks with logical-volume managers gives you added protection from media failure, but the additional writes have a performance cost. Hardware mirroring such as redundant array of independent disks (RAID) has the advantage of requiring less disk space to store the same amount of data than does Informix database server mirroring, but it is slower for write operations.

For more information about mirroring, see the *Administrator's Guide* and *Administrator's Reference*.

Specific-Time Recovery

Use specific-time recovery to restore data after a catastrophic event. Specific-time recovery lets you restore the database to a specific time, perhaps immediately preceding the event. A specific-time recovery can undo mistakes, such as dropping a table, that might not be fixable otherwise.

When you restore the database server to a specific time, some transactions might be lost even though they are included in an existing logical-log backup. The transactions are lost because the database server can only be restored to the last known global point of consistency across all coservers.

For information about specific-time recovery and data restoration, see the *Backup and Restore Guide.*

Single Point of Administration

With Extended Parallel Server, many coservers function together to form a single database server. These coservers usually have the same number of CPUs and equal amounts of memory and disk storage space. This uniformity makes it easier to administer all coservers from a single point.

The following features help you control and manage multiple coservers:

- Centralized configuration file
- Cogroups
- Dbslices
- Centralized message log
- Coordinated data-dictionary cache
- Centralized database server utilities

For more information, see the *Administrator's Guide* and *Administrator's Reference*.

Database Support

Extended Parallel Server supports the following types of databases:

- Relational databases
- Data warehouses
- ANSI-compliant databases
- Distributed databases

Relational Database Concepts

A relational database consists of tables that are made up of rows and columns. An Informix relational database management system (RDBMS) includes the following components:

- A database server
- A database
- One or more client applications

Data Warehousing Concepts

A *data warehouse* is a database that typically contains large stores of historical data. A primary advantage of a data warehouse database is that it provides easy access to, and analysis of, vast stores of information. The term *data warehouse* can be used to refer to one of the following database environments:

- Data warehouse

  A data warehouse is a database that is optimized for data retrieval. Data is not stored at the transaction level; some level of data is summarized.
- Data mart

  A data mart is a subset of a data warehouse that is stored in a smaller database. This DSS operation is not enterprise wide. It draws selected data from OLTP operations or from a data warehouse to answer specific types of questions or to support a specific objective.
- Operational data store

  An operational data store is a subject-oriented system for decision making that is optimized for looking up one or two records at a time. It contains timely, current, integrated information that typically is very granular.
- Repository

  A repository is a system that combines multiple data sources into one normalized database. Data stored in a repository is operational rather than historical.

For details of how to plan a database, build a database model, create the planned database, implement data warehousing, and work with data types and fragmentation, see the *Informix Guide to Database Design and Implementation*.

ANSI-Compliant Databases

Extended Parallel Server supports ANSI-compliant databases. An ANSI-compliant database enforces ANSI requirements, such as implicit transactions and required ownership, that are not enforced in databases that are not ANSI compliant. The benefits of ANSI-compliant databases include portability, object access, name and transaction isolation, and data recovery. For additional considerations, see "Creating a Database" on page 3-23.

For information about ANSI-compliant databases, see the *Informix Guide to Database Design and Implementation* and the *Informix Guide to SQL* set of manuals.

Distributed Databases and Transactions

Extended Parallel Server lets users perform transactions on data from databases that reside on different database servers connected across a network. The database server supports two multiphase commit protocols, two-phase commit protocol and heterogeneous commit protocol, to process transactions that span multiple database servers.

Two-Phase Commit

A two-phase commit protocol ensures that transactions are uniformly committed or rolled back across multiple Informix database servers. The two-phase commit protocol governs the order in which a two-phase commit transaction is performed and provides an automatic recovery mechanism in case a system or media failure occurs during execution of the transaction.

A database server automatically uses the two-phase commit protocol for any transaction that performs modifications to data on more than one database server. The database server uses logical-log records to implement the two-phase commit protocol. You can use these logical-log records to detect heuristic decisions and, if necessary, to help you perform a manual recovery.

Heterogeneous Commit

In the context of Informix database servers, the term *heterogeneous environment* refers to a group of database servers in which at least one is not an Informix database server. Heterogeneous commit is a database server feature that ensures the all-or-nothing basis of distributed transactions in a heterogeneous environment.

Unlike the two-phase commit protocol, the heterogeneous commit protocol supports the participation of a non-Informix participant. The non-Informix participant, called a gateway participant, must communicate with the coordinator through an Informix Enterprise Gateway product.

For information about two-phase commit and heterogeneous commit protocols and concepts and the use of logical-log records in distributed transactions, see the *Administrator's Guide* and *Administrator's Reference*. For information about a specific Informix Enterprise Gateway product, see the appropriate *Informix Enterprise Gateway User Manual*.

Database Server Security

The databases and tables that the database server manages enforce access based on a set of database and table privileges. You can use the following SQL statements to manage these privileges:

- Use the GRANT and REVOKE statements to give or deny access to a database or specific tables and to control the kinds of database uses.
- Use the CREATE PROCEDURE statement to write and compile a stored procedure that controls and monitors access to tables.
- Use the CREATE VIEW statement to prepare a restricted or modified view of the data.
- Combine the GRANT and CREATE VIEW statements to precisely control the parts of a table that a user can modify.

Informix database servers follow UNIX security requirements for making connections. Thus, the UNIX system administrator might need to make modifications to the **/etc/passwd**, **/etc/hosts**, **~/.rhosts**, and other related files.

For the syntax and description of SQL statements, see the *Informix Guide to SQL: Syntax*. For information about database and table privileges and how to control access to databases, see the *Informix Guide to Database Design and Implementation*.

Chapter

Features of Extended Parallel Server

2

In This Chapter

Extended Parallel Server includes features that can improve performance on large databases and make complex database servers easier to manage. It is well-suited to managing large data-warehousing and decision-support system (DSS) applications as well as on-line transaction processing (OLTP) applications.

This chapter provides an overview of the features that make Extended Parallel Server unique.

Completely Parallel SQL Operations

Extended Parallel Server supports the following database operations:

- Aggregation operations (such as GROUP BY, DISTINCT, SUM, MAX, and so on)
- Data and index scans
- Index builds
- Inserts, updates, and deletes
- Joins
- Set operations (such as union, intersection, and difference)
- Sorts

The database server can perform database operations in parallel when tables are fragmented appropriately across all coservers.

For general information about fragmentation, see the *Informix Guide to Database Design and Implementation*. For details of how to use fragmentation and parallel database query (PDQ) to improve database server performance, see your *Performance Guide*.

Data Partitioning

Data partitioning, or *table fragmentation*, is a method of separating a table into potentially balanced fragments so that Extended Parallel Server can use the parallel-processing ability of the underlying hardware to make database operations faster.

You can fragment tables in several ways. A database designer must specify an appropriate distribution scheme for each table. The best method for grouping rows into fragments depends on how the table is used.

The main goals of fragmentation are to:

- improve single-user response time, especially for large DSS queries.
- improve data-load performance.

To meet these goals, a fragmentation method should:

- balance the distribution of data.
- eliminate irrelevant table fragments automatically when certain queries are run.

Use the parallel database query (PDQ) feature to improve the performance of individual queries. PDQ lets the database server perform parallel scans of fragments that are in a single coserver or spread across multiple coservers.

Use fragmentation with the high-performance load and unload feature to improve performance when you load very large databases (VLDBs). For more information, see "High-Performance Data Loading" on page 2-13.

Enhanced Fragmentation Methods

Enhanced fragmentation methods have a number of advantages. The database server can fragment tables across disks that reside on different nodes and are managed by different coservers. Each table fragment can reside in a separate dbspace on different coservers. Dbslices provide the mechanism for managing many dbspaces across multiple coservers.

Enhanced fragmentation in Extended Parallel Server enables you to:

- fragment a table with a system-defined hash distribution scheme and store table fragments in separate dbspaces that are associated with physical disks on different coservers.
- designate a fragmentation expression to put rows that contain specified values in the same fragment.
- fragment a table with a hybrid system-defined hash and expression-based distribution scheme.
- place rows sequentially in fragments and distribute the rows in round-robin fashion.
- store the results of a query in a temporary table that the database server dynamically fragments across disks that belong to different coservers.

Tip: *For VLDBs that are fragmented across many coservers, Informix recommends that you fragment by system-defined hash on the join column.*

The following sections describe the distribution schemes that Extended Parallel Server supports.

Expression-Based Fragmentation

Expression-based fragmentation distributes data according to a user-specified expression that is defined in the WHERE clause of an SQL statement. The range rule of expression-based fragmentation distributes data in table fragments that contain a specified key range, such as the first letter of the customer last name.

Although expression-based and range partitioning cannot ensure that data is distributed evenly, queries that specify a range of the determining key might be faster because they can easily eliminate scans of table fragments that do not contain the required rows.

Hybrid Fragmentation

Hybrid fragmentation lets you specify two fragmentation methods. Usually, you use one method globally to fragment the table across coservers, and use the other method locally on each coserver to fragment the table across the coserver disks.

Users of Extended Parallel Server can specify hash fragmentation to distribute the workload evenly across coservers and expression-based fragmentation to optimize I/O efficiency in the coserver.

Range Fragmentation

Range fragmentation ensures that rows are fragmented evenly across dbspaces. In range distribution, the database server determines the distribution of rows among fragments based on minimum and maximum integer values that the user specifies. Informix recommends a range distribution scheme when the data distribution is both dense and uniform.

Round-Robin Fragmentation

Round-robin fragmentation distributes data sequentially and evenly across specified dbspaces. However, because data is randomly placed in dbspaces, the query optimizer does not know which dbspace contains any specified row. Thus, the optimizer cannot eliminate fragments to scan for queries.

System-Defined Hash Fragmentation

Hash fragmentation maps each row in a table to a set of integers. It uses a system-defined hash algorithm to distribute data evenly by hashing a specified key. In addition to even data distribution, hash fragmentation permits the automatic elimination of fragments for queries that use the hashed key. You can use hash fragmentation for several tables to provide fragment elimination when the tables are joined in queries and to perform more processing on the local coserver.

Fragment Elimination

Fragment elimination is the process of applying a filter predicate to the fragmentation strategy of a table or index and removing the fragments that do not apply to the operation. This feature includes:

- one- or two-column elimination with any combination of supported operators.
- overlapping ranges of expressions in a fragmentation scheme.
- noncontiguous ranges.
- remainder-fragment access only as necessary.

Database Management Features

As databases grow larger and more complex, they require new management features. Extended Parallel Server provides organizational features that make it easier to manage large databases. For example:

- Cogroups simplify administration of a large number of coservers.
- Dbslices simplify management of a large number of dbspaces that contain table fragments.
- Logslices simplify the addition and deletion of log files.
- Logging and nonlogging tables support data warehousing.

Use **onutil** utility commands to create, alter, drop, and otherwise work with cogroups, dbslices, logslices, and other database management objects.

This section provides an overview of these database management features. For complete information, see the *Administrator's Guide* and *Administrator's Reference*.

