

INFORMIX[®]-SE

Database Server

Administrator's Guide

Version 7.2
April 1996
Part No. 000-7895A

Published by INFORMIX® Press

Informix Software, Inc.
4100 Bohannon Drive
Menlo Park, CA 94025

The following are worldwide trademarks of Informix Software, Inc., or its subsidiaries, registered in the United States of America as indicated by “®,” and in numerous other countries worldwide:

INFORMIX®, C-ISAM®, INFORMIX®-OnLine Dynamic Server™

The following are worldwide trademarks of the indicated owners or their subsidiaries, registered in the United States of America as indicated by “®,” and in numerous other countries worldwide:

Adobe Systems Incorporated: PostScript®
Novell, Inc.: NetWare®
X/OpenCompany Ltd.: UNIX®, X/Open®

Some of the products or services mentioned in this document are provided by companies other than Informix. These products or services are identified by the trademark or servicemark of the appropriate company. If you have a question about one of those products or services, please call the company in question directly.

Documentation Team: Brian Deutscher, Geeta Karmarker, Mary Kraemer, Patrice O’Neill, Eileen Wollam

Copyright © 1981-1996 by Informix Software, Inc. All rights reserved.

No part of this work covered by the copyright hereon may be reproduced or used in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems—without permission of the publisher.

To the extent that this software allows the user to store, display, and otherwise manipulate various forms of data, including, without limitation, multimedia content such as photographs, movies, music and other binary large objects (blobs), use of any single blob may potentially infringe upon numerous different third-party intellectual and/or proprietary rights. It is the user's responsibility to avoid infringements of any such third-party rights.

RESTRICTED RIGHTS LEGEND

Software and accompanying materials acquired with United States Federal Government funds or intended for use within or for any United States federal agency are provided with “Restricted Rights” as defined in DFARS 252.227-7013(c)(1)(ii) or FAR 52.227-19.

Table of Contents

Introduction

About This Manual	3
Organization of This Manual	3
Types of Users	4
Software Dependencies	4
Demonstration Database	5
New Features of This Product	8
Conventions	8
Typographical Conventions	9
Icon Conventions	10
Command-Line Conventions	11
Sample-Code Conventions	14
Additional Documentation	14
Printed Documentation	15
On-Line Documentation	16
Related Reading	17
Compliance with Industry Standards	18
Informix Welcomes Your Comments	t19

Chapter 1

Installation and Initial Configuration

Defining a Database Management System	1-3
Planning for SE	1-4
Configuring Hardware	1-4
Placing Active Tables and Files on the Disk	1-4
Considering SE Limits	1-5
Installing SE	1-6
Preparing SE Connections	1-6
Understanding Version 7.2 SE Configurations	1-7
Setting Environment Variables	1-8
Understanding the Communication Files	1-9
Building the sqlhosts File	1-12
Examples of Client/Server Connections	1-18

	Connecting with Different Versions	1-26
	Starting the sqlexecd Daemon	1-35
	Using an NFS-Mounted Directory	1-37
	How Does a Client Application Connect to a Database Server?	1-38
Chapter 2	INFORMIX-SE System Architecture	
	SE Program Files	2-3
	SE System Files	2-4
	The .dat File	2-5
	The .idx File	2-6
	Allocating Space for .dat and .idx Files	2-6
	Transaction-Log Files	2-6
	Allocating Space for the Transaction-Log File	2-8
	Audit-Trail Files	2-8
	Permissions of Database Files and Directories	2-10
	Determining If a Database Is ANSI Compliant	2-12
Chapter 3	Basic Administration and Maintenance	
	Monitoring Disk-Space Use	3-3
	Maintaining Data Integrity	3-4
	Transaction-Log Files	3-4
	Audit-Trail Files	3-6
	Creating Backups	3-6
Chapter 4	INFORMIX-SE Indexing	
	B+ Tree Organization	4-3
	Searching for a Row	4-5
	Adding Keys	4-6
	Removing Keys	4-11
	Index-Table Structure	4-12
	Multiple Indexes	4-13
	Index-Table Formats	4-15
Chapter 5	Symptoms and Solutions	
	Permission Problems	5-3
	Corruption Problems	5-4
	Operating-System Failures	5-4
	Premature Termination of an sqlexec Process	5-5
	Physical Disk Corruption	5-5
	Lost and Damaged Index and Data Files	5-6
	Transaction-Log Corruption	5-7

Disk Fragmentation	5-8
Practices to Avoid	5-8
Performance Tuning	5-9

Chapter 6

INFORMIX-SE Utilities

The secheck Utility	6-3
Choosing Not to Specify the -n or -y Option	6-5
Simple Example	6-6
Output.	6-6
Parenthetical Values	6-7
Printing a Long List of Index Key Values	6-8
An Example Using the -l Option	6-10
Converting Index-Node Size with the -s Option	6-11
Causes for secheck Failure	6-11
The selog Utility	6-12
Displaying the Contents of a Transaction Log	6-12
Use and Output.	6-20

Index

Introduction

About This Manual	3
Organization of This Manual	3
Types of Users	4
Software Dependencies	4
Demonstration Database	5
New Features of This Product	8
Conventions	8
Typographical Conventions	9
Icon Conventions	10
Comment Icons	10
Compliance Icons	10
Command-Line Conventions	11
Sample-Code Conventions	14
Additional Documentation	14
Printed Documentation	15
On-Line Documentation.	16
Error Message Files	16
Release Notes, Documentation Notes, Machine Notes	17
Related Reading	17
Compliance with Industry Standards	18
Informix Welcomes Your Comments.	19

T

his chapter introduces the *INFORMIX-SE Administrator's Guide*. Read this chapter for an overview of the information provided in this manual and for an understanding of the conventions used throughout this manual.

The INFORMIX-SE database server is ideally suited for small- to medium-sized database applications. Informix based the SE database server on the indexed sequential access method (ISAM), a library of C language calls that work with UNIX to manipulate database files. ISAM uses an index to access data instead of performing a scan on the table to access data. SE works automatically and transparently and does not require any special instructions in your database applications or programs.

About This Manual

The *INFORMIX-SE Administrator's Guide* is a complete guide to the operating environment of the SE database server. This manual explains how to configure and use SE. It also explains how to use the SE utilities.

Organization of This Manual

This manual includes the following chapters:

- This Introduction provides an overview of the manual, describes the documentation conventions used, introduces the demonstration database from which the product examples are drawn, describes the ASCII and PostScript error message files, and lists the new features in SE, Version 7.2.
- [Chapter 1, "Installation and Initial Configuration,"](#) provides background information on planning and configuring an SE system.

- [Chapter 2, “INFORMIX-SE System Architecture,”](#) describes the SE program files and system files and the permissions on database files and directories.
- [Chapter 3, “Basic Administration and Maintenance,”](#) describes disk usage and the maintenance of data integrity.
- [Chapter 4, “INFORMIX-SE Indexing,”](#) describes the organization, structure, and format used to index ISAM files.
- [Chapter 5, “Symptoms and Solutions,”](#) provides tips on troubleshooting.
- [Chapter 6, “INFORMIX-SE Utilities,”](#) describes the command-line utilities available for performing administrative tasks.

Types of Users

This manual is written for SE system administrators who are responsible for the following tasks:

- Initial installation and configuration of SE
- General system administration and maintenance
- Troubleshooting of system problems

Software Dependencies

This manual assumes that you are using SE, Version 7.2, as your database server. Informix software can reside on a single computer or on multiple computers across a network. The following Informix software must reside on your computer system:

- An SE database server, which you install on your computer or on another computer over a network
- Either an Informix application development tool, such as INFORMIX-NewEra; an SQL application programming interface (API), such as INFORMIX-ESQL/C; or the DB-Access utility, which Informix ships as part of your database server

The application development tool, SQL API, or DB-Access enables you to compose queries, send queries to the database server, and view the results that the database server returns. You can use DB-Access to try out the SQL statement described in this guide.

Demonstration Database

The DB-Access utility, which is provided with your Informix database server products, includes a demonstration database called **stores7** that contains information about a fictitious wholesale sporting-goods distributor. The sample command files that make up a demonstration application are also included.

Most examples in this manual are based on the **stores7** demonstration database. The **stores7** database is described in detail and its contents are listed in Appendix A of the *Informix Guide to SQL: Reference*.

The script that you use to install the demonstration database is called **dbaccessdemo7** and is located in the **\$INFORMIXDIR/bin** directory. The database name that you supply is the name given to the demonstration database. If you do not supply a database name, the name defaults to **stores7**. Use the following rules for naming your database:

- Names can have a maximum of 18 characters for INFORMIX-OnLine Dynamic Server databases and a maximum of 10 characters for SE databases.
- The first character of a name must be a letter or an underscore (_).
- You can use letters, characters, and underscores (_) for the rest of the name.
- DB-Access makes no distinction between uppercase and lowercase letters.
- The database name must be unique.

When you run **dbaccessdemo7**, you are, as the creator of the database, the owner and Database Administrator (DBA) of that database.

If you install your Informix database server according to the installation instructions, the files that constitute the demonstration database are protected so that you cannot make any changes to the original database.

You can run the **dbaccessdemo7** script again whenever you want to work with a fresh demonstration database. The script prompts you when the creation of the database is complete and asks if you would like to copy the sample command files to the current directory. Enter **N** if you have made changes to the sample files and do not want them replaced with the original versions. Enter **Y** if you want to copy over the sample command files.

To create and populate the stores7 demonstration database

1. Set the **INFORMIXDIR** environment variable so that it contains the name of the directory in which your Informix products are installed.
2. Set **INFORMIXSERVER** to the name of the default database server.

The name of the default database server must exist in the **\$INFORMIXDIR/etc/sqlhosts** file. (For a full description of environment variables, see Chapter 4 of the [Informix Guide to SQL: Reference](#).) For information about the **sqlhosts** file, refer to “[The sqlhosts File](#)” on page 1-12 of this manual.

3. Create a new directory for the SQL command files. Create the directory by entering the following command:

```
mkdir dirname
```

4. Make the new directory the current directory by entering the following command:

```
cd dirname
```

5. Create the demonstration database and copy over the sample command files by entering the **dbaccessdemo7** command.

To create the database without logging, enter the following command:

```
dbaccessdemo7 dbname
```

To create the demonstration database with logging, enter the following command:

```
dbaccessdemo7 -log dbname
dbaccessdemo7 -log dbname -dbspace dbspacename
```

If you are using SE, a subdirectory called **dbname.dbs** is created in your current directory and the database files associated with **stores7** are placed there. You will see both data (**.dat**) and index (**.idx**) files in the **dbname.dbs** directory. (If you specify a dbspace name, it is ignored.)

To use the database and the command files that have been copied to your directory, you must have UNIX read and execute permissions for each directory in the pathname of the directory from which you ran the **dbaccessdemo7** script. Check with your system administrator for more information about operating-system file and directory permissions. UNIX permissions are discussed in [Chapter 2](#) of this manual.

6. To give someone else the permissions to access the command files in your directory, use the UNIX **chmod** command.
7. To give someone else access to the database that you have created, grant them the appropriate privileges using the GRANT statement.

To revoke privileges, use the REVOKE statement. The GRANT and REVOKE statements are described in Chapter 1 of the [Informix Guide to SQL: Syntax](#).

New Features of This Product

The Introduction to each Version 7.2 product manual contains a list of new features for that product. The Introduction to each manual in the Version 7.2 *Informix Guide to SQL* series contains a list of new SQL features.

A comprehensive list of all of the new features for Version 7.2 Informix products is in the Release Notes file called **SERVERS_7.2**.

This section highlights the major new features implemented in Version 7.2 of INFORMIX-SE:

- Global Language Support (GLS)
The GLS feature lets Informix Version 7.2 products handle different languages, cultural conventions, and code sets. GLS functionality supersedes the functionality of Native Language Support (NLS) and Asian Language Support (ALS). GLS eliminates the need to distinguish between internationalized versions of Informix software. ♦
- Multibyte filenames
SE can generate multibyte filenames for the following types of files:
 - Database names for SE (**.dbs**)
 - Table names for SE (**.dat**, **.idx**)
- Multibyte-character support for SE utilities
The **secheck** and **selog** utilities provide multibyte-character support for table names and log filenames.

Conventions

This section describes the conventions that are used in this manual. By becoming familiar with these conventions, you will find it easier to gather information from this and other volumes in the documentation set.

The following conventions are covered:

- Typographical conventions
- Icon conventions

- Command-line conventions
- Sample-code conventions

Typographical Conventions

This manual uses a standard set of conventions to introduce new terms, illustrate screen displays, describe command syntax, and so forth. The following typographical conventions are used throughout this manual.

Convention	Meaning
<i>italics</i>	Within text, new terms and emphasized words are printed in italics. Within syntax diagrams, values that you are to specify are printed in italics.
boldface	Identifiers (names of classes, objects, constants, events, functions, program variables, forms, labels, and reports), environment variables, database names, table names, column names, menu items, command names, and other similar terms are printed in boldface.
monospace	Information that the product displays and information that you enter are printed in a monospace typeface.
KEYWORD	All keywords appear in uppercase letters.
◆	This symbol indicates the end of product- or platform-specific information.






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 described in the following table. This information is always displayed in italics.

Icon	Description
	Identifies paragraphs that contain vital instructions, cautions, or critical information.
	Identifies paragraphs that contain significant information about the feature or operation that is being described.
	Identifies paragraphs that offer additional details or shortcuts for the functionality that is being described.

Compliance Icons

Compliance icons indicate paragraphs that provide guidelines for complying with a standard.

Icon	Description
	Identifies information that is specific to a GLS-compliant database or application.

These icons can apply to a row in a table, one or more paragraphs, or an entire section. A ♦ symbol indicates the end of the compliance information.

Command-Line Conventions

SE supports a variety of command-line options. You enter these commands at the operating-system prompt to perform certain functions in SE.

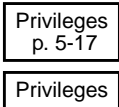
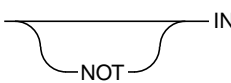
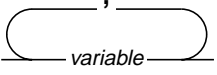
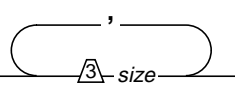
This section defines and illustrates the format of the commands that are available in SE and other Informix products. These commands have their own conventions, which might include alternative forms of a command, required and optional parts of the command, and so forth.

Each diagram displays the sequences of required and optional elements that are valid in a command. A diagram begins at the upper left with a command. It ends at the upper right with a vertical line. Between these points, you can trace any path that does not stop or back up. Each path describes a valid form of the command. You must supply a value for words that are in italics.

You might encounter one or more of the following elements on a command-line path.

Element	Description
command	This required element is usually the product name or other short word that invokes the product or calls the compiler or preprocessor script for a compiled Informix product. It might appear alone or precede one or more options. You must spell a command exactly as shown and must use lowercase letters.
<i>variable</i>	A word in italics represents a value that you must supply, such as a database, file, or program name. A table following the diagram explains the value.
-flag	A flag is usually an abbreviation for a function, menu, or option name or for a compiler or preprocessor argument. You must enter a flag exactly as shown, including the preceding hyphen.
.ext	A filename extension, such as .sql or .dbs , might follow a variable that represents a filename. Type this extension exactly as shown, immediately after the name of the file and a period. The extension might be optional in certain products.