Coserver Groups (Cogroups)

A coserver group (usually referred to as a *cogroup*) is a subset of the coservers in the database server that are grouped together for system and database administration. A coserver can be in more than one cogroup. For example, you might create the following cogroups:

- CUSTOMER_COGROUP, for the coservers that own the data for a customer application
- ACCOUNT_COGROUP, for the coservers that own the data for an accounting application

The default system-defined cogroup is called **cogroup_all**. The **cogroup_all** cogroup includes all of the coservers that your ONCONFIG configuration file defines.

Tip: *Informix recommends that you partition (fragment) VLDBs across many coservers. Because each table fragment is stored in its own dbspace, you might easily create thousands of dbspaces spread across multiple coservers. Managing these individual dbspaces can be complex and error prone unless you can manage them as groups.*

Database Slices (Dbslices)

A *dbspace* is a logical storage unit that provides the ability to control where the database server stores data. A dbspace links the logical and physical storage units and lets the administrator associate physical units with logical units.

To manage many dbspaces, Extended Parallel Server uses the concept of a *dbslice,* which is a named set of dbspaces that can span multiple coservers. You can manage a dbslice as a single storage object.

A dbslice simplifies management of storage objects because you can refer to all of the storage objects for a single table with the dbslice name. For example, to fragment a table across 100 coservers, you can use the following example CREATE TABLE command in the **onutil** utility to specify a single dbslice name instead of 100 dbspace names:

```
CREATE TABLE customer
    (cust_id integer,
    .
    .
    .
    )
    FRAGMENT BY HASH (cust_id)
        IN customer_dbslc;
```

In this example, the SQL operation occurs for all of the underlying dbspaces in the **customer_dbslc** dbslice.

When you create a dbslice, you specify the cogroup name so that the database server knows the coservers on which to create dbspaces. For example, you might create a dbslice from an accounting cogroup. The following example commands show how to create a cogroup and dbslice:

```
% onutil
    CREATE COGROUP acctg_group
    .
    .
    .
    CREATE DBSLICE acctg_dbslc
        FROM acctg_group . . .
```

You do not need to specify the names explicitly for all of the individual dbspaces that are associated with the partitioned tables. The database server generates the dbspace names for you.

Logslices

A *logslice* is a set of log files, owned by multiple coservers, that reside in a dbslice. By treating sets of log files as single entities, logslices simplify the process of adding and deleting log files.

If a dbslice has multiple dbspaces per coserver, each coserver will have multiple log files. You can add multiple logslices to a dbslice as long as all the dbspaces have enough free space to accommodate the new log files.

Each logslice is a distinct set of log files. Log files are not shared across logslices.

You can also create individual log files on a specified coserver. For example, you can vary the number of log files per coserver in a way that reflects the different requirements of each coserver. However, you cannot perform operations on the individual log files that are part of a logslice, and you cannot violate the requirement that each coserver have a minimum of three logical-log files.

Logging and Nonlogging Permanent and Temporary Tables

Data warehousing and similar applications that involve very large amounts of data (and few or no inserts, updates, or deletes) often need a mix of logged and nonlogged tables in the same database. Temporary tables are insufficient in many cases because temporary tables are visible only to the session that created them and they do not persist after the session ends.

Although Extended Parallel Server does not support nonlogging databases, it supports nonlogging tables. By default, the database server creates standard permanent tables with logging. However, to accommodate the needs for both logging and nonlogging tables for data warehousing and other VLDB applications, Extended Parallel Server supports various classes of permanent and temporary tables.

The following table lists the classes of permanent tables.

Class of Permanent Table	Use	Logging or Nonlogging
Operational	For use where the data is derived from a source outside the database server. Operational tables use *light appends* (unbuffered, unlogged insert operations). They allow rollback and recovery but do not allow restoration from archive.	Logging
Raw	For the initial loading and scrubbing of data. Raw tables use light appends, which bypass the buffer cache. They do not support indexes, referential constraints, rollback and recovery, or restoration from archive.	Nonlogging
Standard	The ordinary logged tables in a logged database in Extended Parallel Server. Standard tables do not use light appends but support recovery and rollback.	Logging
Static	For read-only operations with indexing allowed. Static tables use light scans and do not need locking. They allow constraints and nonclustered indexes but do not allow rollback and recovery or restoration from archive. Static tables do not support data manipulation operations (inserts, updates, and deletes). Static tables permit advanced indexing methods that have been created especially for DSS queries.	Nonlogging

The following table lists the classes of temporary tables.

Class of Temporary Table	Use	Logging or Nonlogging
Scratch	Unlogged tables that use light appends. Scratch tables do not support indexes, referential constraints, or rollback.	Nonlogging
Temp	Logged tables that support bulk operations such as light appends. Temp tables support indexes, referential constraints, and rollback but do not support recovery because they are temporary.	Logging

The following table lists the types of temp tables.

Type of Temp Table	Definition
Explicit	A temp table that you create with clauses of the SQL statements CREATE TABLE and SELECT. When a SELECT statement explicitly creates a temporary table, the SELECT statement is parallelized across coservers and the table is fragmented across nodes.
Flexible	A temp table that the database server creates and automatically fragments with the round-robin method. An advantage of a flexible temporary table over a table created with CREATE TABLE syntax is that you do not need to know column names and data types.
Implicit	A temp table that the database server creates as part of processing. The database server deletes an implicit temporary table when the processing that initiated the creation of the table is complete.

For more information about the kinds of tables that Extended Parallel Server supports and their characteristics, see the *Informix Guide to Database Design and Implementation*.

Management Tools and Utilities

Extended Parallel Server features include the following management tools and utilities:

- On-line backup and restore (ON-Bar) (page 2-11)
- Informix Storage Manager (page 2-12)
- High-performance data loader (page 2-13)
- Cost-based optimizer (page 2-15)
- Database server utilities (page 2-15)

ON-Bar

On-line Backup and Restore (ON-Bar) for Extended Parallel Server provides a single-system view for backing up and restoring logs and data that span multiple coservers and multiple computers.

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

ON-Bar provides backup and restore capability for all dbspaces and logical logs. To prevent performance problems that overfull logical-log space can cause, you can have logical logs backed up automatically as soon as they fill. You can schedule backups of dbspaces appropriately as insurance against system failure.

ON-Bar uses the X/Open Backup Services Application Programmer's Interface (XBSA). It connects to either the Informix Storage Manager (ISM) or another XBSA-compliant storage manager that provides device and storage management functionality.

You can activate the **archecker** utility from ON-Bar to check the validity and completeness of the objects that ON-Bar has backed up before you use the objects to restore dbspaces.

For more information about ON-Bar, see the *Backup and Restore Guide*.

Informix Storage Manager

Informix Storage Manager (ISM) provides data-storage management services for your Informix database server. ISM resides on the same computer as ON-Bar and the Informix database server.

ISM receives backup and restore requests from ON-Bar and directs your data to and from storage volumes that are mounted on storage devices. ISM tracks backed-up data through a data life cycle that the database or system administrator determines and can automatically manage your storage devices and storage volumes.

ISM has the following components:

- The ISM server for exchanging backup and recovery information between storage devices and ON-Bar
- The ISM Administrator program for managing and configuring the ISM server, storage media, and storage devices
- The ISM catalog, which maintains up-to-date records of the backup operations that have been performed and the media on which the backed-up data is stored

ISM works with ON-Bar to perform the following kinds of tasks:

- Provide complete media management for all ON-Bar backups and restores
- Track the location of all backup data and move backup data through a managed life cycle
- Provide complete disaster-recovery protection for a database server instance

For information about how to configure and use ISM, see the *Informix Storage Manager Administrator's Guide*.

High-Performance Data Loading

Extended Parallel Server performs high-performance parallel loading and unloading of data with *external tables*. An external table contains an external-table definition that includes all the information needed to define data in an external data file. The information includes data formatting type, external data description fields, and global parameters.

You issue a series of SQL statements, often in parallel, to perform high-performance loading functions such as the following:

- Partition load activity over available resources
- Efficiently transfer operational data from a mainframe to a data warehouse
- Transfer data files across platforms in Informix internal data format
- Use the database server to convert data between delimited ASCII, fixed-ASCII, EBCDIC, and Informix internal (raw) representation
- Specify the mapping of data to new columns in a database table
- Provide parallel standard INSERT operations to load data without dropping indexes
- Load data to and from storage devices, including tape drives and direct network connections to mainframes
- Perform express (high-speed) and deluxe (data-checking) transfers

Extended Parallel Server supports raw and delimited file formats for simple large objects that contain BYTE or TEXT data.

You also can use the load feature to load values into serial columns. The database server can take the values from the original data file or generate them automatically.

Use the following SQL statements to set up load and unload tasks:

- CREATE EXTERNAL TABLE ... USING describes the location of the external file (on disk or from pipe) and the format of external data.
- INSERT INTO ... SELECT transfers the external data to or from the database tables.
- SELECT ... INTO EXTERNAL *table-name* USING creates a default external table description for unloading data.

You can issue these SQL statements in the DB-Access utility or embed them in ESQL/C programs.

You can create scripts that load and unload data at specified intervals and use the resulting tables for periodic query reports and storage of significant data for statistical analysis.

Use high-performance loading with fragmentation to improve the loading and unloading performance of VLDBs. When you use this feature to load a table that is fragmented across multiple coservers, Extended Parallel Server allocates threads to load the data into the fragments in parallel.

There are two high-performance loading modes:

- *Express mode* is high speed. It provides the highest performance during a load but does not log the data. Express mode uses *light appends*, which bypass the buffer pool and eliminate the overhead associated with buffer management.
- *Deluxe mode* performs data checking. It uses regular single-row inserts, which add rows to a table that can contain indexes and also can check unique constraints for each row.

For more information about loading and unloading with external tables, see the *Administrator's Reference*. For the SQL statement syntax, see the *Informix Guide to SQL: Syntax*.

Cost-Based Optimizer

Before a query is executed, the query optimizer formulates an execution plan based on the lowest resource-use cost to fetch the data rows that are required to process a query.

The optimizer evaluates the different ways in which a query might be performed. For example, the optimizer determines whether to use one or multiple indexes. If the query includes a join, the optimizer determines the join method (hash, sort-merge, or nested loop) and the order in which to evaluate or join tables. The optimizer assesses each query plan on a cost basis and selects the plan with the lowest execution cost.

Hash joins can improve query execution speed when tables are joined on arbitrary, unindexed columns. In a hash join, one table is scanned to create a hash table based on the join key. Subsequent tables are read once to look up each row in the hash table to find out if a join can be made. Hash joins are more efficient than sort-merge joins because nothing is sorted. Eliminating sorting makes hash joins particularly efficient for large tables.

For more information on the cost-based optimizer, see your *Performance Guide*.

Database Server Utilities

Extended Parallel Server includes several database server utilities, as the following table outlines.

Utility	Use
oninit	Initializes shared memory and brings the database server on-line
onlog	Displays the contents of a logical-log backup file
onmode	Changes the database server and coserver operating mode and shared memory
onstat	Gathers statistics about Extended Parallel Server and monitors coserver activity
onutil	Defines and modifies Extended Parallel Server storage objects such as cogroups, dbslices, dbspaces, logical logs, and logical logslices. It also includes check commands for monitoring database server status.
xctl	Lets you execute database server utilities such as **oninit**, **onlog**, **onmode**, and **onstat** on one or more coservers and execute operating-system commands on one or more nodes.

These database server utilities are described in the *Administrator's Reference*.

Features for DSS Applications

DSS applications are often used to report on or consolidate data that was captured through OLTP operations over a certain period of time. (For more information, see "Features for OLTP Applications" on page 2-21.)

DSS applications provide information for strategic planning and decision making. Data in the database is typically queried but not updated during DSS operations. Typical DSS applications include payroll, inventory reporting, and end-of-period accounting.

DSS applications on Extended Parallel Server have the following characteristics:

- Complex queries that involve large amounts of data
- Large memory requirements
- Few users
- Periodic requests
- Relatively long response times

DSS applications perform more complex tasks than do OLTP operations. DSS tasks might include scans of entire tables, manipulation of large amounts of data, multiple joins, and the creation of temporary tables. DSS execution times are typically much longer than OLTP execution times.

The increasingly popular concept of *data warehousing* consolidates enterprise-wide data into a specially designed environment. A data warehouse stores business data for a company in a single, integrated relational database that provides a historical perspective on information for DSS applications.

Data-warehousing applications manipulate hundreds of gigabytes of data. Data-warehouse features include extensions to statements to accelerate large decision-support queries and sophisticated indexing methods to accelerate access to data.

Another approach to DSS operations is called a *data mart*. A data mart draws selected data from OLTP operations or from a data warehouse to answer specific types of questions or to support a specific department or initiative in an organization. Extended Parallel Server provides exceptional performance for DSS applications such as data marts and data warehouses and excels at handling the complex queries that are typical of decision-support applications.

For information about how to create a dimensional database (data mart), see the *Informix Guide to Database Design and Implementation*. For more information about Extended Parallel Server and DSS, see the *Administrator's Guide* and *Administrator's Reference*.

SQL Features

Extended Parallel Server supports SQL features that increase the speed and efficiency of large queries.

For detailed information about many of the features described in this section, refer to your *Performance Guide*.

Enhanced SQL Statements

Because of its unique architecture and features, Extended Parallel Server includes modified SQL statement syntax and functionality.

For information about these SQL statements and expressions, see the *Informix Guide to SQL: Syntax*.

Sampling

Sampling provides a user-specified number of randomly selected rows from a table to give users statistical information about data without scanning the entire table. For example, sampling 10,000 records from a 10-million-row table is significantly faster than reading all the rows and can provide information about sales trends.

Sample scans randomly select a specified number of rows from the table. For example, the following query gives an analyst an idea of the average sales amount by sampling 10,000 records instead of scanning the entire sales table:

```
SELECT AVG(s.dollar_amount)
    FROM 10000 SAMPLES of SALES s;
```

Because the sample is random, each execution of the query produces slightly different results. To save a sample set of rows, use an INSERT...SELECT or SELECT...INTO statement so that further statistical tests can be run on the same sample.

The CASE Expression

The CASE expression allows conditional expressions as part of an SQL statement to reduce the complexity of database operations. The following example illustrates a query that the CASE expression simplifies.

```
SELECT
    CASE
        WHEN sales.amount < 100 THEN "small sales"
        WHEN sales.amount < 1000 THEN "mid-range sales"
        ELSE "large sales"
    END,
    SUM (sales.amount)
    GROUP BY 1;
```

The LOCAL Keyword

One key advantage of Extended Parallel Server is its ability to fragment a table across all coservers and to use parallel processing to return query results quickly. However, a user sometimes needs to access only the data on a local table fragment.

The LOCAL keyword lets the client application read only the table fragments on the coserver to which the client is connected. Depending on the method used to fragment a table, a query that uses the LOCAL keyword might return a useful sample of data in the entire table.

The following example query selects records from the local fragment of the **customer** table:

```
SELECT * FROM LOCAL CUSTOMER c;
```

The LOCAL keyword is ignored if the table is not fragmented.

Updating from a JOIN

The UPDATE ... FROM join clause extends the UPDATE statement to update one table from rows in another table that match a set of criteria. This extension can save users from having to update the table manually.

The following example query uses the matching records in the **sales** table to update the telephone numbers of customers in the **customer** table:

```
UPDATE CUSTOMER
    SET telephone_no = SALES.telephone_no
    FROM CUSTOMER c, SALES s
    WHERE c.cid = s.si;
```

Unique Indexing Methods

As described in "Completely Parallel SQL Operations" on page 2-3, Extended Parallel Server architecture and parallel SQL statements promote performance improvement and scalability for query execution. However, in addition to an optimized database architecture that provides fast database server performance, DSS operations require optimal indexing techniques to ensure quick access to data.

In addition to improvements to B-tree indexes, Informix has designed special generalized-key indexing techniques (GK indexes) specifically to optimize performance of DSS applications that use static tables.

Unlike conventional indexes, GK indexes let key values be more than just column values from the indexed table. The query optimizer decides when to use a GK index.

Indexing techniques for DSS applications include the following methods:

- Join indexes

  A *join index* is essentially a precompiled join that consists of references to rows in two or more static tables that satisfy the join condition. For efficiency and reduced storage, bitmaps represent values for rowids in the joined tables in the join index.

  Join indexes are especially useful in queries of databases in which data is represented in a star schema, a common representation method in data-warehouse databases. This type of join is known as a *star join*.

  Several variations of the basic join index are possible and appropriate for some queries. These variations might translate a restriction on a column in one table to a restriction on another table or create an index only on the join keys of dimension tables in a star schema database.

- Virtual columns and virtual-column indexes

  A *virtual-column index* is an index on a virtual column. You can alter a table to create a virtual column that represents an expression on other columns in a static table, or you can create an index on the table to represent the expression. In either case, you can use the column or the index to make queries more efficient when they group or select rows based on the virtual value.

- Selective indexes

  A *selective index* is built on table columns for a subset of a table. It contains entries for table rows selected by a WHERE clause. The query optimizer uses a selective index for queries that select rows included in the index.

For general information about indexing, see the *Informix Guide to SQL* set. For specifics of Extended Parallel Server indexing, such as GK indexes and star joins, see your *Performance Guide*. For an overview of indexes for data-warehousing environments, see the *Informix Guide to Database Design and Implementation*.

Features for OLTP Applications

OLTP applications are often used to capture new data or update existing data. These operations usually involve quick indexed access to a small number of rows. OLTP applications are often multiuser applications where acceptable response times are measured in fractions of a second. An order-entry system is a typical OLTP application.

OLTP applications on Extended Parallel Server have the following characteristics:

- Simple transactions that involve small amounts of data
- Indexed access to data
- Many users
- Frequent requests
- Very fast response times

For more information about Extended Parallel Server and OLTP, see the *Administrator's Guide* and *Administrator's Reference*.

Running DSS Queries in OLTP Environments

DSS queries are sometimes run in OLTP database environments. (For more information, see "Features for DSS Applications" on page 2-16.)

DSS queries might be run either on a regular schedule, such as for inventory or end-of-period accounting reports, or on a one-time basis. Extended Parallel Server provides features that make such queries efficient and reduce their effect on ongoing work in the database.

Managing System Resources for DSS Queries

Configuration parameters, environment variables, and SQL statements let you specify the amount of memory that queries are allowed to use as well as the scheduling priority of the queries. Because you can specify these settings globally, per session, or per query, you can tailor the environment for DSS queries to fit comfortably in an OLTP environment.

The following value settings determine how DSS queries are run on Extended Parallel Server:

- MAX_PDQPRIORITY

  This parameter specifies the total amount of memory that all queries that run on the database server can have. You must set MAX_PDQPRIORITY globally.
- PDQ priority

  This value specifies the amount of memory that a single query can use. You can specify the PDQ priority either for a session as the PDQPRIORITY environment variable, or in a query with the SET PDQPRIORITY statement.
- DS_ADM_POLICY

  The setting of this parameter determines the order in which queries are run. Use this parameter to specify two main methods for running queries.

  - One setting specifies that queries be run on a first-come, first-served basis, depending on the priority that the SQL statement SET SCHEDULE LEVEL requested in the query. The query with the highest scheduling level is the first query to run. If more than one query is at the same level, the earliest one is run.
  - The other setting specifies that queries be run in a more balanced way. This setting allows smaller queries with a lower scheduling level to run before larger queries with a higher scheduling level if the smaller queries were waiting for a long time.

Using Indexes More Efficiently

Each Informix data record in a table is uniquely identified by a fragment identifier (FID) and row identifier (rowid) pair. The FID identifies the dbspace where the record resides, and the rowid identifies the logical page in that dbspace and the location in that page where the record resides.

Extended Parallel Server supports several indexing techniques that are appropriate for both OLTP and DSS applications, as follows:

- Obtain all rowids from the index and sort the rowid list so that a table page is never accessed more than once, and seek time is minimized
- Use all of the indexes that apply to a query instead of selecting only the best index
- Perform index retrievals in large batches so that the B-tree is not traversed for each entry
- Store bitmaps of rowids in the leaf node of the index instead of as single entries

Improvements to B-Tree Indexes

Some of the listed indexing techniques are enhancements to B-tree indexes. B-tree indexes are very effective for retrieving a small number of selected rows, as is common in OLTP applications. However, DSS queries usually retrieve a larger number of rows. Thus, random table access through a single index can be more expensive than reading all of the rows in the table, especially when performed in parallel.

Use of Bitmap Indexes

When appropriate, the rowid entries for the data in the indexed table are compressed as bitmaps for efficient access and disk-space use. Such indexes reduce index size and are particularly useful for performing AND, OR, NOT, and COUNT operations, which are often used in DSS queries.

In a bitmap index, each value has a single bitmap at the leaf node. In a conventional B-tree index, each value contains a list of rowids at the leaf node. Thus bitmap indexes are more efficient for columns that contain a small number of values. In an ideal case where the column has only two values, only two bitmaps are required.

A related data-retrieval method is *sequential skip scan*, which reads rowids from an index and sorts them before it retrieves any row. This method ensures that no page is read more than once and that unnecessary pages are never read.

You can compress the bitmaps to achieve further size reduction and efficiency increases. Logical operations such as AND or NOT between compressed bitmaps are extremely efficient.

Although these performance improvements were designed primarily for DSS query processing in static databases, they might be useful for OLTP databases in which users sometimes run DSS queries. Bitmap-enhanced B-tree indexes are updated automatically when tables are updated. Thus, you can run queries more efficiently than previously in OLTP databases.

For information about bitmap indexes and B-trees, see the *Administrator's Guide* and *Administrator's Reference*. For information about using SQL statements to create indexes, see the *Informix Guide to SQL* set. For information about managing system resources and how indexing techniques can improve database server operations, see your *Performance Guide*.