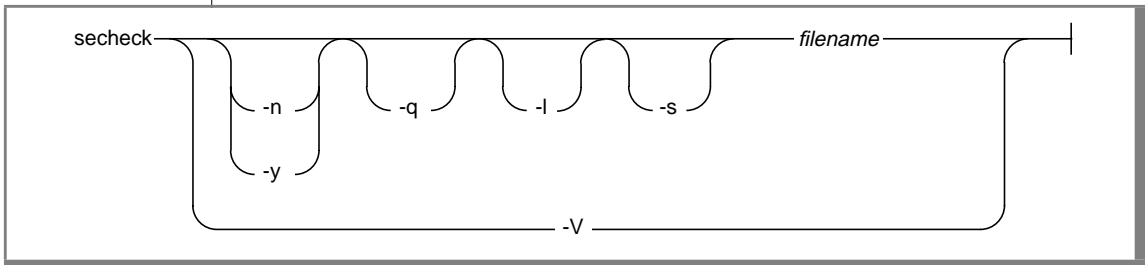
(1 of 2)

Element	Description
(,;+*-/)	Punctuation and mathematical notations are literal symbols that you must enter exactly as shown.
' '	Single quotes are literal symbols that you must enter as shown.
	A reference in a box represents a subdiagram on the same page (if no page is supplied) or another page. Imagine that the subdiagram is spliced into the main diagram at this point.
— ALL —	A shaded option is the default. If you do not explicitly type the option, the default will be in effect unless you choose another option.
→ → →	Syntax enclosed in a pair of arrows indicates that this is a subdiagram.
—	The vertical line is a terminator and indicates that the statement is complete.
	A branch below the main line indicates an optional path. (Any term on the main path is required, unless a branch can circumvent it.)
	A loop indicates a path that you can repeat. Punctuation along the top of the loop indicates the separator symbol for list items, as in this example.
	A gate ($\sqrt{3}$) on a path indicates that you can only use that path the indicated number of times, even if it is part of a larger loop. Here you can specify <i>size</i> no more than three times within this statement segment.

(2 of 2)

Figure 1 shows the flow of the **secheck** utility command.

Figure 1
An Example Command-Line Diagram



To construct a correct command, start at the top left with the command **secheck**. Then follow the diagram to the right, including the options that you want. The elements in the diagram are case sensitive.

To read the example command-line diagram

1. Type the word **secheck**.
2. Choose either the lower path or the upper path.
3. When you choose the lower path, you must type **-v**. That brings you to the terminator and completes the **secheck** command. Press RETURN to execute the command.
4. When you choose the upper path, take the following steps:
 - a. You can choose **-n** or **-y**, but not both.
 - b. You can choose **-q**.
 - c. You can choose **-l**.
 - d. You can choose **-s**.
 - e. Supply a filename.
After you choose *filename*, you come to the terminator. Your command is complete.
5. Press ENTER to execute the command.

Sample-Code Conventions

Examples of SQL code occur throughout this manual. Except where noted, the code is not specific to any single Informix application development tool. If only SQL statements are listed in the example, they are not delimited by semicolons. To use this SQL code for a specific product, you must apply the syntax rules for that product. For example, if you are using the Query-language option of DB-Access, you must delimit multiple statements with semicolons. If you are using an SQL API, you must use EXEC SQL and a semicolon (or other appropriate delimiters) at the start and end of each statement, respectively.

For instance, you might see the code in the following example:

```
CONNECT TO stores7
.
.
.
DELETE FROM customer
      WHERE customer_num = 121
.
.
.
COMMIT WORK
DISCONNECT CURRENT
```

Dots in the example indicate that more code would be added in a full application, but it is not necessary to show it to describe the concept being discussed.

For detailed directions on using SQL statements for a particular application development tool or SQL API, see the manual for your product.

Additional Documentation

The SE documentation set includes printed manuals, on-line manuals, and on-line help.

This section describes the following pieces of the documentation set:

- Printed documentation
- On-line documentation
- Related reading

Printed Documentation

The following printed manuals are included in the SE documentation set:

- The *UNIX Products Installation Guide* contains instructions for installing Informix products on computers that run the UNIX operating system. Keep this guide with your UNIX software documentation for easy reference.
- The *Guide to GLS Functionality* contains information on the language-related topics of Global Language Support (GLS). ♦
- The *Informix Migration Guide* describes the procedures to use when you migrate existing Informix databases to and from SE, Version 7.2. This manual includes information on preparing your host system to support the new features provided by SE 7.2.
- If you have never used Structured Query Language (SQL) or an Informix application development tool, read the *Informix Guide to SQL: Tutorial*. The manual provides a tutorial on SQL as it is implemented by Informix products. It describes the fundamental ideas and terminology that are used when planning, using, and implementing a relational database.
- A companion volume to the Tutorial, the *Informix Guide to SQL: Reference*, provides reference information on the types of Informix databases that you can create, the data types that are supported in Informix products, system catalog tables that are associated with the database, environment variables, and the SQL utilities. The manual also provides a detailed description of the **stores7** demonstration database and contains a glossary.
- An additional companion volume to the Tutorial, the *Informix Guide to SQL: Syntax*, provides a detailed description of all the SQL statements supported by Informix products. The manual also provides a detailed description of Stored Procedure Language (SPL) statements.
- The *DB-Access User Manual* describes how to invoke the DB-Access utility to access, modify, and retrieve information from SE relational databases.
- When errors occur, you can look them up by number and learn their causes and solutions in the *Informix Error Messages* manual. If you prefer, you can look up the error messages in the on-line message file that is described later in this introduction and in the Introduction to the *Informix Error Messages* manual.

On-Line Documentation

Several different types of on-line documentation are available:

- On-line documentation and help
- On-line error messages
- Release notes, documentation notes, and machine notes

Error Message Files

Informix software products provide ASCII files that contain all of the Informix error messages and their corrective actions. To read the error messages in the ASCII file, Informix provides scripts that let you display error messages on the screen (**finderr**) or print formatted error messages (**rofferr**). For a detailed description of these scripts, see the Introduction to the [Informix Error Messages](#) manual.

The optional Informix Messages and Corrections product provides PostScript files that contain the error messages and their corrective actions. If you have installed this product, you can print the PostScript files on a PostScript printer. The PostScript error messages are distributed in a number of files of the format **errmsg1.ps**, **errmsg2.ps**, and so on. These files are located in the **\$INFORMIXDIR/msg** directory.

Release Notes, Documentation Notes, Machine Notes

In addition to the Informix set of manuals, the following on-line files, located in the `$INFORMIXDIR/release/en_us/0333` directory, might supplement the information in this manual:

On-line File	Purpose
Documentation notes	Describes features that are not covered in the manuals or that have been modified since publication. The file that contains the documentation notes for this product is called <code>SEDOC_7.2</code> .
Release notes	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. The file that contains the release notes for Version 7.2 of Informix database server products is called <code>SERVERS_7.2</code> . The release notes also contain a section on limits that describes the maximum values for SE.
Machine notes	Describes any special actions that are required to configure and use Informix products on your computer. Machine notes are named for the product that is described. The machine notes file for SE is <code>SE_7.2</code> .

Please examine these files because they contain vital information about application and performance issues.

Related Reading

For additional technical information on database management, consult the following books. The first book is an introductory text for readers who are new to database management, while the second book is a more complex technical work for SQL programmers and database administrators:

- *An Introduction to Database Systems* by C. J. Date (Addison-Wesley Publishing, 1994)
- *Database: A Primer* by C. J. Date (Addison-Wesley Publishing, 1983)

To learn more about the SQL language, consider the following books:

- *A Guide to the SQL Standard* by C. J. Date with H. Darwen (Addison-Wesley Publishing, 1993)
- *Understanding the New SQL: A Complete Guide* by J. Melton and A. Simon (Morgan Kaufmann Publishers, 1993)
- *Using SQL* by J. Groff and P. Weinberg (Osborne McGraw-Hill, 1990)

The *INFORMIX-SE Administrator's Guide* 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:

- *Introducing the UNIX System* by H. McGilton and R. Morgan (McGraw-Hill Book Company, 1983)
- *Learning the UNIX Operating System* by G. Todino, J. Strang, and J. Peek (O'Reilly & Associates, 1993)
- *A Practical Guide to the UNIX System* by M. Sobell (Benjamin/Cummings Publishing, 1989)
- *UNIX for People* by P. Birns, P. Brown, and J. Muster (Prentice-Hall, 1985)
- *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 on INFORMIX-OnLine Dynamic Server. In addition, many features of OnLine comply with the SQL-92 Intermediate and Full Level and X/Open CAE (common applications environment) standards.

Informix SQL-based products are compliant with ANSI SQL-92 Entry Level (published as ANSI X3.135-1992) on INFORMIX-SE with the following exceptions:

- Effective checking of constraints
- Serializable transactions

Informix Welcomes Your Comments

Please let us know what you like or dislike about our manuals. To help us with future versions of our manuals, please tell us about any corrections or clarifications that you would find useful. 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`

Or, send a facsimile to the Informix Technical Publications Department at:

415-926-6571

Please 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

We appreciate your feedback.

Installation and Initial Configuration

Defining a Database Management System	1-3
Planning for SE	1-4
Configuring Hardware	1-4
Placing Active Tables and Files on the Disk	1-4
Considering SE Limits	1-5
Installing SE	1-6
Preparing SE Connections	1-6
Understanding Version 7.2 SE Configurations	1-7
Setting Environment Variables	1-8
Understanding the Communication Files	1-9
Network Communication Files	1-10
Network Security Files	1-10
The sqlhosts File	1-12
Building the sqlhosts File	1-12
The dbservername Field	1-13
The nettype Field	1-13
The hostname Field	1-15
The servicename Field	1-16
Relationships Among Network-Connection Files for TCP/IP	1-17
Examples of Client/Server Connections	1-18
Local Connections with Pipes	1-19
Network Connections with Version 7.2 Products	1-20
Local-Loopback Connections with Version 7.2 Products	1-22
Connecting with Different Versions	1-26
Local Connections with Version 4.11 or 5.x Client Applications.	1-26
Network Connections with INFORMIX-NET	1-28

Network Connections with 5.x INFORMIX-NET Relay Module	1-29
Network Connections with the Version 7.2 Relay Module	1-32
Starting the sqlxecd Daemon	1-35
Using an NFS-Mounted Directory	1-37
How Does a Client Application Connect to a Database Server?.	1-38

Implementing a database management system (DBMS) requires making many decisions, such as where to store the data, how to access the data, and how to protect the data. How you implement the DBMS can greatly affect the performance of database operations. For example, the physical organization of data and optimization performed by the DBMS directly affect the speed of retrievals from and updates to tables.

This chapter explains the issues that are involved in setting up INFORMIX-SE. You can use this chapter as background to help you understand the effects of the choices that you make as an SE administrator.

This chapter also discusses SE connectivity and communication as well as the contents of the `sqlhosts` file.

Defining a Database Management System

You can divide a DBMS into the following parts:

- A *data language*, which serves as the user interface to the DBMS
- A *database server*, which takes the *data definition* and *data manipulation* language requests and performs the requested operations

Database users instruct the DBMS to perform queries and other operations on a database using language that the DBMS understands. Informix application development tools and SQL APIs use a database definition and manipulation language that is an extension of the ANSI standard SQL to send instructions to the database server.

Planning for SE

Planning can help you avoid costly and time-consuming mistakes. For example, before you install the software, consider where to locate it so that everybody who needs to access it can do so.

After you decide where to locate the product, think about the placement of the database and its associated tables. Do you want to place your tables on different devices? Do you want to put your transaction log or audit trails on separate disks? Also consider the computer itself. Can it support the memory requirements of SE? Can it support the number of open files, locks, and users that the application requires? In addition, you must think about maintenance plans. Prepare a backup schedule. Decide where to keep the tapes.

Configuring Hardware

If you partitioned your disks before you began planning your SE system, examine the physical layout of your disk and verify that the partition in which you want to load your Informix products is large enough to contain the product software. (Informix does not require you to place the product software in the same directory as the data.)

Placing Active Tables and Files on the Disk

A database server that uses the UNIX file system cannot transfer data directly from memory to disk. The intermediate transfer to and from the operating-system buffer increases the time for the transfer. Although intermediate transfer does not always create a performance bottleneck on a database with moderate activity, your most important design goal must be to minimize the time required for disk input/output (I/O).

Keep your most active tables on separate disk devices to reduce contention. Using separate disk devices reduces competition for disk access when joins form among high-demand tables.

Because transaction logs and audit trails have high rates of activity, you must give them priority in disk placement. Ideally, designate the fastest devices or the most central areas of the disk to hold the transaction logs and audit trails.

Remember the following points as you plan the layout for your SE data:

- How you position data on disk devices can minimize head movement.
- The innermost disk partitions generally have the fastest access times—use them for frequently accessed tables or logs.
- SE tables cannot span partitions. Think carefully about where you decide to put tables. When a table fills a partition, you must do one of the following procedures:
 - Move the table to a larger partition.
 - Back up the data, increase the size of the partition, and restore the data.
- SE indexes (.**idx** files) automatically reside on the same partition as your **.dat** files.
- Transaction logs and audit trails grow as you process transactions against the database.

Considering SE Limits

Consider the following limits of SE when you install the product:

- The maximum row size (32,511 bytes)
- The maximum number of open SE tables (255)
- The maximum number of locks per table (operating-system-dependent)

Because SE takes advantage of some of the operating-system facilities that UNIX provides, it also incurs some of the limits that UNIX imposes. The number of open files that your operating system allows represents one such limit. You can increase the value of the following UNIX kernel parameters to improve SE performance (although these modifications do not alter the SE limitations just listed):

- The number of locks
- The number of open files
- The number of inodes (an operating system entity used to uniquely identify a file)

Installing SE

Installation refers to the process of loading the product files onto your UNIX system and running the installation script to correctly set up the product files. Some of the specific steps that you should follow as part of your installation depend on your environment. See the [UNIX Products Installation Guide, Version 7.2](#), for instructions about installing the Version 7.2 SE database server.

Preparing SE Connections

Before you can use SE with a client application, you must first understand the following topics:

- Configuring INFORMIX-SE, Version 7.2, for local and remote connections
- Setting your environment variables
- Establishing communication
- Building the **sqlhosts** file
- Connecting Informix Version 7.2 client applications to Version 7.2 SE database servers
- Connecting Informix Version 4.1 or Version 5.x client applications to Version 7.2 SE database servers
- Starting the **sqlxecd** daemon
- Using NFS-mounted directories

The [UNIX Products Installation Guide](#) suggests that you create the directory **/usr/informix** for your Informix products. For the examples in the following sections, it is assumed that you have installed the Informix Version 4.1 or 5.x products in the **/usr/informix** directory, and that you have installed the Informix Version 7.2 products in the **/usr/version7/informix** directory. However, when you establish local connections using unnamed pipes, you must install client and server products in the same directory.

Understanding Version 7.2 SE Configurations

You must understand local and network configurations before you establish communication.

A *local* client/server configuration exists when a client product connects to a database server on the same computer. A *remote* client/server configuration exists when a client product that resides on one computer establishes a connection across a network to a database server that resides on another computer.

Figure 1-1 illustrates the compatibility between client tools and a local Version 7.2 SE database server.

Figure 1-1
Client/Server Configurations for Client Products Connecting to a Local Version 7.2 SE Database Server

Client	Local SE Database Server, Version 7.2
Version 4.1 ESQL/C, ESQL/COBOL	Supported, but must use the Version 7.2 Relay Module and Version 4.1 syntax
Version 5.x ESQL/C, ESQL/COBOL, ESQL/FORTRAN	Supported, but must use the Version 7.2 Relay Module and Version 5.x syntax
Version 6.x ESQL/C, ESQL/COBOL	Supported, but cannot use syntax specific to Version 7.2 products
Version 7.2 ESQL/C, ESQL/COBOL	Supported

Figure 1-2 illustrates the compatibility between client products and a remote Version 7.2 SE database server.