GLS

Global Language Support (GLS)

Extended Parallel Server provides Global Language Support (GLS). The GLS feature lets the database server handle different languages, cultural conventions, and code sets. GLS provides support for the following language-related items:

- Collation order of characters
- Definition of uppercase and lowercase conventions
- Non-ASCII characters, including multibyte characters
- Culture-specific formatting for numeric, monetary, date, and time values

With GLS support, Extended Parallel Server does not need to specify how to process culture-specific information directly because it resides in a GLS locale. When the database server needs culture-specific information, it makes a call to the GLS library. The GLS library, in turn, accesses the GLS locale and returns the information to the Informix product.

For complete information about GLS support, see the *Informix Guide to GLS Functionality*.

Chapter

3 Getting Up and Running with the Database Server

In This Chapter

This chapter provides information to help you get Extended Parallel Server up and running. It tells you what you need to do to install the database server, configure the environment, and perform administrative and other tasks.

Performing Basic Tasks for Getting Started

Perform the following tasks to bring up Extended Parallel Server successfully:

- Migrate to Extended Parallel Server, Version 8.3 (if needed) (page 3-4)
- Plan your database server
- Set up your hardware configuration (page 3-5)
  - Allocate disk space (page 3-6)
  - Set permissions (page 3-7)
- Check product compatibility (page 3-7)
- Install the database server and related products (page 3-10)
- Configure an Extended Parallel Server environment (page 3-12)
  - Prepare the operating system (page 3-13)
  - Set required environment variables (page 3-13)
  - Prepare the connectivity and configuration files (page 3-14)
- Start the database server and initialize disk space (page 3-21)
- Configure a client or SQL API environment (page 3-22)
- Choose a database type and create a database (page 3-23)

- Perform post-installation tasks (page 3-24)
  - Create the demonstration database (page 3-24)
  - Get error message information (page 3-25)
  - Manage coservers, cogroups, dbslices, and dbspaces (page 3-25)
  - Monitor fragmentation, sessions, and queries (page 3-27)
- Perform administrative tasks (page 3-29)
- Monitor the database to maintain performance (page 3-30)

Migrating to Extended Parallel Server

If you migrate to Extended Parallel Server, Version 8.3, from an earlier version of the database server, start with the information that the *Informix Migration Guide* provides.

Planning a Database Server

Before you install and create a database server, make sure that you can answer the following questions:

- What do you want to name the database server and coservers?
- Do you want to use default settings appropriate to your hardware platform, or do you want to customize some settings?

  If you choose to customize settings, be prepared to answer these questions:

  - Do you expect to use this database server primarily for OLTP or for DSS applications?
  - What percent of CPU and memory on each node do you want to allow the database server to use?
  - Which of the available nodes do you want to use for this database server?
  - Should ON-Bar back up logical logs automatically as soon as they fill?

- ❑ Which nodes are connected to backup storage managers that actually write backups to media?
- ❑ Which disk partitions should contain dbspaces?
- ❑ Which disk partitions should contain the **root** dbspace and its mirror?
- ❑ Do users connect to the new database server across a network as well as through shared-memory sessions on a coserver node?

After you answer these questions, create the database server and bring it on-line. The complexity of the database server, the number of coservers, and the number of disks each coserver has determine how long this process takes.

Instructions for creating a customized database server appear in the *Administrator's Guide*.

Deciding on Your Hardware Configuration

You can configure Extended Parallel Server to run on several different hardware configurations. These include:

- single node on single coserver.
- single node on multiple coservers.
- multiple nodes with single coserver on each.
- multiple nodes with multiple coservers on each.

The hardware architecture of your computer and operating system and the intended use of your database server play a role in determining the appropriate number of coservers that you configure. A basic guide for configuring coservers is one coserver per node. However, the following exceptions apply:

- If your platform consists of only one node with up to 10 CPUs, you can take advantage of the single-coserver configuration option for improved performance.
- If your platform is an SMP computer with a large number of CPUs and a large amount of memory, you might be able to obtain improved performance by configuring multiple coservers on your computer.

Important: *For ease of administration, Informix recommends that you use the same hardware configuration for all nodes that make up a platform that supports Extended Parallel Server.*

For more information, see "Supported Coserver Configurations" on page 1-9 and the *Administrator's Guide*.

Configuring Disk Space for Multiple Coservers

Proper disk configuration is possibly the most important task for obtaining optimum performance with VLDBs. Disk I/O requires the longest amount of response time for an SQL operation that scans a large amount of data.

Extended Parallel Server allows parallel access to multiple disks on multiple coservers. Perform the following steps to accommodate multiple coservers:

- Create standard device names across all coservers.
- Set up access to disks across all nodes.

Creating Standard Device Names

Informix recommends using standard device names across all coservers.

Use symbolic links to assign abbreviated standard device names. Use the UNIX *link* command (usually **ln**) to create a link between the character-special device name and another filename.

Execute the UNIX command **ls -l** (**ls -lg** on BSD) on your device directory to verify that both the devices and the links exist. The following example shows links to raw devices. If your operating system does not support symbolic links, use hard links.

```
% ls -lg
crw-rw---   /dev/rxy0h
crw-rw---   /dev/rxy0a
lrwxrwxrwx  /dev/my_root@->/dev/rxy0h
lrwxrwxrwx  /dev/raw_dev2@->/dev/rxy0a
```

Setting Up Disk Access Across Nodes

All nodes that are defined for Extended Parallel Server must be mounted on and exported to the file system on which the **$INFORMIXDIR** directory is installed.

After you install the database server, you must replicate the following utilities on each node:

- **oninit**
- **onmode**
- **onstat**

The directory in which these utilities reside must appear in the search path before **$INFORMIXDIR** on each node.

Setting Permissions, Ownership, and Group

Set read and write permissions for each raw device, as the following example shows:

```
chmod 660 device-name
```

Set the owner and group of the device to **informix**, as the following example shows:

```
chgrp informix device-name
chown informix device-name
```

Checking Product Compatibility

Extended Parallel Server supports the following Informix client products:

- DB-Access
- Informix ESQL/C
- Informix ODBC Driver
- Informix GLS

Other Informix products such as the High-Performance Loader (HPL), Informix Storage Manager (ISM), and database server utilities, are discussed in Chapter 2. Extended Parallel Server also might support additional development tools that this manual does not describe.

For information about specific products, versions, and platforms that Extended Parallel Server supports, see the on-line release notes and machine notes files that are described in "Documentation Notes, Release Notes, Machine Notes" on page 10 of the Introduction. See also the manual for the client product or SQL API that you want to use or the Informix Client Software Developer's Kit for the specific group of products.

DB-Access

DB-Access is a menu-driven utility that is included with Informix database servers. DB-Access lets you connect to the database server and access, modify, and retrieve information. Use DB-Access menus and screens to access and manipulate the data in a relational database and perform a variety of data-management tasks such as organizing, storing, retrieving, and viewing data.

For more information, see the *DB-Access User's Manual*.

Client SDK Products

The Informix Client Software Developer's Kit (Client SDK) package includes several application-programming interfaces (APIs) that developers can use to write applications for Informix database servers in the language with which they are familiar. Informix Connect contains the runtime libraries of the APIs in the Client SDK.

For information about available client products that Extended Parallel Server supports, see your release notes.

Informix OBDC Driver

Informix ODBC Driver is the Informix implementation of the Microsoft Open Database Connectivity (ODBC) standard. It supports SQL statements with a library of C functions that an application calls to implement ODBC functionality. The Informix ODBC Driver API lets you access an Informix database and interact with an Informix database server.

Informix ODBC Driver consists of the following components:

- Informix ODBC libraries, which provide the functions and values for the API
- A driver manager, which provides an interface between an Informix ODBC application and the Informix ODBC Driver and checks parameters and transitions
- The Informix ODBC Driver, which provides an interface between a data source and a driver manager or an application

For more information, see the *Informix ODBC Driver Programmer's Manual*.

Informix ESQL/C

Informix ESQL/C is the Informix implementation of embedded SQL for C that programmers can use to create client applications with database-management capabilities. ESQL/C is an SQL API that lets programmers embed SQL statements directly into a C program to interact with the database server, access databases, manipulate the data in a program, and check for errors.

Informix ESQL/C consists of the following components:

- ESQL/C libraries of C functions, which provide access to the database server
- ESQL/C header files, which provide definitions for the data structures, constants, and macros useful to the ESQL/C program
- **esql**, a command that manages the source-code processing to convert a C file that contains SQL statements into an object file
- ESQL client-interface *dynamic link libraries* (DLLs), which let an ESQL/C application run in a Windows environment

For more information, see the *Informix ESQL/C Programmer's Manual*.

Informix GLS

The Informix GLS library that is provided with the Informix GLS API lets programmers develop internationalized Informix ESQL/C client applications. Informix GLS accesses GLS locales to handle different languages, cultural conventions, and code sets. For information on the GLS feature, see "Global Language Support (GLS)" on page 2-25.

Informix GLS provides procedures, macros, and functions to:

- process single-byte, multibyte, and wide characters and strings.
  - String-processing operations include string traversal, concatenation, copying, and character searching.
  - Character-processing operations include character classification, case conversion, code-set conversion, and character comparison.
- convert date, time, monetary, and number values from and to locale-specific data formats.

  The Informix GLS library supports conversion of a locale-specific string to its internal database representation and formatting of an internal database representation to a locale-specific string.

For more information, see the *Informix GLS Programmer's Manual*. In addition, Informix GLS provides man pages as HTML documentation that you can access with a web browser. The URL must include the full pathname of the directory that your **INFORMIXDIR** environment variable designates, such as **$INFORMIXDIR/doc/gls_api/en_us/0333/index.htm**.

Installing Informix Products

This section provides an overview of how to install Extended Parallel Server and client products.

To install the database server on UNIX, set up the UNIX nodes where you plan to run Extended Parallel Server with the specified memory and disk-space configuration. Load the product files that Informix supplies and use one of the following installation methods:

- A graphical wizard automates the installation steps and configures and initializes a database server after the database server software is installed.
- The manual installation method requires you to type all commands on the UNIX command line.

If you use the command-line installation method and you plan to install more than one Informix product on the same computer, you must install the products in the following order:

1. Client products (such as Informix Client Software Developer's Kit or Informix Connect)
2. Database servers

Complete all installation procedures for each product before you start to install the next product. Do not load the files from another Informix product onto your computer until you complete the current installation.

To preserve product files of earlier versions, create separate directories for each version of your Informix products. If you install multiple versions of an Informix product, set the **INFORMIXDIR** environment variable to the appropriate directory for the version that you want to access.

For detailed information about loading the executables and installing, configuring, and initializing Extended Parallel Server and client products, see the following manuals.

Product	Manual
Extended Parallel Server	*Installation Guide for Informix Extended Parallel Server on UNIX*
Client products	*Informix Client Products Installation Guide for UNIX* and *Installation Guide*

For information about configuring disk space and setting up nodes, see the *Administrator's Guide*. For information about the **INFORMIXDIR** environment variable, see the *Administrator's Guide* and the *Informix Guide to SQL: Reference*. For information about platform-specific requirements, refer to the on-line machine notes.

Configuring the Environment

After you install Extended Parallel Server, you can configure the database server environment. *Configuration* refers to setting specific parameters that affect data processing in the client / server environment.

You must configure your database server environment before you can connect a client application with a database server. The following list identifies the basic configuration requirements:

- Prepare the operating system (page 3-13)
- Set environment variables (page 3-13)
- Prepare the **sqlhosts** file (page 3-14)
- Prepare the ONCONFIG file (page 3-16)
- Allocate and initialize disk space (page 3-19)

This section summarizes these configuration requirements and also outlines the following optional activities:

- Create dbspaces and dbslices (page 3-19)
- Add logical-log files and logslices (page 3-20)
- Create logging and nonlogging tables (page 3-20)

Before you can start configuring the database server, you must configure the operating system appropriately. You might need the assistance of the system administrator for this task.

For a full discussion of requirements, see the *Administrator's Guide*. For information about configuration parameters and files that Extended Parallel Server uses, see the *Administrator's Reference*.

Preparing the Operating System

To prepare your operating system for the database server, set up node names. The basic rule is to configure one coserver per node, with the following exceptions:

- If your platform consists of a single computer with up to 10 CPUs, a single-coserver configuration might make administration easier.
- If your platform is an SMP computer with more than 10 CPUs, or if the physical memory available on the computer is more than twice the size of the virtual address space of a single process, you can improve performance by configuring multiple coservers.
- If your platform is an SMP computer that can be partitioned, you can make administration easier as well as improve performance by partitioning the computer into independent computing subsystems and configuring a single coserver for each subsystem. Each subsystem is regarded as an individual node for configuration purposes.

For more information, see the *Administrator's Guide*.

Setting Environment Variables

This section describes the minimum environment variables that you should set to work with Extended Parallel Server. It also lists some useful optional environment variables.

The following table lists required and useful environment variables and their purpose.

Environment Variable	Restrictions	Use
INFORMIXDIR	**required**	Connect to the database server
INFORMIXSERVER	**required**	Connect to the connection coserver
ONCONFIG	**required**	Connect to the connection coserver
PATH	**required**	Connect to the database server
CLIENT_LOCALE	GLS only	Work with GLS

(1 of 2)

Environment Variable	Restrictions	Use
DBSPACETEMP		Work with dbspaces and dbslices
DB_LOCALE	GLS only	Work with GLS
INFORMIXTERM	UNIX only	Connect to the client
PSORT_NPROCS		Improve query performance
SERVER_LOCALE	GLS only	Work with GLS
TERMINFO	UNIX only	Connect to the client

(2 of 2)

Set UNIX environment variables in one of the following ways:

- At the system prompt
- In an environment-configuration file (**$INFORMIXDIR/etc/informix.rc**)
- In your **.profile** or **.login** file

Tip: *Informix recommends that you set the environment variables in the appropriate start-up file for your shell.*

To override environment variables that were set automatically, use a private environment-variable file (**~/.informix**) or assign new values to environment variables individually.

You can use the **chkenv** utility to check the validity of environment variables.

For detailed information about valid environment variables, see the *Informix Guide to SQL: Reference*, the *Administrator's Guide*, and the *Informix Guide to GLS Functionality*. For information on how certain environment variables can affect performance, see your *Performance Guide*. For information on environment variables that might be required if you want to work with an SQL API such as Informix ESQL/C, see the manual for that product.

The sqlhosts Connectivity File

The connectivity files contain information that allows client/server communication. The database server administrator manages the **$INFORMIXDIR/etc/sqlhosts** file. The system (or network) administrator manages the operating-system files such as network-configuration and network-security files.

The connectivity configuration files must be present for a client application to establish a connection to Extended Parallel Server or to any Informix database server on the network.

An Extended Parallel Server database server is made up of multiple coservers. You can define database server groups in **sqlhosts** to accommodate this architecture. This feature makes it possible to treat multiple related **sqlhosts** entries as one logical entity for establishing or changing client/server connections. Use this feature only to connect to a database server, not for initializing a database server.

The **sqlhosts** file must contain an entry for each coserver. Each entry in **sqlhosts** includes the fields **dbservername**, **nettype**, **hostname**, **servicename**, and **options**. If you use multiple communication protocols for a coserver, you must have a separate entry for each connection type and specify a unique **dbservername**.

The client application looks for the file **$INFORMIXDIR/etc/sqlhosts**. You can use the **INFORMIXSQLHOSTS** environment variable to change the location or name of **sqlhosts**.

For information about the **INFORMIXSQLHOSTS** environment variable, see the *Informix Guide to SQL: Reference*. For detailed information about how to prepare **sqlhosts**, refer to the *Administrator's Guide* and *Administrator's Reference*.

The ONCONFIG Configuration File

The ONCONFIG configuration file contains values for parameters that describe the database server environment. You customize an ONCONFIG file to describe a specific database server environment. The location of the ONCONFIG file is **$INFORMIXDIR/etc/**$**ONCONFIG**.

The database server contains one centralized ONCONFIG configuration file that all coservers share. The ONCONFIG file contains both global and coserver-specific configuration parameters. It also holds values for configuration parameters used for connectivity.

The **onconfig.std** and **onconfig.xps** files in the **$INFORMIXDIR/etc** directory contain initial settings for the configuration parameters and serve as templates for customized configuration files. The template that you use depends on nodes that you use:

- For a multiple-coserver configuration, use the **onconfig.std** file as a configuration template.
- For all other configurations, use the **onconfig.xps** file as a configuration template.

To prepare the ONCONFIG configuration file on UNIX, use a text editor and complete the following steps:

1. Make a copy of the appropriate template file and store the new file in the **$INFORMIXDIR/etc** directory. You can rename the file according to the requirements of your operating system.
2. Edit your new ONCONFIG file to modify the configuration parameters that you decide to change.
3. Set your **ONCONFIG** environment variable to the name of your new ONCONFIG file.

For information about how to prepare the ONCONFIG file and configure Extended Parallel Server as well as a summary of configuration parameters, see the *Administrator's Guide*. For a discussion of configuration parameters and example configuration files, see the *Administrator's Reference*. For information on how configuration parameter settings can affect performance, see your *Performance Guide*. For additional information about the performance impact of certain configuration parameters, especially settings for SMP coserver nodes, see the machine notes file.

Configuring Multiple Coservers

For a multiple-coserver configuration, the database server contains a centralized configuration file that contains the following types of configuration parameters:

- Global

  These parameters apply to all coservers.
- Coserver-specific

  These parameters apply to individual coservers. To configure your database server with multiple coservers, you must include a coserver-specific section in your ONCONFIG file.

  The ONCONFIG file for a single-coserver configuration contains no coserver-specific section.
- Storage-manager-specific

  If your database server uses multiple storage managers, you must define each one in the storage-manager section of the ONCONFIG file.

Tip: *Informix recommends that you use global configuration parameters whenever possible and that you avoid overriding those values on specific coservers.*

Global Configuration Parameters

Global configuration parameters apply to all coservers in your database server. Specify these parameters once at the beginning of the ONCONFIG file. The database server uses the parameter value in the global section of the ONCONFIG file as the default value for that parameter on every coserver.

You can leave most global parameters set to their default values for the initial configuration of the database server. However, you must review and appropriately modify parameters for the following items:

- **Root** and log dbspaces on multiple coservers
- Database server identification
- Processors
- Message files
- Shared-memory size allocation
- ON-Bar backup and restore

Coserver-Specific Configuration Parameters

The parameters that describe each coserver are at the bottom of the ONCONFIG configuration file. When you specify the ROOTSLICE and PHYSSLICE global configuration parameters, you usually need to specify only the following coserver-specific parameters:

- COSERVER
- NODE
- END

You can also override global configuration parameters with the following coserver-specific ones:

- ROOTNAME
- ROOTPATH
- ROOTOFFSET
- ROOTSIZE
- MIRRORPATH
- MIRROROFFSET
- PHYSDBS
- PHYSFILE

Storage-Manager-Specific Configuration Parameters

The parameters that describe each storage manager are in the storage-manager section of the ONCONFIG configuration file. Each storage-manager section begins with the BAR_SM parameter and ends with the END parameter. The BAR_SM parameter cannot be embedded in the coserver-specific section or nested.

Platform-Specific Configuration Parameters

Additional coserver-specific parameters might be listed in the machine notes file for certain UNIX platforms.

Configuration Parameters That Affect Performance

The settings that you choose for various configuration parameters can affect the performance of Extended Parallel Server. The parameter settings affect the following kinds of performance:

- CPU use
- Memory use
- Critical media
- Input/output

For information about the configuration parameters that are listed in this section, see your *Performance Guide*.

Allocating and Initializing Disk Space

You must correctly configure your disks if you want to obtain optimum performance with very large databases (VLDBs). Extended Parallel Server offers the advantage of parallel access to multiple disks that are spread across many coservers.

Tip: *Before you allocate disk space, study the information about disk space in your operating-system administration guide.*

You initialize disk space when you bring Extended Parallel Server on-line for the first time.

For details of how to allocate and initialize disk space, see the *Administrator's Guide* and *Administrator's Reference*.

Creating Dbslices and Dbspaces

Once Extended Parallel Server is initialized, you can create dbslices and dbspaces. See "Database Slices (Dbslices)" on page 2-8. To get maximum performance, you can fragment all tables (except very small tables) across all available coservers, or use cogroups for node partitioning.

Use **onutil** utility commands to create and modify dbslices and dbspaces.

For information about dbslices, dbspaces, and **onutil** see the *Administrator's Guide* and *Administrator's Reference*.

Adding Logical-Log Files and Logslices

Logslices simplify the process of adding and deleting log files by treating sets of them as single entities. See "Logslices" on page 2-9.

Use **onutil** utility commands to add and drop logical-log files and to modify logslices.

For information about log files, logslices, and **onutil**, see the *Administrator's Guide* and *Administrator's Reference*.

Creating Logging and Nonlogging Tables

Extended Parallel Server logs tables by default. However, to accommodate data warehousing and other VLDB applications, the database server provides the following types of tables.

Type of Table	Mode
Operational permanent	Logging
Standard permanent	Logging
Temp temporary	Logging
Raw permanent	Nonlogging
Static permanent	Nonlogging
Scratch temporary	Nonlogging

Use SQL statements to create and change these table types. For more information about supported table types, see "Logging and Nonlogging Permanent and Temporary Tables" on page 2-9.

For SQL statement syntax and use when you create and change tables, see the *Informix Guide to SQL: Syntax*. For details of table characteristics, see the *Administrator's Guide*, *Administrator's Reference*, and *Informix Guide to Database Design and Implementation*.

Starting the Database Server and Initializing Disk Space

To bring an existing instance of Extended Parallel Server on-line, you must log in as user **root** or **informix**. Enter the following command to start all the coservers defined in your ONCONFIG file:

```
xctl -C oninit -y
```

If you are starting a new database server, use the following command to initialize the disk space and bring Extended Parallel Server on-line on all coservers:

```
xctl -C - oninit -iy
```

Important: *When you execute this command, all existing data in the disk space is destroyed. Use the* ***-i*** *flag* only *when you are starting a completely new database server instance.*

To verify that all coservers are on-line, issue the following command to display a one-line summary of the status of each coserver:

```
xctl onstat -
```

To verify that all **root** dbspaces are initialized, issue the following command:

```
xctl onstat -d
```

To shut down all coservers, issue the following command:

```
xctl onmode -ky
```

For specific information about these commands, see the *Administrator's Guide* and *Administrator's Reference*.