Figure 1-2
Client/Server Configurations for Client Products Connecting to a Remote Version 7.2 SE Database Server

Client	Remote SE Database Server, Version 7.2
Version 4.11 INFORMIX-SQL, INFORMIX-4GL	Supported, but must establish connection using Version 4.1 or 5.x INFORMIX-NET or the Version 7.2 Relay Module. The Version 7.2 Relay Module is an integral part of every Informix Version 7.2 database server. Must also use Informix Version 4.1 syntax.
Version 5.x ESQL/C, ESQL/COBOL, ESQL/FORTRAN	Supported, but must establish connection using Version 5.x INFORMIX-NET or the Version 7.2 Relay Module. The Version 7.2 Relay Module is an integral part of every Informix Version 7.2 database server. Must also use Informix Version 5.x syntax.
Version 6.x ESQL/C, ESQL/COBOL	Supported, but cannot use syntax specific to Version 7.2 products
Version 7.2 ESQL/C, ESQL/COBOL	Supported

Setting Environment Variables

You must set certain UNIX and Informix environment variables correctly for your Informix products to work. These environment variables are documented in the [Informix Guide to SQL: Reference](#) and the [Guide to GLS Functionality](#). Pay particular attention to the following environment variables:

DBPATH	identifies directories and database servers that contain databases.
INFORMIXDIR	specifies the directory where you install your product files.
INFORMIXSERVER	specifies the name of the default database server.
INFORMIXSQLHOSTS	specifies the full pathname and filename of a file that contains connectivity information (optional).

INFORMIXTERM	specifies whether to use the termcap file or terminfo directory (optional).
PATH	finds executable files.
SQLEXEC	is required for some network configurations.
SQLRM	is required for some network configurations.
SQLRMDIR	is required for some network configurations.
TERM	enables your client product to recognize and communicate with the terminal you are using. Sometimes you must also set TERMCAP and TERMINFO . See your UNIX system administrator for details.

GLS

The Global Language Support (GLS) feature of SE lets you use non-English characters, monetary conventions, and collating sequences. If you are using the GLS features of SE, you need to set the GLS-related environment variables, which are documented in Chapter 5 of the [Guide to GLS Functionality](#). ♦

Understanding the Communication Files

The *communication files* contain information necessary for client applications and database servers to communicate with one another. The files that are associated with network communication fall into the following groups:

- Network communication files: **/etc/hosts** and **/etc/services** (used only for the TCP/IP network protocol)
- Network security files
- **\$INFORMIXDIR/etc/sqlhosts**

Of these files, you maintain only the **\$INFORMIXDIR/etc/sqlhosts** file. The UNIX system (or network) administrator, or the end user (in the case of some network security files), manages the other files.

Important: For the remainder of this chapter, **\$INFORMIXDIR/etc/sqlhosts** is referred to as the **sqlhosts** file.



Network Communication Files

The network administrator maintains the network communication files. You use information from **/etc/hosts** and **/etc/services** for the **sqlhosts** file when you prepare the TCP/IP network connections. For IPX/SPX network connections, you need the name of the Novell NetWare configuration file. Also, you need to work closely with the network administrator to make sure you are using accurate information.

For information about these files, refer to the systems manuals for your installation and to the UNIX man pages for **hosts** and **services**.

Network Security Files

A client application cannot connect to a remote database server unless the user has the proper access permission. Therefore, you must know about the implications of the network security files (**/etc/passwd**, **~/.netrc**, **/etc/shadow**, **/etc/host.equiv**, **~/.rhosts**) on interhost communication (communication between host computers).

Informix products follow standard UNIX security procedures, governed by information contained in the network security files. For information about these procedures, refer to the systems manuals for your installation and to the UNIX man pages **hosts.equiv** and **netrc**.

The following methods allow you to gain access to the network:

- The trusted host method, described in the UNIX man pages for `hosts.equiv` and `rhosts`, establishes connections between trusted host computers over a network. This method does not require you to supply a password.

When you set up the `hosts.equiv` and `.rhosts` files on the database server host, you usually want to make a client host a *trusted host*. You need to specify the client host name in the `/etc/hosts.equiv` or `.rhosts` file on the database server host. However, on some networks, the host name that the network uses to refer to that computer might not be exactly the same as the host name that the computer uses to refer to itself. For example, the network host name might contain the full domain name, as shown in the following example:

```
viking.informix.com
```

But the computer might refer to itself using the local host name shown in the following example:

```
viking
```

If this occurs, make sure that you specify both host names in your `/etc/host.equiv` and `.rhosts` files.

- INFORMIX-NET PC, INFORMIX-NET *for Windows*, INFORMIX-NET *for Macintosh*, and INFORMIX-NET *for OS/2* prompt you to enter a password when you attempt to establish a network connection.
- The SQL CONNECT statement in an Informix SQL API, such as INFORMIX-ESQL/C or INFORMIX-ESQL/COBOL, allows you to specify a password for establishing a network connection. However, you cannot use an SQL CONNECT statement to specify a password in DB-Access.
- The `.netrc` file holds password information used to establish connections to remote hosts. Refer to the UNIX man pages for information on how to use `netrc`.

The sqlhosts File

The `$INFORMIXDIR/etc/sqlhosts` file is the *connectivity file*. It contains information that enables an Informix client application to connect to any Informix database server on the network. It specifies the database server name, the type of connection, the name of the host computer, and the service name.

You must prepare the `sqlhosts` file even if the client application and the SE database server reside on the same computer. The `sqlhosts` file is described in “Building the sqlhosts File” below.

The `sqlhosts` file has one entry (one line) for each type of potential connection from a client application to a database server. When you use a remote connection, an `sqlhosts` file must reside on the client computer and the database server computer.

The client application expects to find the `sqlhosts` file in the `$INFORMIXDIR/etc` directory; however, you can change this location or the name of the file with the `INFORMIXSQLHOSTS` environment variable. For more information on this environment variable, refer to Chapter 4 of the [Informix Guide to SQL: Reference](#).

Building the sqlhosts File

Figure 1-3 shows an example of `sqlhosts` fields. The next four sections provide detailed information about the following fields:

- Database servername field
- Network protocol field
- Hostname field
- Servicename field

You can edit the `sqlhosts` file with any convenient text editor. The entries in the first three fields can include any printable character but not an uppercase character, a field delimiter, a new-line character, or a comment character. The entries in the fourth field can also include uppercase characters. You delimit the fields shown in Figure 1-3 with spaces or tabs.

Figure 1-3
Example of Fields in an sqlhosts File

dbservername	nettype	hostname	servicename
valley_se	setlitcp	valley	valley_service
river_se	seipcpi	river	sqlxec

The dbservername Field

The **dbservername** field contains the database server name that the **INFORMIXSERVER** environment variable specifies. Each database server on the network must possess a unique name that cannot exceed 18 characters.

The nettype Field

The **nettype** field, a string of eight letters composed of three subgroups, describes the type of connection that a client can use to connect to the database server. Figure 1-4 illustrates the **nettype** field.

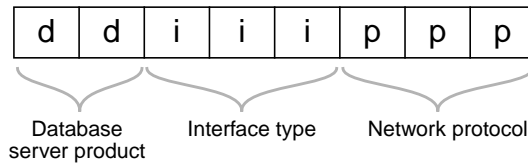


Figure 1-4
Format of the nettype Field

The following list describes the subfields of **nettype** and the valid values allowed for SE, as shown in Figure 1-5:

- The first two letters represent the database server product:
 - The *se* represents INFORMIX-SE.
 - The *on* represents the INFORMIX-OnLine Dynamic Server (not used in this situation).
 - The *gw* represents INFORMIX-Gateway *with DRDA* (not used in this situation).
- The middle three letters represent the internal programming interface connection type that enables communication:
 - The *ipc* represents a UNIX-based *interprocess communication* (IPC) connection that is used only for local loopback connections.
 - The *tli* represents a *transport-level interface* (TLI) network interface that is used for local-loopback and remote-host connections.
 - The *soc* represents a socket type of *network interface* that is used for local-loopback and remote-host connections.

The installation package for this product contains the Machine Notes file, **SE_7.1**, that discusses the network interface that your platform supports.

- The final three letters represent the specific IPC mechanism or the network protocol:
 - The *pip* represents unnamed pipes. Use *pip* only for local connections.
 - The *tcp* represents the TCP/IP protocol. Use *tcp* for network connections.
 - The *spx* represents the IPX/SPX protocol. Use *spx* for network connections.

The network interface describes the behavior between the computer and the network. The network protocol describes the behavior of the network itself.

Figure 1-5 shows the valid values for **nettype** used with SE.

Figure 1-5
Valid nettype Values for SE

nettype	Description
seipcpi	SE using unnamed pipes for local mode
setlitcp	SE using TLI with TCP/IP protocol
sesoectp	SE using sockets with TCP/IP protocol
setlispx	SE using TLI with IPX/SPX protocol

The hostname Field

The **hostname** field specifies the computer where the database server product resides. When you use the TCP/IP connection protocol, the **hostname** field must correspond to the host-name entry in the **/etc/hosts** file, which provides the network address of the host computer. As a rule, the **hostname** entry in the **/etc/hosts** file is the same as the name of your computer. See your network administrator to determine what name is assigned to a specific host, and make sure that you both agree on the **hostname** field.

Figure 1-6 shows an example of the **/etc/hosts** fields. The **host-alias** field is optional.

Figure 1-6
Example of hostname Fields in an /etc/hosts File

net address	hostname	host-alias
29.9.925.6	valley	accounts

Informix bases its implementation of IPX/SPX on the NetWare implementation of IPX/SPX. When you use the IPX/SPX protocol, the **hostname** field must contain the name of the NetWare file server rather than the actual name of the host computer. The implementation of IPX/SPX varies among vendors, so refer to the vendor documentation for more details about the required files.

The servicename Field

The interpretation of the **servicename** field depends on the type of network connection that is specified in the **nettype** field.

When you use unnamed pipes, the **servicename** field must contain the name of the SE executable file, distributed as **sqlexec**.

When you specify a **nettype** field whose last three letters are *tcp*, the connection is a TCP/IP network connection. When you use the TCP/IP connection protocol, the service name must correspond to a service-name entry in the **/etc/services** file, as illustrated in Figure 1-7. The **/etc/services** file tells the network software how to find the database server on the specified host computer. It does not matter what service name you choose as long as you and your network administrator agree on a name.

Figure 1-7

Example of servicename Field in an /etc/services File

servicename	port #/protocol	service-alias
valley_service	1536/tcp	



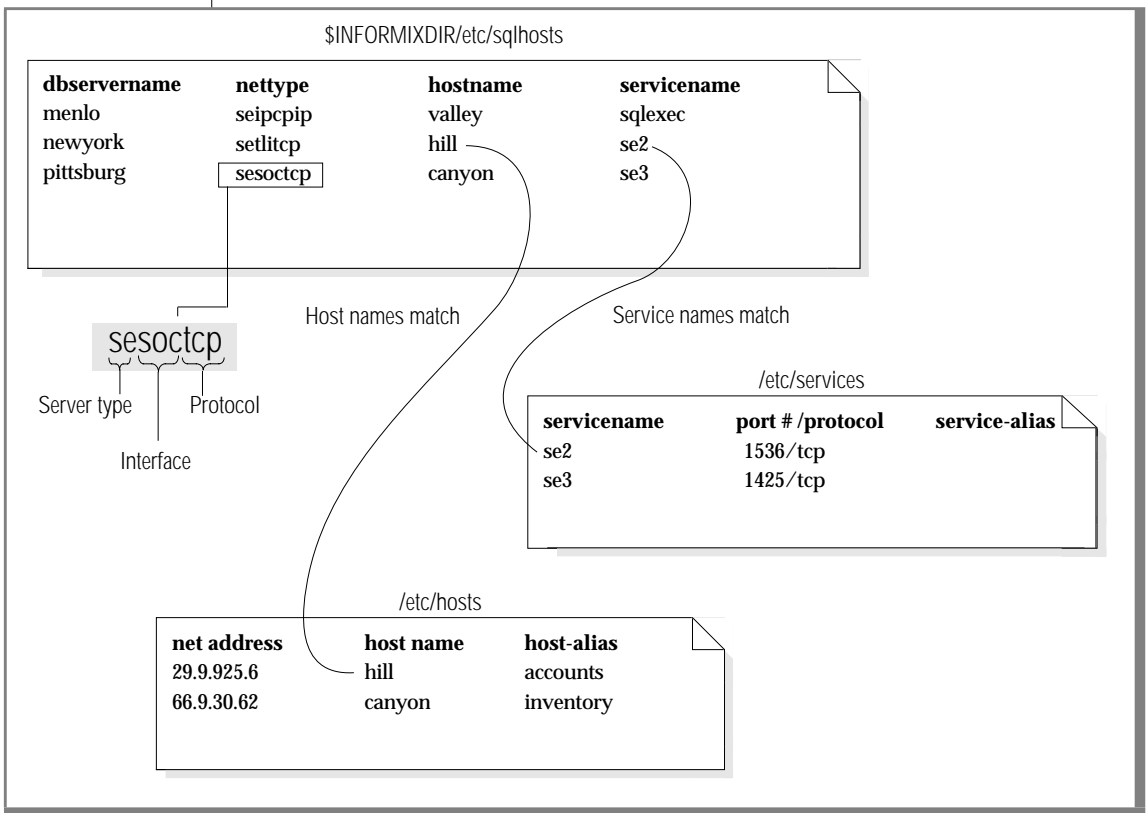
Tip: When you are using NIS, type `yycat services` and press RETURN to verify the services available in your domain.

When you specify a **nettype** field whose last three letters are *spx*, you support an IPX/SPX network connection. When you use the IPX/SPX connection protocol, the service name must match **dbservername**.

Relationships Among Network-Connection Files for TCP/IP

Common information stored in specific fields relates the **sqlhosts**, **/etc/hosts**, and **/etc/services** files to each other for TCP/IP connections. Figure 1-8 illustrates the relationships among the **\$INFORMIXDIR/etc/sqlhosts** file, the **/etc/services** file, and the **/etc/hosts** file. The **servicename** fields in the **sqlhosts** and **/etc/services** files match. The **hostname** fields in the **sqlhosts** and **/etc/hosts** files also match.

Figure 1-8
Relationships Among Network-Connection Files



Examples of Client/Server Connections

This section discusses client/server configurations for making connections between an Informix Version 7.2 client and a Version 7.2 SE database server. The following configurations are possible:

- A local connection with unnamed pipes
- A remote (network) connection
- A local loopback connection

The connection information that this section describes is true for all Version 6.0 and later clients and database servers.

For information about connecting Version 7.2 SE database servers with an Informix Version 5.x or 4.1 client, see [“Connecting with Different Versions” on page 1-26](#).

In all the configurations in the next sections, you must correctly set the following environment variables for the user:

- **PATH**
- **TERM**
- **DBPATH** (optional)

PATH must always include **\$INFORMIXDIR/bin** so that the computer can find the Informix products. **TERM** provides information to the UNIX operating system so that the key set and display function properly. **DBPATH** shows the location of databases and database servers.

In the following examples, it is assumed that you have installed the Informix Version 7.2 products in the **/usr/version7/informix** directory. In other words, when you perform the installation, set the **INFORMIXDIR** environment variable to **/usr/version7/informix**.

Local Connections with Pipes

When the client application and the SE database server reside on the same host computer, the client application accesses the local database server using unnamed pipes, as shown in Figure 1-9.