Configuring a Client or SQL API Environment

After you install an SQL API or client product, you can configure the product environment. All coservers in your database server can accept client connections. A coserver that accepts client connections is known as a connection coserver.

To identify each connection coserver uniquely, Extended Parallel Server uses dbservernames of the following form:

```
dbservername.coserver-number
```

Before you can compile, link, and run a client application or SQL API, you must set the **INFORMIXTERM** and **TERMINFO** environment variables on the computer on which you have installed the SQL API. See "Setting Environment Variables" on page 3-13.

GLS

If your client application or SQL API uses a nondefault client locale or accesses a database that has a nondefault database locale, you must set GLS environment variables. ♦

For information about environment variables that affect SQL API programs, see the *Informix Guide to SQL: Reference*. For information about GLS environment variables and locales, see the *Informix Guide to GLS Functionality*.

To set up a client connection

1. Specify connectivity configuration parameters in your ONCONFIG file.

   These parameters include COSERVER, DBSERVERNAME, and NETTYPE, and, for multiple communication protocols, DBSERVERALIASES.
2. Set up appropriate entries in the connectivity files on your platform.

   Prepare your **sqlhosts** file to include an entry for each connection coserver. Each **sqlhosts** entry includes the fields **dbservername**, **nettype**, **hostname**, **servicename**, and **options**.
3. Specify connectivity environment variables in your UNIX initialization scripts. See "Setting Environment Variables" on page 3-13.
4. Define a dbserver group for the database server in the **sqlhosts** file.

For specific information about these steps, see the *Administrator's Guide*.

Important: *If no database server resides on the computer where the client application runs, an **sqlhosts** file is required on the host computers of both the client application and the database server.*

Creating a Database

When the database server is on-line, you can connect client applications and SQL APIs to it and begin to create databases. However, before any users can connect to the database server, you must take the following actions:

- Set environment variables that specify the location of the Extended Parallel Server software and the name of the database server for each user process. See "Setting Environment Variables" on page 3-13.
- Decide whether you want any of the databases to be ANSI compliant.

  The major differences between ANSI-compliant databases and databases that are not ANSI compliant lie in the following areas:

  - Transactions and transaction logging
  - Owner naming
  - Privileges on and access to objects
  - Default isolation level
  - Character and decimal data types
  - Escape characters
  - Cursor behavior
  - SQLCODE of the SQL Communications Area (SQLCA)
  - Allowed SQL statements
  - Synonym behavior

For information about ANSI-compliant databases, see the *Informix Guide to Database Design and Implementation*. For information about ANSI syntax in SQL statements, see the *Informix Guide to SQL: Syntax*.

Performing Post-Installation Tasks

Once Extended Parallel Server is up and running, you can perform a number of tasks, as outlined in this section.

Create a Demonstration Database

The DB-Access utility includes scripts that create the demonstration databases that are available with Informix products. These scripts are located in the **$INFORMIXDIR/bin** directory. Many examples in the manuals are based on the demonstration database.

- The **stores_demo** database is a relational database that works with all Informix database servers. Install it with the **dbaccessdemo** script.
- The **sales_demo** database is a dimensional database that provides practice with data-warehousing concepts. Install it with the **dbaccessdemo** script and appropriate SQL command files.

Informix ESQL/C includes an **esqldemo** script that creates a demonstration database. The script is also located in the **$INFORMIXDIR/bin** directory.

For information on how to create and work with a demonstration database, see the *DB-Access User's Manual*. For a description of the structure and contents of the **stores_demo** and **sales_demo** databases, see the *Informix Guide to SQL: Reference*. For information on how to work with the **sales_demo** database, see the *Informix Guide to Database Design and Implementation*. For information on the permissions you need to use the command files, see the *Administrator's Guide*.

Get Error Message Information

Informix software products provide text files that contain all the Informix error messages and their corrective actions. Informix provides scripts that let you display error messages on the screen (**finderr**) or print formatted error messages (**rofferr**).

For information on the error message files and a detailed description of the **finderr** and **rofferr** scripts, see *Informix Error Messages* in Answers OnLine.

Manage Coservers and Cogroups

Use the **xctl** utility to execute Extended Parallel Server utilities on one or more coservers to perform the following management tasks:

- Start and stop coservers

  Use **xctl onmode** commands to manage coservers that are defined in your ONCONFIG configuration file. To bring the database server to on-line or any other mode, you must log in as **root**.

- Define and modify cogroups

  Use the system-defined **cogroup_all** or create a new cogroup with **xctl onutil** commands. If you add a node, you can modify the cogroup to change the coservers it includes.

- Monitor coserver activities

  You can monitor resources across all coservers or on individual coservers. Use **xctl onstat** commands to verify that coservers are in on-line mode, monitor query execution across coservers, and check the status of the root dbspace and chunks across all coservers.

For information about all Extended Parallel Server utilities and how to manage and monitor coservers and cogroups, see the *Administrator's Guide* and *Administrator's Reference*.

Manage Dbslices and Dbspaces

You can manage dbslices and dbspaces across coservers in Extended Parallel Server:

- Create tables in a dbslice

  You can fragment tables across many coservers. Each table fragment resides in a separate dbspace on each coserver. Use the SQL statement CREATE TABLE to create tables fragmented in a dbslice that the **onutil** utility creates.

- Create tables in dbspaces without dbslices

  You can create dbspaces across coservers without defining a dbslice, using information from the ONCONFIG file and **onutil** utility commands.

- Monitor dbspaces and chunks in a dbslice

  Use **xctl onstat** utility commands to monitor dbspaces and chunks that reside on different coservers. Monitor chunks for chunk size and number of free pages to track the disk space that chunks use, chunk I/O activity, and fragmentation.

- Modify mirroring of all **root** dbspaces

  You can use **xctl** commands to change mirroring for all **root** dbspaces on all coservers in your database server.

- Define temporary tables

  Create an explicit temporary table with specific options of the CREATE TABLE or SELECT statement. An implicit temporary table is created as part of Extended Parallel Server processing. A flexible temporary table is an explicit temporary table that the database server creates and for which the database server determines the fragmentation strategy.

For detailed information about utilities and how to manage and monitor dbslices and dbspaces, see the *Administrator's Guide* and *Administrator's Reference*.

Manage Logs and Logslices

You can use Extended Parallel Server utilities to manage log files and logslices:

- Modify logical logs and logslices

  You can use **onutil** commands to create or drop logslices or add to an existing logslice. Use **onutil** or edit the ONCONFIG file to change the size of all log files in a logslice.
- Modify the physical log

  You can change the location or size of a physical log on a specific coserver with **xctl** commands and modifications to the ONCONFIG file.
- Monitor the status of log backups

  Use **xctl onstat** commands to see the status of every log file on every coserver in your Extended Parallel Server database server.

For detailed information about utilities and how to manage and monitor log files and logslices, see the *Administrator's Guide* and *Administrator's Reference*.

Monitor Fragmentation

You can monitor fragmentation across coservers or on an individual coserver from the control workstation. The Extended Parallel Server administrator should monitor the following aspects of fragmentation:

- Data distribution over table fragments

  Query the **sysfragments** table to monitor tblspaces.
- I/O request balancing over fragments

  Monitor I/O request queues for data contained in fragments. Use **xctl onstat** commands to monitor chunks for chunk size, the number of free pages, and tables in a chunk.
- The status of chunks that contain fragments

  Monitor fragments for availability and take appropriate steps when a dbspace that contains one or more fragments fails.

For information about Extended Parallel Server utilities, chunks, fragmentation, and monitoring activities, see the *Administrator's Guide*. For information about the **sysfragments** system catalog table, see the *Informix Guide to SQL: Reference*. For additional information about fragmentation strategy, see the *Informix Guide to Database Design and Implementation* for general guidance and your *Performance Guide* for performance considerations.

Monitor Sessions and Queries

As administrator, you must select the tools that you want to use to monitor database server performance. You can use the following methods to monitor OLTP sessions and DSS queries:

- Extended Parallel Server extensions to the **onstat** utility

  Display global session IDs that let you correlate **onstat** displays from multiple coservers to a specific user session, or display information about a specific query plan on the connection coserver. You also can monitor SQL operator statistics, parallel database queries (PDQ), and resources on a single coserver or across multiple coservers.
- The SQL statement SET EXPLAIN

  Execute the SET EXPLAIN ON statement before you enter the SELECT statement for that query to display the chosen query optimizer plan. View the explanation of each subsequent query plan in the **sqexplain.out** file or the file that you have so designated.

For information about **onstat** and other utilities, see the *Administrator's Reference*. For information about performance monitoring, see your *Performance Guide*. For information about the SQL statement SET EXPLAIN, see the *Informix Guide to SQL: Syntax*.

Control Storage of Data

A database server administrator controls where Extended Parallel Server stores data. An administrator can improve performance by storing high-access tables or critical media (**root** dbspace, physical log, and logical log) on the fastest disk drives. By storing critical media and data on separate physical devices, the administrator ensures that when one of the disks that holds noncritical media fails, the failure affects the availability of data on that disk only.

For information about how data storage affects performance, see your *Performance Guide*.

Performing Standard Administrative Tasks

In addition to the activities outlined in "Performing Post-Installation Tasks" on page 3-24, the database server administrator should routinely perform the following tasks after Extended Parallel Server is initialized:

- Check that users have set the correct environment variables.

  Every user of an Informix product must have the correct environment variables set to work with the database server and client applications and, if needed, to use GLS locales.
- Create and load tables that are fragmented across and in coservers.

  After you have created dbslices and dbspaces, you can create tables that are fragmented across multiple coservers. The high-performance load and unload feature lets you load multiple table fragments in parallel across multiple coservers.
- Review the Extended Parallel Server configuration parameters.

  You can examine the configuration of a database server in various ways.
- Migrate data that was created on other Informix database servers.

  Use a supported unload utility and import the data to Extended Parallel Server with the high-performance loader.
- Warn the system administrator about **cron** jobs.

  Some UNIX systems run **cron** jobs that routinely delete all files from the **/tmp** directory.

For complete information about these and other Extended Parallel Server administrative tasks, see the *Administrator's Guide* and *Administrator's Reference*.

Monitoring and Maintaining Performance

All Extended Parallel Server users are responsible to some degree for maintaining good performance. This section highlights those responsibilities, with an emphasis on database server administrator tasks.

Application developers should:

- carefully design applications to use the concurrency and sorting facilities that the database server provides, rather than attempt to implement similar facilities in the application.
- keep the scope and duration of locks to a minimum to avoid contention for database resources.
- include routines in applications that, when temporarily enabled at runtime, allow the database server administrator to monitor response times and transaction throughput.

Database users should:

- pay attention to performance and report problems to the database server administrator promptly.
- be courteous when scheduling large, decision-support queries and request as few resources as possible to get the work done.