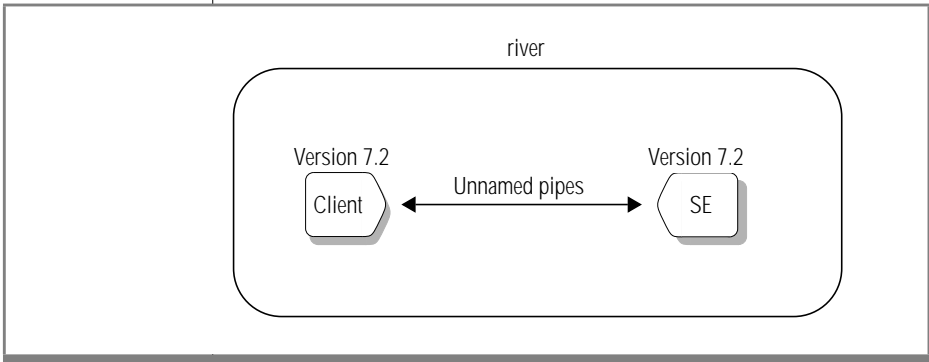


Figure 1-9
A Local Configuration Using Unnamed Pipes



Important: The preceding diagram is also accurate for a connection between a Version 7.x client and a Version 6.x SE database server or a Version 6.x client and a Version 7.x SE database server.

The user must set the **INFORMIXDIR** and **INFORMIXSERVER** environment variables to the values that are shown in Figure 1-10.

Figure 1-10
Environment Variables and Required Settings

Environment Variable	Value
INFORMIXDIR	/usr/version7/informix
INFORMIXSERVER	local_se



Tip: Remember, **/usr/version7/informix** represents the installation directory for all Informix Version 7.2 products.

The **\$INFORMIXDIR/etc/sqlhosts** file must include the following entry:

```
local_se seipcpip river sqlexec
```

The fourth item in the **sqlhosts** entry (**sqlxec**) represents the name of the executable file for SE, as installed by the Informix installation procedure. When you do not modify the installation, **sqlxec** is the executable file.

The SE database server process starts when the client executes one of the following statements:

- `CONNECT TO database_name@local_se`
- `CONNECT TO @local_se`

Because these statements represent a purely local connection, the **/etc/hosts** and **/etc/services** files are unaffected.

Network Connections with Version 7.2 Products

When the client application resides on one computer and the database server resides on another computer, the connection is called a *remote or network* configuration, as shown in Figure 1-11.

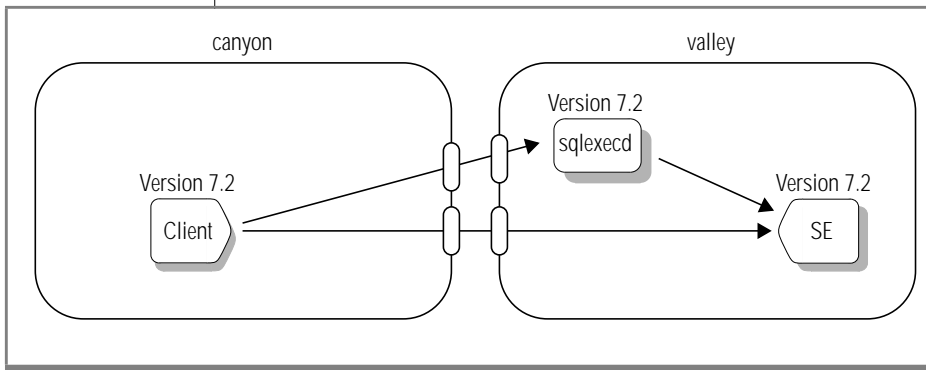


Figure 1-11
Network Configuration with an Informix Version 7.2 Client Application and an INFORMIX-SE, Version 7.2, Database Server



Important: The preceding diagram is also accurate for a connection between a Version 7.x client and a Version 6.x SE database server or a Version 6.x client and a Version 7.x SE database server.

Setting Up the canyon and valley Host Computers

For the **canyon** computer, set the **INFORMIXDIR** and **INFORMIXSERVER** environment variables to the values shown in Figure 1-12.

Figure 1-12
*Environment Variables and
Required Settings*

Environment Variable	Value
INFORMIXDIR	/usr/version7/informix
INFORMIXSERVER	valley_se

The **sqlxecd** daemon must be running. To start the daemon, you must log in as **root** and enter the following command:

```
/usr/version7/informix/lib/sqlxecd valley_se
```

For information about the **sqlxecd** daemon, refer to “[Starting the sqlxecd Daemon](#)” on page 1-35.

Setting Up the Communication Files

Each host computer (**hilltop** and **valley**) must contain an **sqlhosts** file that includes connection information associated with the network interface and network protocol that you are using. Figure 1-13 shows the **sqlhosts** file entries for different network interfaces.

Figure 1-13
*Network Interfaces and Associated
sqlhosts File Entries*

Network Interface	sqlhosts File Entry
Sockets on TCP/IP	valley_se sesoctcp valley valley_service
TLI interface on TCP/IP	valley_se setlitcp valley valley_service

When you use a TCP/IP network, the **/etc/hosts** file must contain an entry for the **valley** computer and the **/etc/services** file must contain an entry for **valley_service**.

When the host computer uses the TLI network interface on an IPX/SPX network, the **hostname** field contains the name of the NetWare file server instead of the computer host name. On the display screens that are associated with the preparation of the NetWare connections, the screen displays the NetWare file server name in uppercase letters; for example, **NW_SVR**. However, in the **sqlhosts** file, you must enter the name in lowercase letters; for example, **nw_svr**.

For an IPX/SPX connection, the value in the **servicename** field can contain an arbitrary string, but that string must represent a unique name among the names of services available on the IPX/SPX network. In other words, you cannot use the string as a service name on any other file server on the network when that file server does not use an Informix product. You can use the **dbservername** value in the **servicename** field, as shown in Figure 1-14, providing that the **dbservername** value exists as a unique service name.

Figure 1-14
Using the dbservername Value in the servicename Field

Affected Field	sqlhosts File Entry
servicename	valley_se setlisp nw_svr valley_se

Local-Loopback Connections with Version 7.2 Products

A network connection between a client application and a database server on the same computer is called a *local-loopback* connection. A local-loopback connection uses network connections even though the client and the database server reside on the same computer. You can make a local-loopback connection if your computer is equipped to process network transactions.

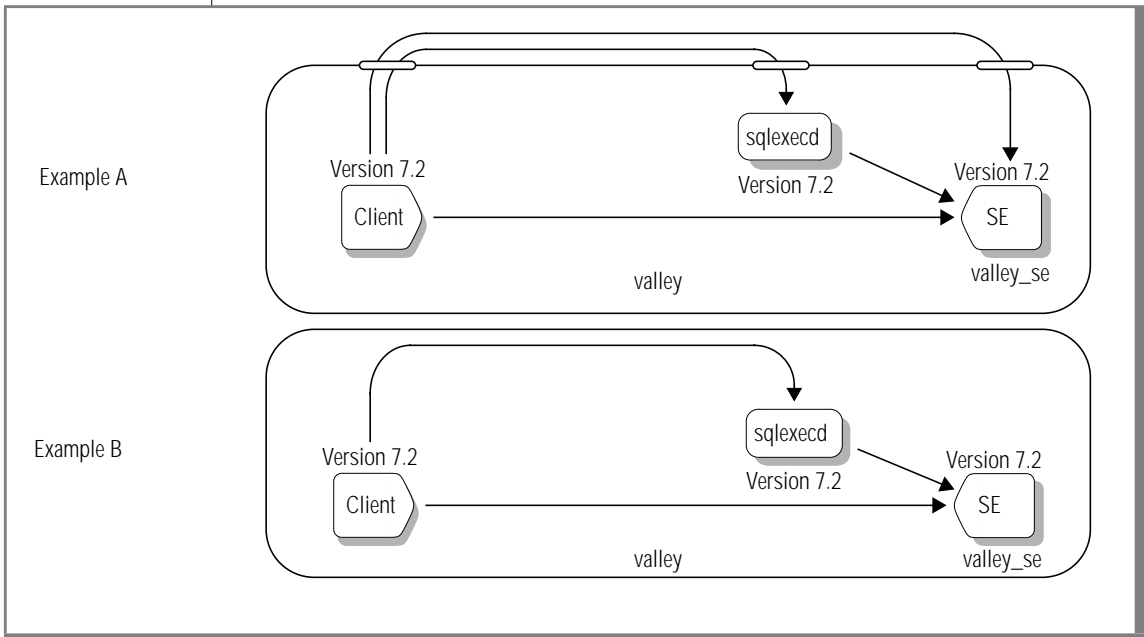
Figure 1-15 shows a local-loopback connection between an Informix Version 7.2 client application and a Version 7.2 SE database server. You can think of this configuration in the following ways:

- The connections go outside the computer **valley** and come back inside again, as shown in Example A of Figure 1-15.
- The connections remain within the **valley** host computer, as shown in Example B of Figure 1-15.

Tip: You can use a local-loopback connection to test network operations without a remote computer.



Figure 1-15
A Local-Loopback Configuration with a Version 7.2 Client Application and Version 7.2 SE Database Server



Important: This diagram is also accurate for a connection between a Version 7.2 client and a Version 6.x SE database server or a Version 6.x client and a Version 7.2 SE database server.

Setting Up the valley Host Computer

The user must set the **INFORMIXDIR** and **INFORMIXSERVER** environment variables to the values shown in Figure 1-16.

Figure 1-16
*Environment Variables and
Required Settings*

Environment Variable	Value
INFORMIXDIR	/usr/version7/informix
INFORMIXSERVER	valley_se

The **sqlxecd** daemon must be running. To start the daemon, log in as **root** and enter the following command:

```
/usr/version7/informix/lib/sqlxecd valley_se
```

For information about the **sqlxecd** daemon, refer to [“Starting the sqlxecd Daemon”](#) on page 1-35.

Setting Up the Communication Files

Each host computer (**hilltop** and **valley**) must contain an **sqlhosts** file that includes connection information associated with the network interface and network protocol that you are using. Figure 1-17 shows the **sqlhosts** file entries for different network interfaces.

Figure 1-17
*Network Interfaces and Associated
sqlhosts File Entries*

Network Interface	sqlhosts File Entry
Sockets on TCP/IP	valley_se sesoctcp valley valley_service
TLI interface on TCP/IP	valley_se setlitcp valley valley_service

When you use a TCP/IP network, the **/etc/hosts** file must contain an entry for the **valley** computer and the **/etc/services** file must contain an entry for **valley_service**.

When the host computer uses the TLI network interface on an IPX/SPX network, the **hostname** field contains the name of the NetWare file server instead of the computer host name. On the display screens that are associated with the preparation of the NetWare connections, the screen displays the NetWare file server in uppercase letters; for example, **NW_SVR**. However, in the **sqlhosts** file, the name appears in lowercase letters; for example, **nw_svr**.

For an IPX/SPX connection, the value in the **servicename** field can contain an arbitrary string, but that string must represent a unique name among the names of services available on the IPX/SPX network. In other words, you cannot use the string as a service name on any other file server on the network, even if that file server does not use an Informix product. You can use the **dbservername** value in the **servicename** field, as shown in Figure 1-18, providing that the **dbservername** value exists as a unique service name.

Figure 1-18
Using the *dbservername* Value
in the *servicename* Field

Affected Field	sqlhosts File Entry
servicename	valley_se setlisp nw_svr valley_se

Connecting with Different Versions

This section shows how to connect an Informix Version 4.1 or Version 5.x client application to a Version 7.2 SE database server. You can connect the client and server components in the following ways:

- An Informix Version 4.1 or 5.x client application connects directly to a local Version 7.2 SE database server.
- An Informix Version 4.11 client application connects to an Informix Version 7.2 database server using Version 4.11 INFORMIX-NET, with INFORMIX-SE or the Relay Module component of an Informix Version 7.2 database server.
- An Informix Version 5.x client application connects to an Informix Version 7.2 database server using Version 5.x INFORMIX-NET with INFORMIX-SE, a Version 5.x Relay Module, or the Relay Module component of an Informix Version 7.2 database server. For more information about using INFORMIX-NET with INFORMIX-SE or the INFORMIX-NET Relay Module, refer to the *INFORMIX-NET/INFORMIX-STAR Installation and Configuration Guide*. For more information about using the Relay Module component of a Version 7.2 SE database server, refer to “Local Connections with Version 4.11 or 5.x Client Applications” below.

Local Connections with Version 4.11 or 5.x Client Applications

Figure 1-19 shows a configuration with an Informix Version 4.11 or 5.x client application connecting to a Version 7.2 SE database server:

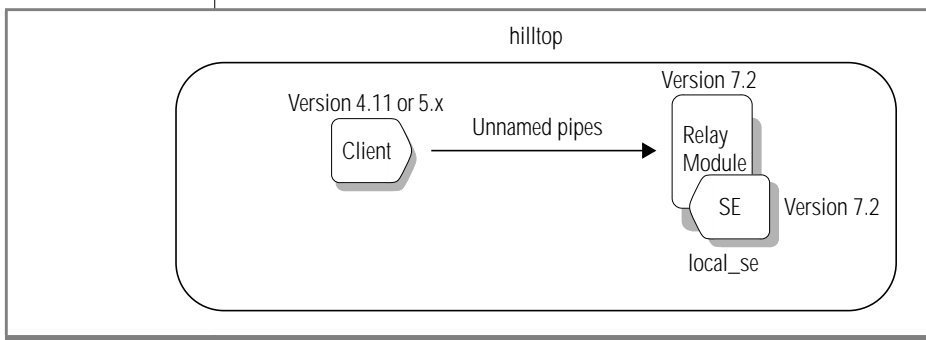


Figure 1-19
Local Connection from a Version 4.11 or 5.x Client Application to a Version 7.2 Database Server

The Version 7.2 Relay Module, a component of Informix Version 7.2 database servers, is specifically designed to allow Informix Version 4.11 and 5.x client applications to connect to an Informix Version 7.2 database server.

For the configuration in Figure 1-19, the user must set the **INFORMIXDIR** and **INFORMIXSERVER** environment variables to the values shown in Figure 1-20.

Figure 1-20
Environment Variables and
Required Settings

Environment Variable	Value
INFORMIXDIR	/usr/informix
INFORMIXSERVER	local_se
SQLEXEC	/usr/informix/lib/sqlrm

You must set the **SQLEXEC** environment variable to the complete pathname of the executable file for the Version 7.2 Relay Module.

Important: When you use unnamed pipes, the Informix Version 4.1 and 5.x client application must reside in the same directory as the Informix Version 7.2 database server.

The **sqlhosts** file must include the following entry:

```
local_se seipcpip hilltop sqlxec
```

The **sqlhosts** file belongs in the **\$INFORMIXDIR/etc** directory. For local connections, the **/etc/hosts** and **/etc/services** files remain unaffected.

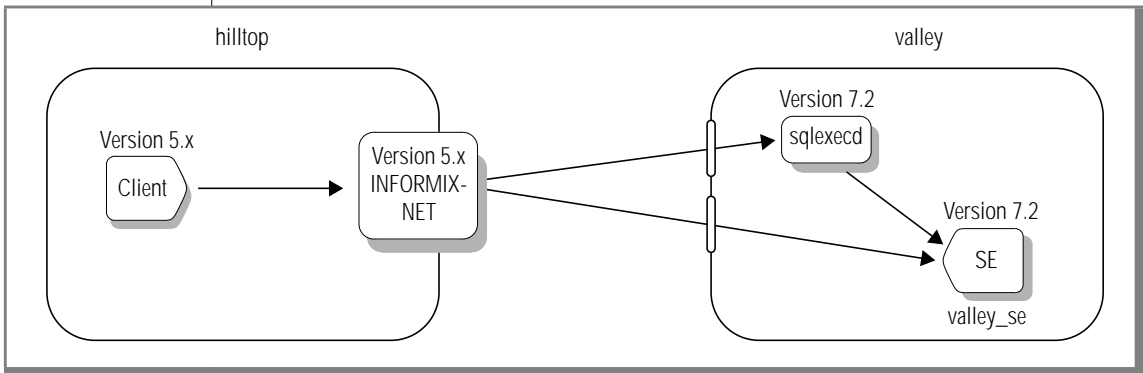


Network Connections with INFORMIX-NET

You can connect from an Informix Version 5.x client to a remote Version 7.2 SE database server using Version 5.x INFORMIX-NET. That configuration also applies to an Informix Version 4.1 client connecting to an Informix Version 7.2 database server using Version 4.1 INFORMIX-NET, or an Informix Version 4.1 client connecting to an Informix Version 7.2 database server using Version 5.x INFORMIX-NET. Figure 1-21 shows the configuration using Version 5.x INFORMIX-NET.

Figure 1-21

A Remote Connection Using Version 5.x INFORMIX-NET



Setting Up the hilltop and valley Host Computers

For the **hilltop** computer, set the **INFORMIXDIR** and **SQLXEC** environment variables to the values shown in Figure 1-22. For the **valley** computer, set the **INFORMIXDIR** environment variable to the value shown in Figure 1-22.

Figure 1-22

Environment Variables and Required Settings

Host Computer	Environment Variable	Value
hilltop	INFORMIXDIR	/usr/informix
hilltop	SQLXEC	sqlxec
valley	INFORMIXDIR	/usr/version7/informix

The **sqlxecd** daemon must be running. To start the daemon, log in as **root** and enter the following command:

```
/usr/version7/informix/lib/sqlxecd valley_se
```

For information about the **sqlxecd** daemon, refer to [“Starting the sqlxecd Daemon” on page 1-35](#).

Setting Up the Communication Files

Each host computer (**hilltop** and **valley**) must contain an **sqlhosts** file that includes connection information associated with the network interface and network protocol that you are using. Figure 1-23 shows the **sqlhosts** file entries for different network interfaces.

Figure 1-23
*Network Interfaces and Associated
sqlhosts File Entries*

Network Interface	sqlhosts File Entry
Sockets on TCP/IP	valley_se sesoctcp valley valley_service
TLI interface on TCP/IP	valley_se setlitcp valley valley_service

For TCP/IP connections, the **/etc/hosts** file must contain an entry for the **valley** host computer, and the **/etc/services** file must contain an entry for **valley_service**.

Do not confuse the **servicename** entry for a remote connection (as illustrated in Figure 1-21) with the **servicename** entry for a local connection (as illustrated in Figure 1-19). For a local connection, the service name indicates the SE process as a pathname or a program name. For remote connections, the service name in the **sqlhosts** file points to an entry in the **/etc/services** file.

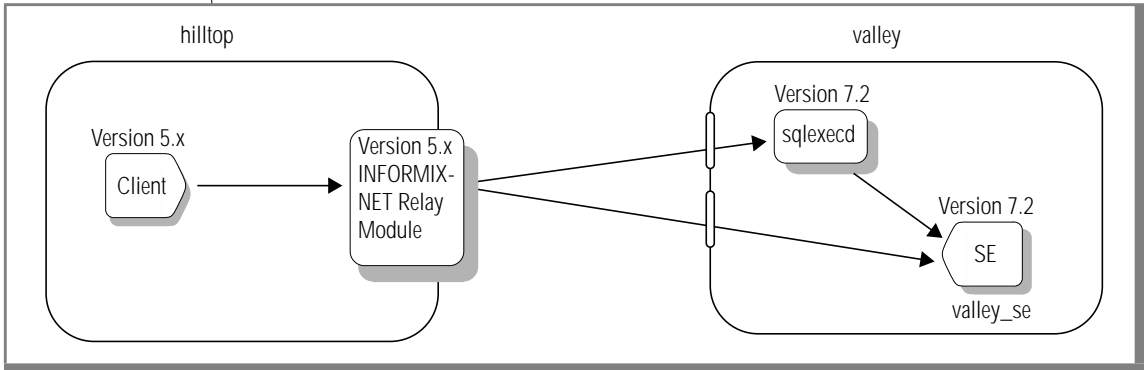
Network Connections with 5.x INFORMIX-NET Relay Module

Figure 1-24 shows an Informix Version 5.x client application connecting to a remote Version 7.2 SE database server using the Version 5.x INFORMIX-NET Relay Module.



Important: Informix does not provide a Relay Module with Informix Version 4.1 products.

Figure 1-24
A Remote Connection Using the Version 5.x
INFORMIX-NET Relay Module



Setting Up the hilltop and valley Host Computers

For the **hilltop** computer, set the **INFORMIXDIR**, **SQLXEXEC**, and **SQLRMDIR** environment variables to the values shown in Figure 1-25. For the **valley** computer, set the **INFORMIXDIR** environment variable to the value shown in Figure 1-25.

Figure 1-25
Environment Variables and Required Settings

Host Computer	Environment Variable	Value
hilltop	INFORMIXDIR	/usr/informix
hilltop	SQLXEXEC	sqlxec
hilltop	SQLRMDIR	\$INFORMIXDIR/lib
valley	INFORMIXDIR	/usr/version7/informix

The configurations shown in Figure 1-21 and Figure 1-24 do not use the **INFORMIXSERVER** environment variable. Only Informix Version 6.0 and later products use the **INFORMIXSERVER** environment variable. Because *only* Informix Version 5.x products reside on the client host computer (**hilltop**), you do not use **INFORMIXSERVER**.

You must set the **SQLRM** environment variable for the **hilltop** computer. Figure 1-26 shows the **SQLRM** environment variable entry for different network interfaces.

Figure 1-26
SQLRM Environment Variable Entries

Network Interface	SQLRM Environment Variable Entry
Sockets on TCP/IP	sqlrmsoc tcp
TLI interface on TCP/IP	sqlrmtl tcp

The **sqlxecd** daemon must be running. To start the daemon, log in as **root** and enter the following command:

```
/usr/version7/informix/lib/sqlxecd valley_se
```

For information about the **sqlxecd** daemon, refer to [“Starting the sqlxecd Daemon”](#) on page 1-35.

Setting Up the Communication Files

Each host computer (**hilltop** and **valley**) must contain an **sqlhosts** file that includes connection information that is associated with the network interface and network protocol that you are using. Figure 1-27 shows the **sqlhosts** file entries for different network interfaces.

Figure 1-27
Network Interfaces and Associated
sqlhosts File Entries

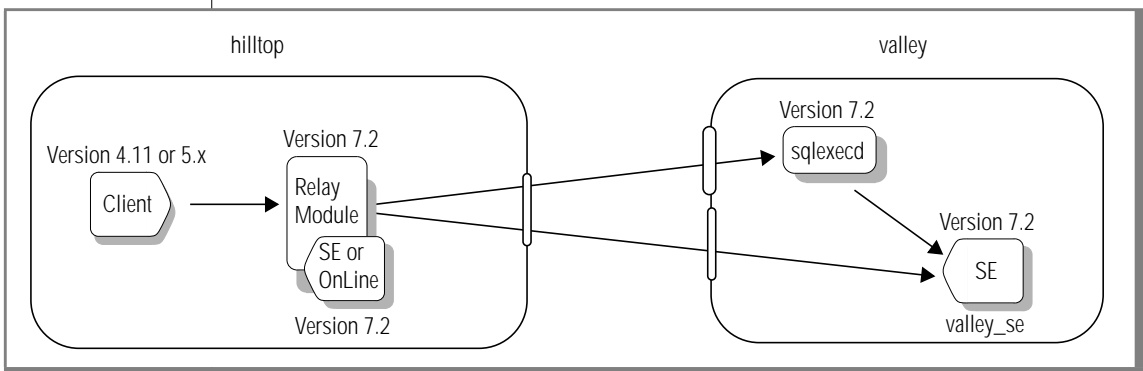
Network Interface	sqlhosts File Entry
Sockets on TCP/IP	valley_se sesoctcp valley valley_service
TLI interface on TCP/IP	valley_se setlitcp valley valley_service
TLI interface on IPX/SPX	valley_se setlispX valley valley_service

The **/etc/hosts** file must contain an entry for the **valley** host computer, and the **/etc/services** file must contain an entry for **valley_service**.

Network Connections with the Version 7.2 Relay Module

Figure 1-28 shows a configuration that connects an Informix Version 4.1 or 5.x client application to a remote Informix Version 7.2 database server using the Version 7.2 Relay Module. When an Informix Version 4.11 client application connects to an Informix Version 7.2 database server, you use the configuration shown in Figure 1-28.

Figure 1-28
A Remote Connection Using the
Version 7.2 Relay Module



In Figure 1-28, the Relay Module is the only component of the database server that remains *active* on the **hilltop** computer. The database server on the **valley** computer does not show a Relay Module component because the connection between **hilltop** and **valley** exists as a Version 7.2-to-Version 7.2 connection. You use the Version 7.2 Relay Module only for local connections between different versions.

Setting Up the hilltop and valley Host Computers

For the **hilltop** computer, set the **INFORMIXDIR**, **INFORMIXSERVER**, and **SQLEXEC** environment variables to the values shown in Figure 1-29. For the **valley** computer, set the **INFORMIXDIR** environment variable to the value shown in Figure 1-29.

Figure 1-29
*Environment Variables and
Required Settings*

Host Computer	Environment Variable	Value
hilltop	INFORMIXDIR	/usr/informix
hilltop	INFORMIXSERVER	valley_se
hilltop	SQLEXEC	/usr/informix/lib/sqlrm
valley	INFORMIXDIR	/usr/version7/informix

The **SQLEXEC** environment variable points to the executable file for the Version 7.2 Relay Module.

The **sqlxecd** daemon must be running. To start the daemon, log in as **root** and enter the following command:

```
/usr/version7/informix/lib/sqlxecd valley_se
```

For information about the **sqlxecd** daemon, refer to [“Starting the sqlxecd Daemon”](#) on page 1-35.

Setting Up the Communication Files

Each host computer (**hilltop** and **valley**) must contain an **sqlhosts** file that includes connection information that is associated with the network interface and network protocol that you are using. Figure 1-30 shows the **sqlhosts** file entries for different network interfaces.

Figure 1-30
*Network Interfaces and Associated
sqlhosts File Entries*

Network Interface	sqlhosts File Entry
Sockets on TCP/IP	valley_se sesoctcp valley valley_service
TLI interface on TCP/IP	valley_se setlitcp valley valley_service
TLI interface on IPX/SPX	valley_se setlispix valley valley_service

When you use a TCP/IP network, the **/etc/hosts** file must contain an entry for the **valley** computer, and the **/etc/services** file must contain an entry for **valley_service**.

When the host computer uses the TLI network interface on an IPX/SPX network, the **hostname** field contains the name of the NetWare file server instead of the computer host name. On the display screens that are associated with the preparation of the NetWare connections, the screen displays the NetWare file server name in uppercase letters; for example, **NW_SVR**. However, in the **sqlhosts** file, you enter lowercase letters; for example, **nw_svr**.

For an IPX/SPX connection, the value in the **servicename** field can contain an arbitrary string, but that string must represent a unique name among the names of services available on the IPX/SPX network. (In other words, you cannot use the string as a service name on any other file server on the network, even if that file server does not use an Informix product.) You can use the **dbservername** value in the **servicename** field, as shown in Figure 1-31, providing the **dbservername** value exists as a unique service name.

Figure 1-31
Using the `dbservername` Value in the `servicename` Field

Affected Field	<code>sqlhosts</code> File Entry
<code>servicename</code>	<code>valley_se setlisp nw_svr valley_se</code>

Starting the `sqlxecd` Daemon

Log in as **root** to start the **`sqlxecd`** daemon. The **`sqlxecd`** daemon enables SE to take the following actions:

- Receive a connection request from a remote client
- Establish a local-loopback connection with a local client

The command to start **`sqlxecd`** requires a valid `dbservername` value that corresponds to an entry in the **`sqlhosts`** file.

The **`sqlxecd`** daemon resides in the `$INFORMIXDIR/lib` directory. The following diagram illustrates the command syntax for starting the **`sqlxecd`** daemon.

```
sqlxecd _____ dbservername _____ -l logfile
```

Element	Purpose	Key Considerations
<i>dbservername</i>	Identifies the database server name of the database server.	Restrictions: The <i>dbservername</i> argument must correspond to a <i>dbservername</i> entry in the sqlhosts file.
<i>-l logfile</i>	Maintains records on all client connection activity.	Restrictions: When you specify the <i>-l</i> option, you must also specify the name of a log file. Additional Information: If no log file exists, SE creates a log file for you and gives it the name you specify in the <i>-l</i> option. If the log file already exists, SE appends the new client connection activity information to the existing log file. If you are using an existing log file, make sure it has sufficient space for new client connection activity information. You can specify the <i>-l</i> option to start the sqlxecd daemon with a log file in the current directory. You can also specify the full pathname of the log file when the log file does not reside in the current directory.

The following example illustrates how to start **sqlxecd** with the *-l* option:

```
$INFORMIXDIR/lib/sqlxecd acctg_tcp -l mylogfile
```

In this example, *acctg_tcp* represents the name of the database server, and **mylogfile** represents the name of the log file.

The SE database server adds a date/time stamp to the **sqlxecd** log file to provide timing information for database server event tracing. The following table shows an example of a log file and its format.

date	time	sqlxec	clientname	user	operation/ dbpathname
1994-02-23	23:58:30.123456	sqlxec	stationconn6	leeai	-d/work/payroll
1994-02-23	23:59:22.321342	sqlxec	stationconn2	markl	-d/mis
1994-02-24	00:01:04.324155	sqlxec	stationstan5	usr98	-x/db/inventory



Important: The *microseconds* component is operating-system dependent and does not appear on all platforms.

The following list describes the operations that can appear in the log file:

- c creates a database.
- d selects a database.
- n represents the remote network server that INFORMIX-STAR accesses.
- p accesses from a client with a password.
- r removes a database.
- s starts an executable file other than **sqlexec**.

Using an NFS-Mounted Directory

When you want your database to reside across a network on a Network File System (NFS), SE allows you to designate the current directory as an NFS-mounted directory. When you designate your current directory as an NFS-mounted directory, you must take one of the following actions:

- When you are using a pipes connection, you must be in a local directory. When the NFS-mounted directory is not essential for running your application, change your current directory to a local directory that does not reside on NFS.
- When you are using a network connection (that is, TCP/IP or IPX/SPX) ask your Informix database administrator (DBA) to make sure that an SE **sqlexecd** daemon is set up and running on the host computer where the NFS-mounted directory resides. Also, make sure that the *dbservername* for the local SE database server and the remote SE database server resides in the **sqlhosts** files on both hosts.

How Does a Client Application Connect to a Database Server?

A client application establishes a connection with a database server by using the SQL statements `CONNECT` or `DATABASE`. For example, to connect to the database server **my_server**, your client application might use the following form of the `CONNECT` statement:

```
CONNECT TO '@my_server'
```

For more information about the `CONNECT` and `DATABASE` statements, see the [Informix Guide to SQL: Reference](#).

INFORMIX-SE System Architecture

SE Program Files	2-3
SE System Files	2-4
The .dat File	2-5
The .idx File	2-6
Allocating Space for .dat and .idx Files.	2-6
Transaction-Log Files	2-6
Allocating Space for the Transaction-Log File	2-8
Audit-Trail Files	2-8
Permissions of Database Files and Directories	2-10
Determining If a Database Is ANSI Compliant	2-12

T

his chapter includes the following topics:

- INFORMIX-SE program files
- INFORMIX-SE system files
- Transaction-log files
- Audit-trail files
- Permissions of database files and directories
- Determining if a database is ANSI compliant

SE Program Files

When you install SE, the installation program creates several directories and subdirectories. Refer to Figure 2-1 for a list of those directories and their descriptions.