The database server administrator should:

- be aware of all performance-related activities that occur.
- educate users about the importance of performance, how performance-related activities affect them, and how they can assist in achieving and maintaining optimal performance.
- pay attention to:
  - how tables and queries affect the overall performance of the database server.
  - the placement of tables and fragments.
  - how the distribution of data across coservers and disks affects performance.

Once Extended Parallel Server is up and running, you might, as database server administrator, be responsible for maintaining the optimum performance of the database server, coservers, and database applications. To accomplish this goal, you must:

- establish performance goals.
- create a performance history.
- monitor performance at scheduled times as well as ongoing system performance.
- identify symptoms of performance problems.
- measure the following attributes:
  - throughput
  - response time
  - cost per transaction
  - resource utilization
- evaluate results and make appropriate adjustments.
- tune configurations.

For complete information on performance-tuning issues and methods that are relevant to daily database server administration and query execution, see your *Performance Guide*.

Monitoring OLTP Sessions and DSS Queries

The kinds of data that a database server administrator needs to collect depend on the kinds of applications that run on your system. The causes of performance problems on systems used for OLTP applications are different than the causes of problems on systems that are used mainly for DSS query applications.

You can use command-line utilities such as **xctl onstat** to monitor OLTP sessions and DSS queries. In addition, you can monitor DSS queries with the SQL statement SET EXPLAIN.

Creating a Performance History

You can use tools from the host operating system and run command-line utilities at regular intervals from scripts or batch files to monitor Extended Parallel Server performance. You also can use graphical interface performance monitoring tools to monitor critical aspects of performance as queries and transactions are performed.

Using Operating-System Tools

The database server relies on the operating system of the host computer to provide access to system resources such as the CPU, memory, and various unbuffered disk I/O interfaces and files. Each operating system has its own set of utilities for reporting how system resources are used. Different implementations of some operating systems have monitoring utilities with the same name but different options and informational displays.

You might be able to use some of the following typical UNIX operating-system resource-monitor utilities.

UNIX Utility	Description
vmstat	Displays virtual-memory statistics.
iostat	Displays I/O utilization statistics.
sar	Displays a variety of resource statistics.
ps	Displays active process information.

To capture the status of system resources at regular intervals, use scheduling tools that are available with your host operating system (for example, **cron**) as part of your performance monitoring system.

Your operating system might also provide graphical monitoring tools that you can use to monitor resources dynamically across coservers. For example, **3dmon** displays a variety of resource statistics in a three-dimensional graph.

For information on how to monitor your operating-system resources, consult your operating-system reference manual or your system administration guide.

Using Command-Line Utilities

In Extended Parallel Server, you can use command-line utility programs as arguments to the **xctl** utility to collect performance data from all coservers in the database server, or from specified coservers, and display it in a single output report. You can also run the command-line utilities on individual coservers to retrieve performance information for that coserver only.

The **xctl** utility allows you to run coserver-specific utilities on multiple coservers and execute operating-system commands on multiple nodes. You can include a coserver-specific utility such as **onlog** or **onstat**, or an operating-system command in an **xctl** command line, along with syntax to designate the nodes or coservers on which the command is to run.

Use the following utilities with the **xctl** utility to capture database server performance:

Utility or Tool	Description
onlog	Used with **xctl**, displays all or selected portions of the logical log or checks the status of the logical logs on all coservers. **onlog** can display information from selected log files, the entire logical log, or an archive tape of previous log files.
onstat	Used with **xctl**, arguments display information about the current status and activities of the database server on all or specified coservers. When run without **xctl**, **onstat** displays information about the local coserver only.
SMI	Lets you use SQL SELECT statements to query the system-monitoring interface from your application. The SMI tables are a collection of tables and pseudo-tables in the sysmaster database that maintains dynamically updated information about the operation of the database server. The tables are constructed in memory but are not recorded on disk.
DB-Access	Used with **cron** and SQL scripts to query SMI tables at regular intervals, use **cron** jobs and SQL scripts with DB-Access.

For more information on **onlog**, **onstat**, and other Extended Parallel Server utilities, see "Database Server Utilities" on page 2-16. For information on SMI and other utilities, see the *Administrator's Reference*. For information about SQL scripts, see the *DB-Access User's Manual*. Using Database Server Utilities

Extended Parallel Server provides command-line utilities to capture snapshot information about your configuration and performance. The database server also provides the system-monitoring interface (SMI) for monitoring performance from your application.

You can use these utilities regularly to gather information to identify and manage your high-impact activities or to adjust your database server or operating-system configuration.

The following utilities help you capture database server performance:

Utility	Description
DB-Access	Used with **cron**, SQL scripts, and the SMI, regularly queries SMI tables.
onlog	Invoked by the **cron** scheduler, regularly captures performance-related information and builds a historical performance profile of an Extended Parallel Server application.
onstat	Invoked by the **cron** scheduler, regularly captures performance-related information and builds a historical performance profile of an Extended Parallel Server application.

For information about SQL scripts and DB-Access, see the *DB-Access User's Manual*. For information about **onlog** and **onstat**, see the *Administrator's Reference*.

Chapter

4 Using the Documentation

In This Chapter

This chapter provides an overview of Extended Parallel Server users and the tasks that each type of user is likely to perform. It includes a matrix of those tasks and where to find relevant information throughout the Informix documentation set.

This chapter also contains an alphabetical list of the manuals that are provided with the database server software and a list of the documentation for some related Informix client products.

Tip: *For current information about what products Extended Parallel Server, Version 8.3, supports, see the release notes. For additional information that postdates the manuals, see the documentation notes.*

Who Uses Extended Parallel Server?

The following major groups use Extended Parallel Server:

- Extended Parallel Server administrators and operators
- Database administrators
- Programmers and application developers
- Database users

Database Server Administrators and Operators

If you are an Extended Parallel Server administrator, you are responsible for the installation, maintenance, administration, and operation of an entire Extended Parallel Server database server that might manage many individual databases and include many coservers.

If you are an Extended Parallel Server operator, you are responsible for backing up and restoring databases and for carrying out similar routine tasks associated with database server administration.

In Figure 4-1 on page 4-6, look for the term **ADMIN** in the **Role** column to identify database server administrator tasks and the books that will help you complete your tasks.

Database Administrators

If you are a database administrator (DBA), you are primarily responsible for such tasks as creating and managing access control for databases. You use SQL statements to grant and revoke privileges to ensure that the correct individuals are able to perform the actions they need to and that untrained or unscrupulous users are kept from performing potentially damaging or inappropriate activities.

In Figure 4-1 on page 4-6, look for the term **DBA** in the **Role** column to identify database administrator tasks and the books that will help you complete your tasks.

Programmers and Application Developers

If you develop client applications, Extended Parallel Server offers a number of possibilities for data management, data warehousing, multimedia, isolation levels, and so on, using an SQL-based relational database.

In Figure 4-1 on page 4-6, look for the term **DEV** in the **Role** column to identify application developer tasks and the books that will help you complete your tasks.

Database Users

If you are a database user, you access, insert, update, and manage information in databases with SQL, which is often embedded in client applications.

In Figure 4-1 on page 4-6, look for the term **USER** in the **Role** column to identify database user tasks and the books that will help you complete your tasks.

Task-Documentation Matrix

The task-documentation matrix in Figure 4-1 provides a quick reference to all the documentation that is provided with Extended Parallel Server. The matrix includes the following columns:

- **If You Want To**. A task you might want to perform.
- **Manual**. The primary book that contains information to help you perform the task.
- **Role**. The person most likely to perform the task. (**ALL** indicates that all types of users can benefit from the documentation listed.)

Use the matrix to associate specific tasks with the manuals that will help you perform the tasks. For a list of all the Extended Parallel Server documentation, see “The Documentation Set” on page 4-14 and “Manuals for Additional Supported Products” on page 4-17.

Figure 4-1
Task-Documentation Matrix

If You Want To:	Manual	Role
Install Extended Parallel Server in different environments	*Installation Guide for Informix Extended Parallel Server on UNIX*	ADMIN
Understand Extended Parallel Server architecture Configure Extended Parallel Server Learn administrator tasks and tools Understand client/server communications and multiple residency Initialize Extended Parallel Server and manage operating modes Manage database-logging status, logical-log files, and the physical log Manage virtual processors, shared memory, disk space, and fragmentation Identify sources of information for monitoring an Informix database server Perform distributed transactions Perform mirroring operations and consistency checking Perform parallel database queries (PDQ) Recover manually from a failed two-phase commit Create and drop cogroups, dbslices, dbspaces, logical logs, and logslices Alter dbspaces Work with external tables	*Administrator's Guide*	ADMIN

(1 of 8)

If You Want To:	Manual	Role
Use the SMI tables of the **sysmaster** database to monitor Extended Parallel Server Locate complete information on all Extended Parallel Server configuration parameters Use utilities such as **onlog**, **onstat**, and **onutil** to perform administrative tasks Work with the **oninit** utility Use the **onmode -I** option to collect diagnostic information Interpret logical-log records and message-log messages Understand database server disk structures and storage See a list of the files that that you use when you configure and use Extended Parallel Server	*Administrator's Reference*	ADMIN
Create logging and nonlogging tables for data warehousing	*Administrator's Guide* *Informix Guide to SQL: Syntax*	ADMIN
Configure and use the ON-Bar backup and restore system Copy your data and logical logs as insurance against lost or corrupted data Back up the logical log Back up dbspaces and dbslices Use the **archecker** utility to verify backed-up objects before you use them to restore data Restore data Monitor the status of storage objects	*Backup and Restore Guide*	ADMIN
Connect your Informix database server to storage devices for backup and restore operations Manage backup media and storage devices for all ON-Bar backups and restores Track the location of all backup data Move backup data through a managed life cycle Provide disaster recovery for an Extended Parallel Server instance	*Informix Storage Manager Administrator's Guide*	ADMIN

(2 of 8)

If You Want To:	Manual	Role
Identify the database activities that affect critical resources	*Performance Guide*	ADMIN
Identify and monitor system resources and costs (CPU, memory, disk) that are critical to query performance		
Monitor Extended Parallel Server performance with the coserver monitor and query monitor		
Eliminate performance bottlenecks, for example, by balancing the load on system resources or adjusting configuration parameters		
Improve the performance of a query		
Use the Resource Grant Manager (RGM) to manage resources for parallel database queries (PDQ)		
Choose the fragmentation strategy that will maximize performance		
Manage GK indexes, bitmap indexes, and B-tree indexes		
Use secondary access methods (B-trees and R-trees)		
Control placement and size of tables, fragments, and table extents		
Configure and operate Extended Parallel Server to achieve optimum performance with OLTP and DSS applications		
Use data-warehousing and data-mart functionality to perform data mining to maximum advantage		
Migrate to Extended Parallel Server from an earlier version of an Informix database server	*Informix Migration Guide*	ADMIN
Move data between different physical equipment (computer and storage devices) and different operating systems		
Move data between database servers that have different language support		
Work with these utilities: **dbexport**, **dbimport**, **dbload**, **dbschema**, **onload**, **onunload**, **onmode -b**		
Revert from Extended Parallel Server to an earlier version of an Informix database server		

(3 of 8)