*Figure 2-1
Descriptions of Directories Created During Installation*

Directory	Description
\$INFORMIXDIR	Contains the SE product installation script
\$INFORMIXDIR/bin	Contains the binary executable files of utility programs and the demonstration script
\$INFORMIXDIR/lib	Contains the database server product libraries, the Informix product communication files, the Informix SQL API or application development tool libraries (after you install your SQL API or application development tool), and the software you need to access the database server

(1 of 2)

Directory	Description
\$INFORMIXDIR/etc	Contains miscellaneous files, common installation scripts that the product installation script in the \$INFORMIXDIR directory calls, and branding scripts
\$INFORMIXDIR/msg	Contains the binary-readable error and warning message files of Informix products
\$INFORMIXDIR/demo	Contains subdirectories that contain product-specific command files and the application examples for the stores7 demonstration database
\$INFORMIXDIR/release	Contains the documentation notes, machine notes, and release notes files

(2 of 2)

The installation procedure sets up **\$INFORMIXDIR** and all Informix program files with operating-system permissions that enable any user to run the installed program. When you want to restrict access to the **\$INFORMIXDIR** directory, change permissions on the directory *after* you run the installation procedure. Do not alter owner, group, or system permissions on Informix files. For more information, see [“Permissions of Database Files and Directories” on page 2-10](#).

SE System Files

GLS

For information about how to enable SE to generate filenames with multibyte characters, refer to the [Guide to GLS Functionality](#). ♦

When you create an SE database, SE creates a directory for the database with a **.dbs** extension in the current directory. For example, if you create a database that is named **stores7**, SE creates a directory in the current directory that is named **stores7.dbs**. For each table that you create in a database, SE creates two files, a data file and an index file. SE automatically stores the files in the **.dbs** directory. SE names the two files with the first five characters of the table name, a unique number starting at 100, and the extensions described in Figure 2-2.

Figure 2-2
File Extensions for Data and Index Files

File Extension	Description
.dat	Represents the extension for data files
.idx	Represents the extension for index files

When the table name has less than five characters, SE pads the table name with underscores (_) to create the five character table-name portion of the filename. When the table name has more than five characters, SE truncates the table name at five characters. Therefore, filenames can appear in different forms, as shown in Figure 2-3.

Figure 2-3
INFORMIX-SE Filename Forms

Table Name	Data Filename Form	Index Filename Form
cost	cost_00100.dat	cost_00100.idx
customer	custo00102.dat	custo00102.idx
stock	stock00103.dat	stock00103.idx
stockitem	stock00104.dat	stock00104.idx

The .dat File

The **.dat** file stores all data for a single table. All files contain fixed-length data rows and a delete flag at the end of each record. If the flag equals zero (ASCII null), SE deletes the record.

The .idx File

The **.idx** file stores information about all the indexes for a single table. For tables that contain a SERIAL field, SE stores the highest SERIAL value in the **.idx** file.

Creating an index is not the same as creating an index file. An **.idx** file exists for every table, even if the table does not include an index. Each index file represents a collection of *pointers* to the data in the **.dat** file. For information about the organization, structure, and format of index structures, refer to [Chapter 4, “INFORMIX-SE Indexing.”](#)

Allocating Space for .dat and .idx Files

The SE database server allocates space for the **.dat** and **.idx** files one *block* at a time. Refer to your operating-system documentation for details about block allocation. A block is disk space that your operating system allocates in units of a specific size. The type of computer and the type of operating system on that computer determine block size. Typically, a block ranges from 4 kilobytes to 512 kilobytes.

SE relies on ISAM for its organization and access method, allocating a 1-kilobyte block at a time for an **.idx** file and one record at a time for a **.dat** file. To estimate your space requirements, you need to know your operating-system block size and your data row size. Once you know these sizes, you can calculate the number of records per block. To estimate block use and growth rates, you must first successfully estimate the ratio of inserts to deletes. You must also allow for index size and growth. As indexes grow, they can consume large portions of space.

Transaction-Log Files

A transaction-log file contains records of modifications that were made to the database. Transaction logging provides the following benefits:

- You can treat a series of operations as a single unit of work.
- You can recover a database when the data becomes corrupted due to a disk crash or other event.

The effort that is directed into maintaining a log creates a potential drawback to transaction logging.

Transaction logging does not start automatically when you create the database. You can specify whether to use transaction logging when you create the database or at any time thereafter. See [“Transaction-Log Files” on page 3-4](#) for information about initiating and maintaining transaction-log files.

A transaction-file record contains a fixed-length header and other information, depending on the transaction type. The transaction-file record header has a length of 18 bytes. Figure 2-4 illustrates the header format.

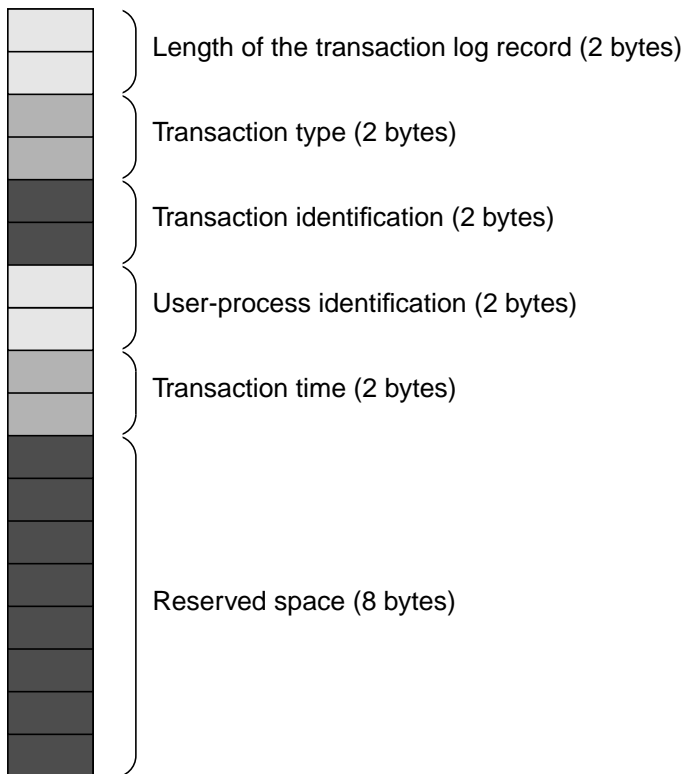


Figure 2-4
Transaction-File
Record Header
Format

Allocating Space for the Transaction-Log File

You allocate space for the transaction-log file one block at a time. Depending on the number and type of transactions that you perform, the transaction-log file can be huge. Purge the transaction-log file on a regular basis, as explained in [“Maintaining a Transaction-Log File” on page 3-5](#).

Audit-Trail Files

An audit-trail file contains a history of additions, deletions, and updates made to a database table. Audit trails let you record modifications to a single important table without maintaining a transaction log on the entire database.

Tip: When you use transaction logging on a database, you can concurrently use audit-trail files on the tables in that database.

Figure 2-5 illustrates the differences between audit trails and transaction logs.



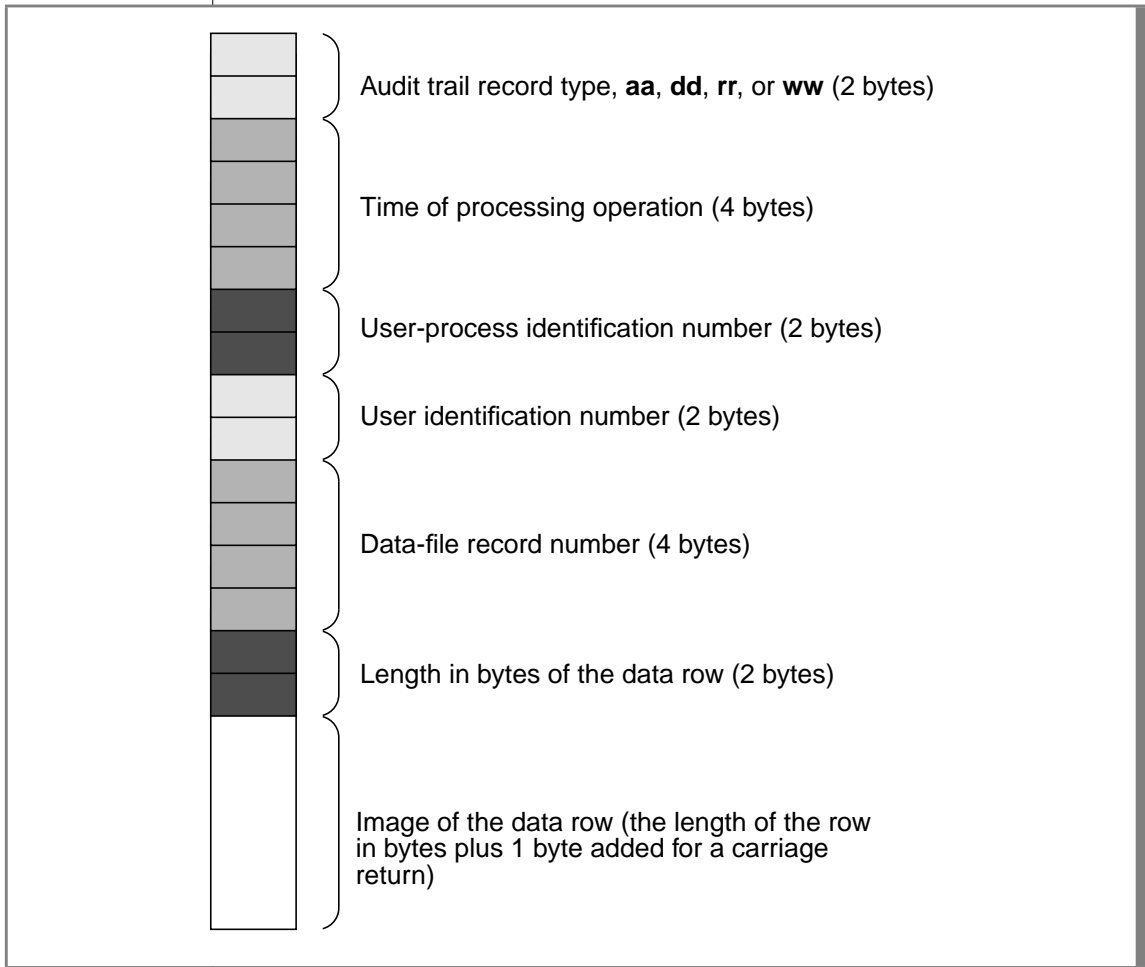
Figure 2-5
Differences Between Audit Trails and Transaction Logs

Audit Trail	Transaction Log
Records modifications to a single table	Records modifications to an entire database
Does not protect against partial completion of SQL statements	Protects against partial completion of SQL statements
Can recover a single table	Can recover an entire database
Does not incur the effort of maintaining a log for an entire database	Incurs the effort of maintaining a log for an entire database

Use audit trails when you have only one or a few critical tables and you do not need the additional benefits that transaction logs provide. When you need to maintain the integrity of the database as a whole, or to guarantee that you execute SQL statements either completely or not at all, you must use transaction logs.

The audit-trail file contains records that consist of a fixed-length header and an image of a data row. Figure 2-6 illustrates the format of audit-trail files.

Figure 2-6
Format of an Audit-Trail File



Important: You can create only one audit trail for each table.

For an insert operation, SE designates the audit-trail record as type **aa**. For a delete operation, SE designates the record as type **dd**. For a rewrite operation, SE records both the *before-* and *after-images* in an audit-trail file. SE lists the before-image first, as type **rr**, then lists the after-image, as type **ww**. SE assigns the same record number to both.

The RECOVER TABLE statement in the *Informix Guide to SQL: Syntax* partially describes the recommended procedure for making archive copies of a database that includes audit trails and the procedure for recovering a table using audit trails.

Important: An Informix audit-trail file exists as a binary-readable file. You cannot read that file like a transaction-log file.



Permissions of Database Files and Directories

When you use Informix programs to create a database and tables, SE assigns default UNIX permissions to the database directory and its files. Figure 2-7 shows a partial listing of the default permissions placed on the **stores7** demonstration database files. The UNIX command `ls -l` produces the first permissions line about the **stores7** database directory. The UNIX command `ls -l stores7.dbs` produces the remaining permissions lines about the files within the **stores7** database.

Figure 2-7

Default UNIX Permissions on stores7 Database Files

```
drwxrwx--- 2 owner      1536 Feb 12 09:11 stores7.dbs /
.
.
.
-rw-rw---- 1 owner      3072 Feb 12 09:11 custo00100.dat
-rw-rw---- 1 owner      4096 Feb 12 09:11 custo00100.idx
-rw-rw---- 1 owner      1024 Feb 12 09:11 items00102.dat
-rw-rw---- 1 owner      4096 Feb 12 09:11 items00102.idx
-rw-rw---- 1 owner      1024 Feb 12 09:11 manuf00105.dat
-rw-rw---- 1 owner      3072 Feb 12 09:11 manuf00105.idx
-rw-rw---- 1 owner      2048 Feb 12 09:11 order00101.dat
```

The default file permissions give access to the owner and the group, but not to other users. (To see both owner and group names, execute the UNIX command `ls -lg`.) The user who creates database files and directories also owns them. SE automatically grants read and write permission to members of the group **informix**.

SE assumes the group id of **informix**, which can access any database file on the system, enables any user with database access privileges to read and write to a database directory and its tables, regardless of system permission.



Warning: Do not designate any user of Informix products as a member of group **informix**. This action can lead to unintended and uncontrolled database access.

Do not alter these permissions. When you move a database to another directory, check the UNIX file permissions to verify that those permissions were properly set.

Although SQL database and table privileges govern user access to the database and its tables, you can also apply UNIX permissions to the directory where the database resides. You use the GRANT and REVOKE statements to establish SQL privileges. For information on those statements, refer to the [Informix Guide to SQL: Syntax](#).

You can change the operating-system permissions for the *parent* directories of the database directory. When you restrict read permissions on the parent directories, other users cannot list the database directory at the operating-system level. Restricting public access to any parent directory of the database directory prevents all other users from accessing the database through an Informix program.



Important: On some operating system platforms, you must start your client application from a directory in which you have at least execute permission by user or others. Having access permission by group only is not sufficient because SE assumes the group id of **informix**, not the group id of the user. Execute permission is required for searching a directory.

Determining If a Database Is ANSI Compliant

You can query the **systables** system catalog table to determine if a database is ANSI compliant. When SE creates a database with the **MODE ANSI** keywords, **systables** lists a row with a **tablename** of **ANSI** and a **tabid** of **100**. For example, if the following query on **systables** returns a row, the database is ANSI compliant:

```
SELECT * FROM 'informix'.systables WHERE tablename = 'ANSI'
```

For a description of the Informix system catalog, refer to the [Informix Guide to SQL: Reference](#).

Basic Administration and Maintenance

Monitoring Disk-Space Use	3-3
Maintaining Data Integrity	3-4
Transaction-Log Files	3-4
Creating a Transaction-Log File	3-4
Maintaining a Transaction-Log File	3-5
Turning Off a Transaction-Log File	3-5
Audit-Trail Files	3-6
Creating Backups	3-6

T

his chapter discusses the following administrative and basic maintenance issues:

- Monitoring disk space
- Maintaining data integrity
- Creating backups

Monitoring Disk-Space Use

To monitor INFORMIX-SE disk space, monitor the size of the files where you store the data and indexes. SE files cannot span disk partitions. When your partition becomes full, you probably need to perform one of the following procedures:

- Move your data to another partition
- Back up your data, increase the size of the partition, and then restore the data

Monitoring disk space allows you to delete unnecessary files and reclaim space before the partition becomes full. Use UNIX operating-system commands to monitor the disk space (**du** or **quot** utilities). For information on these utilities, see your UNIX **man** pages.

SE eventually reuses space from deleted rows, but you can make that space available immediately with the ALTER TABLE statement. For information about the ALTER TABLE statement, refer to the [Informix Guide to SQL: Syntax](#).

Warning: To use the ALTER TABLE statement, enough disk space must exist to store the new table and the old table concurrently. However, SE automatically drops the old table after you create the new one, causing only a momentary need for space.



Maintaining Data Integrity

Data integrity means that correct data resides in a database and that the database management system (DBMS) can recover from errors. With SE, you implement *transactions* to support data integrity. A transaction represents a series of database operations. When *all* the database operations in a transaction succeed, the transaction succeeds. When *any* database operation in a transaction fails, the transaction fails and the effects of all successful database operations in that transaction are rolled back to the state that existed before the transaction began. For more information about transactions, refer to the [Informix Guide to SQL: Tutorial](#). SE provides transaction-log files and audit-trail files to protect data integrity.

Transaction-Log Files

A transaction-log file serves as a record of operations that were performed on data stored in a database. These operations include inserts, updates, and deletes.

You can use the CREATE DATABASE statement to request either unbuffered logging (with the WITH LOG IN keywords) or ANSI-compliant logging (with the MODE ANSI keywords). In the event of failure, unbuffered logging ensures that you lose only the single alteration in progress at the time of failure. ANSI-compliant logging is the same as unbuffered logging, but the ANSI rules for transaction processing are also enabled. For further information about ANSI SQL, refer to the [Informix Guide to SQL: Syntax](#).

For more information about transaction logs, refer to “[Transaction-Log Files](#)” on page 2-6.

Creating a Transaction-Log File

Users create transaction-log files with the CREATE DATABASE or START DATABASE statements. For information on these statements, refer to the [Informix Guide to SQL: Syntax](#).

Make sure that the transaction-log file resides on a different physical disk drive than the one on which the database itself resides. This arrangement allows you to recover from failure of either disk drive.

Maintaining a Transaction-Log File

The transaction-log file can become quite large. Periodically, back up the log file on tape and initiate another log file. Make sure that you back up your database and its associated log file *before* you initiate a new log file. In general, you must save every transaction-log file with a corresponding backup copy of the database.



Important: *Make sure that no other users are using the system when you back up the database and log file, or when you create a new log file.*

After backing up the database and the log file (see “[Creating Backups](#)” on [page 3-6](#)), you must specify a new log-file name. To reuse the same log-file name, create an empty log file with the same name as the old one, as shown in the following example:

```
cat /dev/null > log_file_name
```

To learn the location of the transaction-log file, invoke DB-Access and execute the following SQL statement:

```
SELECT dirpath FROM systables WHERE tabid = 0
```

To change the name of the log file for some reason related to your environment, execute the START DATABASE statement just before you make a backup of the database and specify the new log-file name.



Important: *Before you execute the START DATABASE statement, make sure that no other users are using the same database; then issue a CLOSE DATABASE statement before you create and start a transaction log.*

Turning Off a Transaction-Log File

To stop transaction logging on an SE database, issue the START DATABASE statement with the WITH NO LOG clause for that database.

Audit-Trail Files

An audit trail is a record of transactions that were processed in a table.

Users create audit-trail files with the CREATE AUDIT statement. For information about the CREATE AUDIT statement, refer to the [Informix Guide to SQL: Syntax](#).

Make a backup copy of the database files that are associated with a table immediately after you create the audit trail. When possible, store the audit-trail file on a different physical device from the one that holds the data, so that a failure of one does not affect the other



***Important:** Before you make backup copies of database files and create an audit trail, make sure that no other users are using the same table.*

The RECOVER TABLE statement in the [Informix Guide to SQL: Syntax](#) describes how to back up a database that includes audit trails and how to recover a table using audit trails.

Creating Backups

Create backups on a regular basis and during times when you do not use the database. This way, when you lose files and must recover from backups, your data will be as current as the latest backup and the logical logs that you saved to tape.

Use the UNIX utility of your choice (for example, **dump**, **tar**, or **cpio**) to back up your database and the associated SE files.

INFORMIX-SE Indexing

B+ Tree Organization	4-3
Searching for a Row	4-5
Adding Keys	4-6
Removing Keys.	4-11
Index-Table Structure	4-12
Multiple Indexes	4-13
Index-Table Formats	4-15

T

he INFORMIX-SE database server stores data in indexed sequential access method (ISAM) tables. Indexing allows quick access to specific rows in the ISAM table and creates an order for sequential processing of the table. This chapter discusses ISAM indexes and covers the following topics:

- Index organization and B+ trees
- Index-table structure
- Index-table format

ISAM maintains indexes so that programs can find rows quickly. You can add, delete, or modify the index keys with minimum impact on the performance of the programs that use the table. SE knows which indexes exist and can be used. Read this chapter if you need to know how SE implements indexing. You do not need this information to administer SE.



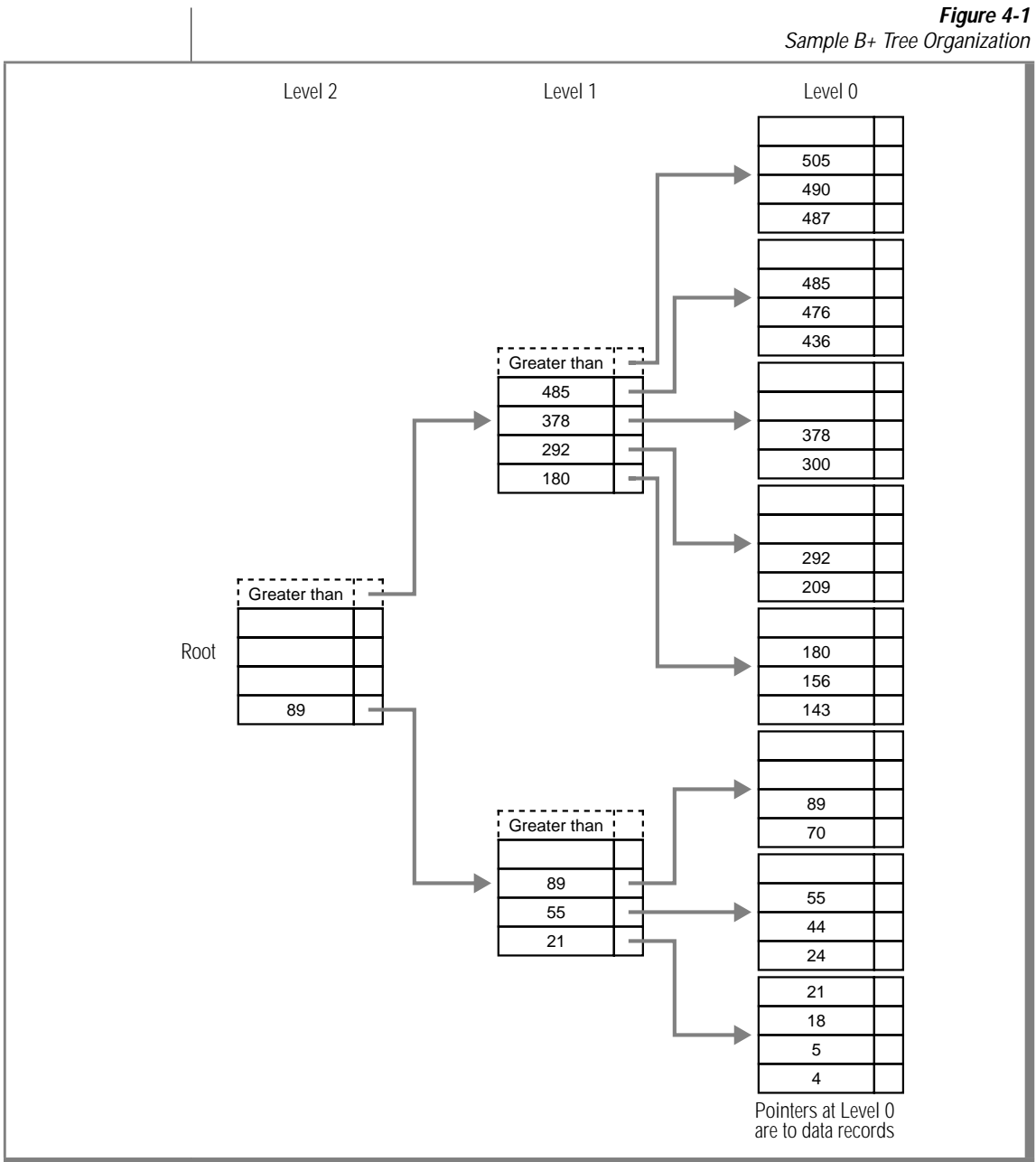
Important: In this chapter, blocks and nodes represent the same thing.

B+ Tree Organization

ISAM organizes indexes in B+ trees. A B+ tree is a set of nodes that contain keys and pointers and that are arranged in a hierarchy. A *key* is a value from the data row; for example, an employee number. The pointer points either to another node in the tree or to a data row. The *root* node resides at the top of the tree-structure hierarchy.

Figure 4-1 illustrates this hierarchy for a hypothetical index. The numbers in the nodes are the keys that you can also find in the data rows. The arrows represent the pointers. Unused nodes represent empty parts of the B+ tree.

Figure 4-1
Sample B+ Tree Organization



SE logically organizes the nodes into levels. Level 0 contains a pointer to each data row. At levels higher than zero, the pointer for each key points to a node one level down that contains keys that are less than or equal to the key at the higher level.

At levels higher than zero, a node can contain an additional pointer that is not associated with a specific key. When it exists, it points to a node that contains keys that are greater than the largest key in the higher level node. A node always has at least as many pointers as it has keys.

Figure 4-1 shows space for only four keys in each node. In reality, SE puts as many keys as possible in each node. The maximum number of keys in different nodes can vary because SE allows keys to vary in length.

Consider the root node in Figure 4-1. It has only one key, with the value 89. Two pointers reside in the root. One points to a node that contains keys with values less than or equal to 89. The other points to a node that contains keys with values greater than the values in this node; in this case, values greater than 89.

Levels indicate the distance, in nodes, between a node and the pointer to an actual data row. In Figure 4-1, the root node resides at Level 2. For nonzero levels, SE directs pointers to index nodes at a lower level.

The pointers at Level 0 point to rows in the data table; they do not point to nodes in the index table. SE ensures that Level 0 represents every key whether or not the B+ tree represents a key at a higher level.

Searching for a Row

To begin accessing a specific row in an ISAM table, a function compares the search value with the keys in the root node. The search value represents the key passed to the function. The function follows the appropriate pointers to the Level 0 node. At Level 0, when a key matches the search value, the key pointer points to the data row. When no match occurs at Level 0, the data row does not exist.

For example, take a search value equal to 44, and use Figure 4-1 to trace the path a function takes to find the row. The function examines the root first and then follows the less-than-or-equal-to pointer for key 89 because 44 is less than 89. Next, the function examines the node on Level 1 that contains keys 21, 55, and 89. The function follows the pointer for key 55 because 44 is less than 55 but greater than 21. The Level 0 node contains keys 24, 44, and 55. Because a match occurs at Level 0, the function finds the data row by following the pointer for key 44.

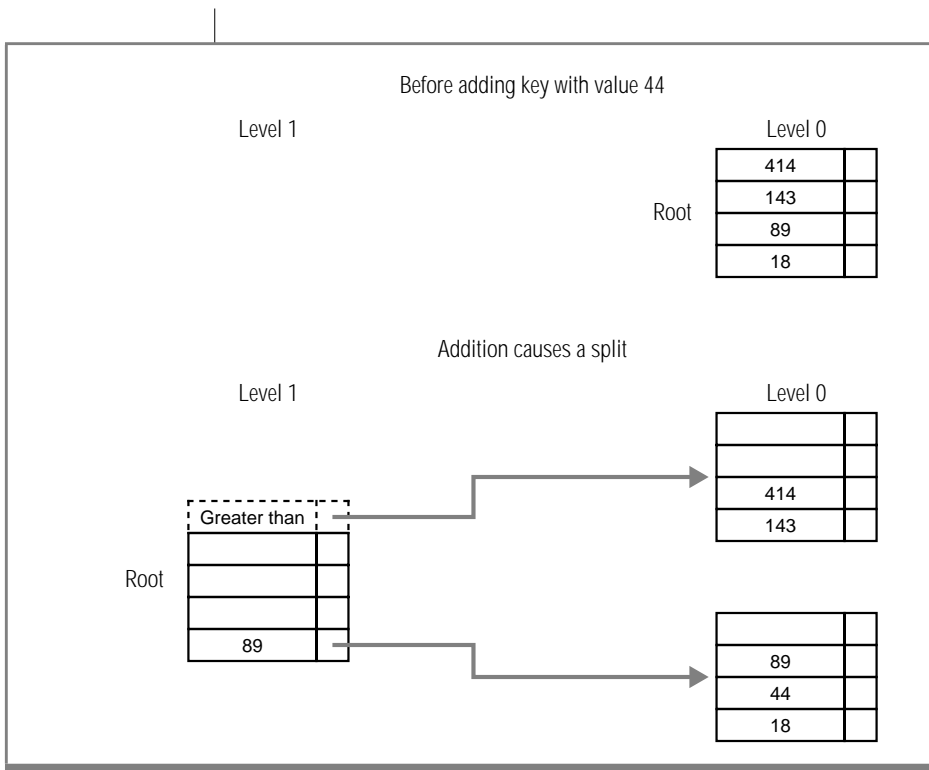
Repeating the process for a search value equal to 475, the function examines the root and follows the greater-than pointer for this node because 475 is greater than 89, the largest key in the node. The node at Level 1 contains keys 180, 292, 378, and 485. The function follows the less-than-or-equal-to pointer from key 485 because 475 is less than 485 but greater than 378. Level 0 represents the keys 436, 476, and 485. Because no key matches the search value 475, a data row does not exist.

Adding Keys

When you create a table, the index contains no indexing information but it does contain dictionary information about the table. (See [“Index-Table Structure” on page 4-12](#).) Figure 4-2 shows a B+ tree that can hold only four keys per node. In Figure 4-2, the first four keys are added (18, 89, 143, and 414) to the root node. Each key entry points to a data row because the root node resides at Level 0.

When you add the next key, with a value of 44, the completely full node splits to accommodate the new key.

Figure 4-2
Growth of a B+ Tree



SE splits a node by finding the middle value of the keys in the node, including the value of the key that causes the split. SE puts approximately half the entries into a new node and keeps the remaining entries in the original node. These two nodes still reside in Level 0 after the split, and their keys still point to data rows. SE promotes the middle value of the keys, 89 in this case, to the next higher level.

Because no higher level node exists to receive the promoted value, SE creates a new root node. The new root node resides on Level 1, and the pointer for key 89 points to the original node. (The original node now contains the keys that are less than or equal to 89.) SE forms another pointer directed toward the new Level 0 node. This Level 0 node contains keys that are greater than the highest key value in the next higher level node; in this case, 89, in the Level 1 node.

B+ trees grow toward the root from the lowest level, Level 0. Attempting to add a key into a full node forces a split into two nodes and promotion of the middle key value into a node at a higher level. The promotion of a key to the next higher level can also cause a split in the higher level node, even if this higher level is the root. When the root node splits, the tree grows by one level and creates a new root node.