If You Want To:	Manual	Role
Use the Informix SNMP subagent to extract information from an Informix database server and pass that information to a network manager Perform low-impact sampling of database server performance Interpret the contents of application, Informix, and RDBMS Management Information Bases (MIBs) Use SNMP applications to remotely monitor the status of all Informix database servers, operating systems, routers, printers, and other devices on a network	*Informix SNMP Subagent Guide*	ADMIN
Work with these utilities:		
oninit, **onlog**, **onmode**, **onstat**, **onutil**, **xctl**	*Administrator's Guide*	ADMIN
dbexport, **dbimport**, **dbload**, **dbschema**, **onmode -b**	*Informix Migration Guide*	DBA
Use the high-performance loader to load and unload tables and perform data-format conversions	*Administrator's Guide*	DBA
Work with the tables in the **sysmaster** database	*Administrator's Reference*	DBA
Understand the effect of database design and use on performance	*Performance Guide*	DBA
Design data models and implement relational and dimensional databases for Extended Parallel Server Understand data-warehousing and data-mart concepts Identify and define data and data objects for Informix databases Use supported data types and SQL to implement a database Define a fragmentation strategy or distribution schema Work with the **sales_demo** database Grant and limit access to a database	*Informix Guide to Database Design and Implementation*	DBA

(4 of 8)

If You Want To:	Manual	Role
Perform data-definition tasks such as specifying the data types for table columns Perform data-management tasks such as storing, viewing, and changing table data Connect to one or more database servers and transfer data Verify database server status Create databases and run one-time queries Execute SQL statements and SPL routines Display system catalog tables and the Information Schema	*DB-Access User's Manual*	DBA
Generate Information Schema views Use the system catalog tables to track objects Assign data types to columns	*Informix Guide to SQL: Reference*	DBA
Create and manage database access Compose correct SQL statements Write procedures with SPL and store them in a database	*Informix Guide to SQL: Syntax*	DBA
Query and modify data in a relational database Create and use SQL statements and SPL routines	*Informix Guide to SQL: Tutorial*	DBA
Execute and debug SQL statements and SPL routines Create databases, run one-time queries, test database applications that you intend to store for use in a production environment Access, modify, and retrieve information from Informix database servers Verify database server status Connect to one or more databases Display system catalog tables and the Information Schema	*DB-Access User's Manual*	DEV
Assign data types to columns	*Informix Guide to Database Design and Implementation* *Informix Guide to SQL: Reference* *Informix Guide to SQL: Syntax* *Informix Guide to SQL: Tutorial*	DEV

(5 of 8)

If You Want To:	**Manual**	**Role**
Set environment variables Use supported data types in programs Use the system catalog tables to track objects Find a description of the tables in the **stores_demo** database Find a description of the tables in the **sales_demo** database	*Informix Guide to SQL: Reference*	DEV
Use a primary access method Compose correct SQL statements Write procedures with SPL and store them in a database Perform parallel database queries (PDQ)	*Informix Guide to SQL: Syntax*	DEV
Compose SELECT statements Query and modify data in a relational database Use embedded SQL in programs Program in a multiuser environment Create and use SPL routines	*Informix Guide to SQL: Tutorial*	DEV
Create data types	*Informix Guide to SQL: Tutorial* *Informix Guide to Database Design and Implementation*	DEV
Use Informix ODBC Driver to access Informix databases with SQL and interact with an Informix database server Create custom applications with Informix ODBC Driver	*Informix ODBC Driver Programmer's Manual*	DEV
Embed SQL statements directly into C programs	*Informix ESQL/C Programmer's Manual*	DEV
Use the GLS feature that lets Informix SQL APIs and database servers handle different languages, cultural conventions, and code sets Specify a nondefault locale	*Informix Guide to GLS Functionality*	DEV
Internationalize ESQL/C programs with Informix GLS	*Informix GLS Programmer's Manual*	DEV
Learn how GLS affects database server migration	*Informix Migration Guide*	DEV

(6 of 8)

If You Want To:	Manual	Role
Learn how GLS affects database server migration	*Informix Migration Guide*	DEV
Find a list of all the environment variables that Informix database servers support	*Informix Guide to SQL: Reference* *Informix Migration Guide*	DEV
Set environment variables Use supported data types Understand the Information Schema Learn the structure and contents of the tables in the **stores_demo** database Learn the structure of the tables in the **sales_demo** database	*Informix Guide to SQL: Reference*	USER
Compose correct SQL statements Learn the categories of SQL statements Write procedures with SPL and store them in a database	*Informix Guide to SQL: Syntax*	USER
Compose basic and advanced SELECT statements Query and modify data in a relational database Use embedded SQL in programs	*Informix Guide to SQL: Tutorial*	USER
Invoke the DB-Access utility Use menus, screens, SQL statements, and SPL routines to view, access, retrieve, store, and modify data in a database Connect to or create one or more databases and transfer data between a database and external text files Display information about a database and verify database server status Perform one-time queries that you execute once or infrequently Practice the statements and examples provided with the demonstration databases	*DB-Access User's Manual*	USER
Access, retrieve, and modify data from a database Invoke the DB-Access utility Work with a demonstration database	*DB-Access User's Manual*	USER
Compose correct SQL statements	*Informix Guide to SQL: Syntax*	USER

(7 of 8)

If You Want To:	Manual	Role
Install client products that Extended Parallel Server supports	*Informix Client Products Installation Guide for UNIX*	ALL
Set the appropriate environment variables for your Informix product Acquaint yourself with terms used in Informix database server manuals	*Informix Guide to SQL: Reference*	ALL
Work with the relational (**stores_demo**) and dimensional (**sales_demo**) demonstration databases that are provided with Informix products	*DB-Access User's Manual*	ALL
Find corrective actions to error messages	*Informix Error Messages*	ALL

(8 of 8)

The Documentation Set

This section summarizes the documentation that is distributed with Extended Parallel Server, Version 8.3. Figure 4-2 lists manuals alphabetically.

Figure 4-2
Manuals in the Documentation Set

Book Title	**Description**
Administrator's Guide for Informix Extended Parallel Server	This guide for system and database server administrators discusses the features, concepts, procedures, and syntax for managing Extended Parallel Server. It is intended to help you understand, configure, and use the database server.
DB-Access User's Manual	This guide describes how to use the DB-Access utility to access, modify, and retrieve information from Informix database servers. It provides information on how to use the interface for SQL statements and SPL routines to perform data-definition and data-management tasks. The guide includes the SQL command files for the demonstration databases that are shipped with Informix database servers.
Getting Started with Informix Extended Parallel Server	This guide provides an overview of Extended Parallel Server, summarizes important features of Informix products, and provides information to help you use the documentation that is included with Informix database server products.
Informix Administrator's Reference	This reference manual for system and database server administrators provides the syntax and reference information for the Extended Parallel Server utilities, **sysmaster** tables and SMI, disk structures, and configuration parameters. It also discusses files that the database server uses, message-log messages, trapping errors, and how to read logical-log records.
Informix Backup and Restore Guide	This guide and reference manual explains the concepts and methods you need to back up and restore data with the ON-Bar backup and restore system. It includes information about the **archecker** utility.

(1 of 3)

Book Title	Description
Informix Error Messages	This Answers OnLine product provides a complete list of Informix-specific error messages and describes their corrective actions for current Informix products and earlier Informix products that are still supported. It also describes how to use the **finderr** and **rofferr** scripts to read error messages on-line. Use the UNIX **finderr** and **rofferr** utilities to locate the latest information on error messages.
Informix Guide to Database Design and Implementation	This guide documents how to design and implement databases for Informix database servers. It describes the fundamental ideas and terminology for planning, implementing, and using a relational or dimensional database management system. The guide discusses tasks that the DBA usually performs and addresses data definition with Informix databases. It describes how to design different data models, such as data warehousing and data marts, and how to use the supported data types and SQL to implement a database.
Informix Guide to GLS Functionality	This manual describes the Global Language Support (GLS) feature that is available in Informix products. The GLS feature allows Informix client products and Informix database servers to handle different languages, cultural conventions, and code sets. This manual describes only the language-related topics that are unique to GLS.
Informix Guide to SQL: Reference	This manual describes the Informix system catalog tables, the data types supported by Extended Parallel Server, and Informix and common environment variables that you might need to set. It contains a complete description of the **stores_demo** demonstration database and the structure of the **sales_demo** database, as well as a glossary of terms that Informix manuals use.
Informix Guide to SQL: Syntax	This manual contains the complete syntax descriptions for SQL and Stored Procedure Language (SPL) statements and segments.
Informix Guide to SQL: Tutorial	This tutorial includes instructions for using basic and advanced SQL to query and modify data in a relational database. It also describes the fundamental ideas and terminology for planning, implementing, and using a relational database management system.

(2 of 3)

Book Title	Description
Informix Migration Guide	This manual describes the tasks that you perform when you move data from one location to another and when you migrate existing Informix databases to various Informix database servers. It discusses such database server utilities as **dbexport**, **dbimport**, **dbload**, **dbschema**, and **onmode -b**.
Informix SNMP Subagent Guide	This manual introduces Simple Network Management Protocol (SNMP) and describes the **onsnmp** subagent that provides information about Informix database servers to network-management tools. The guide also documents the Management Information Bases (MIBs) that specify the information that the onSNMP program provides to the network-management tools. It includes a glossary of terms used in the guide.
Informix Storage Manager Administrator's Guide	This guide describes how to connect your Informix database server to storage devices for backup and restore operations, and how to manage backup media with the Informix Storage Manager.
Installation Guide for Informix Extended Parallel Server on UNIX	This guide contains instructions for installing Informix Version 8.3 database server products on computers that run the UNIX operating system. It also describes common installation problems and indicates how to solve them.
Performance Guide for Informix Extended Parallel Server	This manual explains how to configure and operate Extended Parallel Server to improve overall system throughput and how to improve the performance of SQL queries.

(3 of 3)

Manuals for Additional Supported Products

This section summarizes the Informix client and Client SDK manuals that you can use when you work with the Extended Parallel Server, Version 8.3. Figure 4-3 lists manuals for Informix client products alphabetically.

Figure 4-3
Client Manuals for Extended Parallel Server

Book Title	Description
Informix ODBC Driver Programmer's Manual	This user guide and reference manual discusses the features that make up the Informix implementation of the Microsoft Open Database Connectivity (ODBC) interface. The guide explains how to use Informix ODBC Driver to access Informix databases, manipulate the data in your program, interact with the database server, and check for errors.
Informix Client Products Installation Guide for UNIX	This guide describes how to install Informix client products on computers that run the UNIX operating system. The guide also lists GLS support files and describes how to solve installation problems.
Informix ESQL/C Programmer's Manual	This manual explains how to use Informix ESQL/C, the Informix implementation of embedded SQL for C, to create client applications with database-management capabilities. The guide is a complete guide to the features of ESQL/C that let programmers interact with the database server, access databases, manipulate the data in programs, and check for errors.
Informix GLS Programmer's Manual	This manual describes the Informix GLS application-programming interface that is available in Informix products and that gives ESQL/C programmers and developers the ability to write or change programs to work with different locales.

Index