When a split occurs, approximately half the entries remain in the original node, and the remainder are transferred to a new node. This process leaves approximately half of each node available to accommodate additional entries. This strategy is useful when the new key values have a random distribution.

If rows are added in sequential order, this splitting strategy creates half-full nodes that never receive other keys. More space is needed to store all the keys, and the tree requires more levels to index the same number of data rows.

Figure 4-3 shows what happens when you add the key values 415 through 426 sequentially to the B+ tree in Figure 4-2, using the splitting algorithm for the random case.

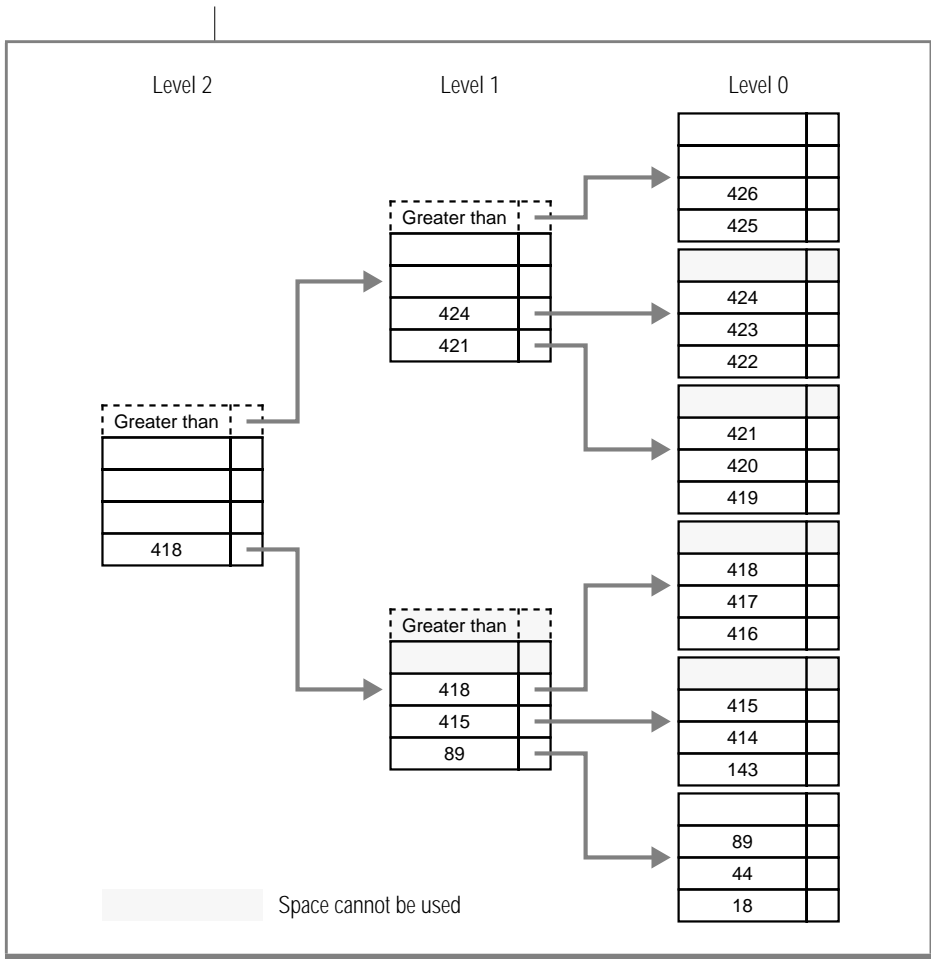


Figure 4-3
Wasted Space
in B+ Trees

To avoid this problem, SE uses a different strategy. When the value that causes the split is greater than the other keys in the node, SE puts that value into its own node during the split.

Adding Keys

Figure 4-4 shows a split caused by adding the key values 415, 416, and 417 to the B+ tree in Figure 4-2.

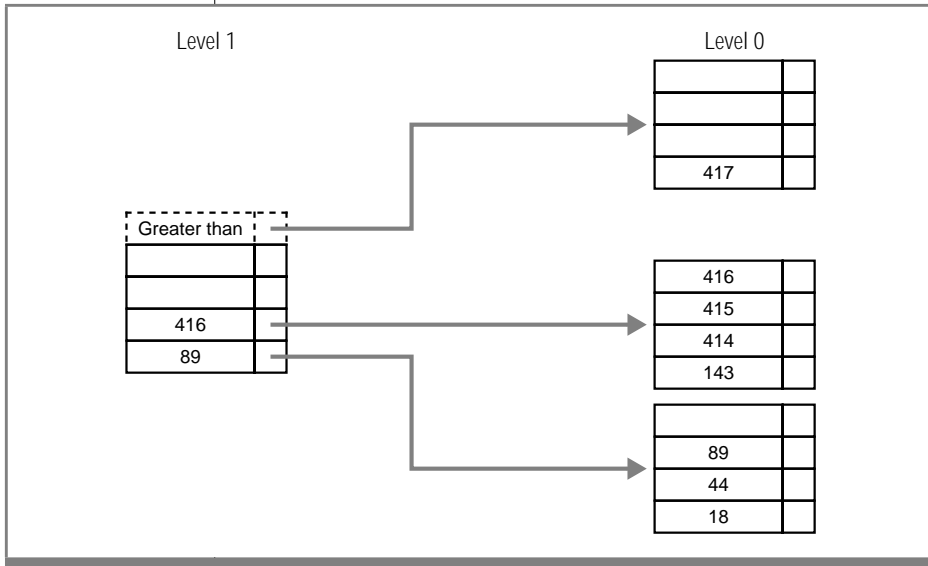
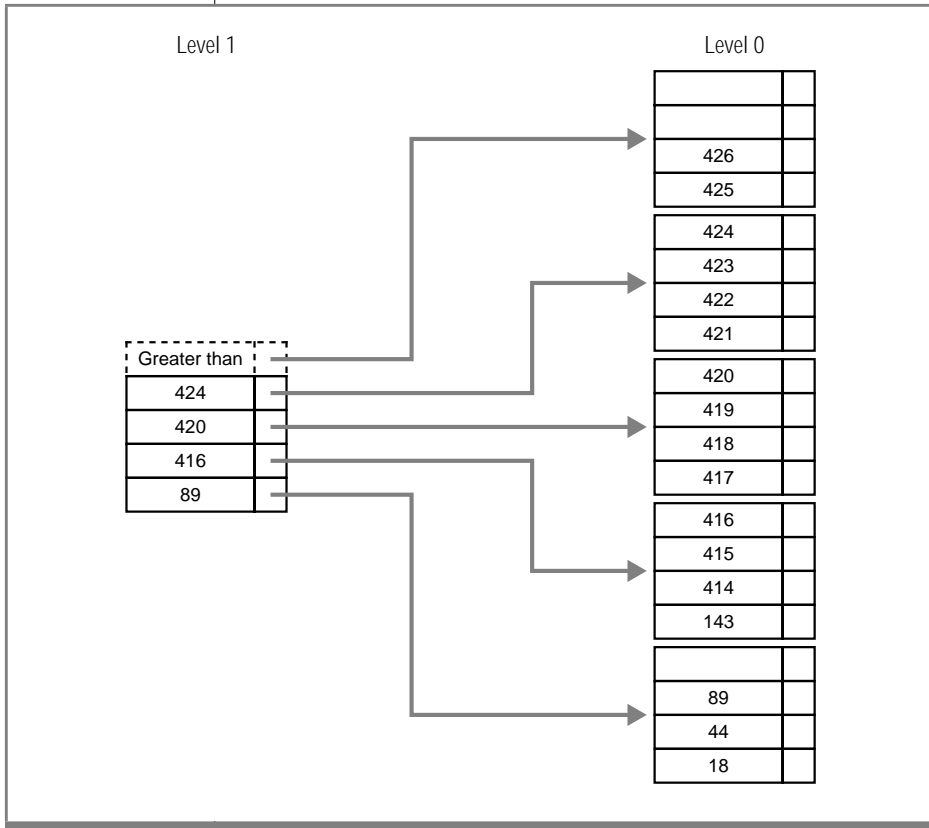


Figure 4-4
*Efficient Growth
of B+ Trees*

Figure 4-5 shows the effect of this strategy when you add key values 415 through 426 to the B+ tree from Figure 4-4.

Figure 4-5
Efficient Sequential
Addition of Keys to a
B+ Tree



Removing Keys

When you delete a row, SE removes the key from the index. When SE removes all keys in a node, the node becomes free. SE maintains a list of free nodes (see “[Index-Table Structure](#)” on page 4-12) and recycles free nodes. ISAM indexes do not require reorganization.

Index-Table Structure

SE stores the index nodes and control information in operating-system tables with the `.idx` extension. The data table stores only data rows.

The index table always contains the following kinds of nodes:

- A dictionary node
- Key-description nodes
- Index nodes containing keys and pointers (B+ tree nodes)
- Free-list nodes
- Audit trail nodes

A one-to-one correspondence usually exists between nodes and the unit of transfer between the disk and memory. The unit of transfer is called a *block*. “[Index-Table Formats](#)” on page 4-15 documents the index-table nodes.

Each index table contains one *dictionary block*. This block contains pointers to all the index nodes in the index table and also contains other information about the ISAM table. Figure 4-6 shows the relationships among the various nodes in the index table.

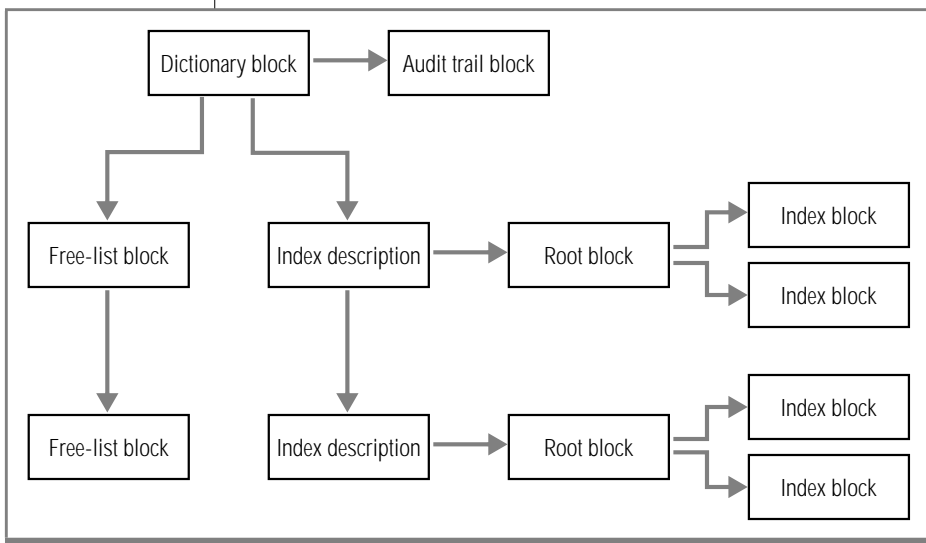


Figure 4-6
Index-Table
Structure

The dictionary block points to the first *key-description block* (marked as “Index description” in Figure 4-6). Each key-description block stores information about the indexes, including the address of the *root block* for each index. All other blocks for an index are addressed using its root block. SE chains ISAM key-description blocks together, and you can find any index root by following the chain from the dictionary block.

The dictionary block also contains a pointer to the first *free-list block* for the **.idx** table. Free-list blocks are chained together. The free-list block holds the block numbers that are unused within the table.

When an index block becomes free, SE places the block number on the free list. When SE needs a new block, SE examines the free list first. SE removes the block number of an available block from the list and reuses that block. SE uses all free blocks before it extends the length of the table.

Multiple Indexes

Indexing allows fast access to specific rows in an ISAM table. Changes to an index, however, require SE to update the index. Maintenance of the index imposes an overhead on the use of the table.

When you add a row to a table that contains only one index, the operation requires a maximum of five disk operations: three to read the index to determine that the row did not exist, one to add the row to the data table, and one to update the index. The five disk operations perform the following specific tasks:

- Read the index dictionary node to verify that your buffer cache remains intact
- Read the data row-free list
- Read in the data page
- Write back the data page
- Write back the dictionary node indicating that a new row exists in the table

If you create two indexes on the table, the number of disk operations, in the worst case, can reach nine: four for each index and one for the data row.

The root level of the index and the level that the root points to are often in memory because the operating system buffers the most-recently used index blocks. Therefore, two fewer disk operations are required per update for each index. The updates that occur in key sequence reduce the overhead.

A linear relationship exists, however, between the time it takes to update a row and the number of indexes that SE must update. It takes twice as much time to update a table with two indexes as the same table with only one index.

When you need additional indexes, consider creating the index that you need before processing, then deleting it after you finish. For example, use this method when you want to process the table in a different order at the end of each day.

If you read only rows, or rewrite rows without changing any key fields, the number of indexes does not affect the speed of processing. However, the read operations sometimes work faster on an update that uses a two-operation read and write process.

To build an index

1. Create a table.
2. Create the index.
3. Load the data.

To build an index when data is sorted

1. Create a table.
2. Load the data.
3. Create the index.

Index-Table Formats

This section provides format information about the following five nodes that are found in ISAM index (.idx) tables:

- Dictionary node
- Key-description node
- Index node that contains keys and pointers (B+ tree node)
- Free-list node
- Audit-trail node

Figures 4-7 through 4-11 provide the byte and value information of the preceding nodes.

Figure 4-7 displays the byte offsets, the lengths (in bytes) of the offsets, and the description of the meaning of the bytes for the dictionary node.

Figure 4-7
Dictionary-Node Format

Byte Offset	Number of Bytes	Item	Value
0	2	Validation	FE53
2	1	Number of reserved bytes at start of index node	2
3	1	Number of reserved bytes at end of index node	2
4	1	Number of reserved bytes per key entry—includes row number	4
5	1	Reserved	4
6	2	Index table node length - 1	(511 or 1023)
8	2	Number of keys	
10	2	Reserved	
12	1	Table version number	

(1 of 2)

Byte Offset	Number of Bytes	Item	Value
13	2	Data row length in bytes	
15	4	Index node number of first key description	
19	6	Reserved	
25	4	Index node number of free data row list	
29	4	Index node number of free index node list	
33	4	Row number of last row in data table	
37	4	Index node number of last node in index table	
41	4	Transaction number	
45	4	Unique id	
49	4	Pointer to audit trail information	
53	2	Lock method	
55	4	Alter table count	
59	2	Reserved	
61	4	Reserved	
65	4	Reserved	
69	4	Reserved	
73	4	Reserved	
77	4	Reserved	

(2 of 2)

In addition, when you create a table with a serial column, SE stores the maximum serial number in the dictionary page of the index table (offset 45).

Figure 4-8 displays the byte offsets, the lengths (in bytes) of the offsets, and the description of the meaning of the bytes for the key-description node.

Figure 4-8
Key-Description Node Format

Byte Offset	Number of Bytes	Item	Value
0	2	Number of bytes used in this node	
2	4	Index node for continuation of key descriptions	
6	2	Length of description	
8	4	Index node number of root	
12	1	Compression flags	
13	2	Length of key part 1 (top bit = duplicates)	
15	2	Position in data record	
17	1	Data type parameter	
n-2	1	Flag	FF
n-1	1	End of key description node	7E

Repeats for each key

Repeats for each part of the key

Figure 4-9 displays the byte offsets, the lengths (in bytes) of the offsets, and the description of the meaning of the bytes for the B+ tree node.

Figure 4-9
B+ Tree Node

Byte Offset	Number of Bytes	Item	Value
0	2	Number of bytes used in this node	
2	1	Count of leading bytes (if compressed)	
3	1	Count of trailing blanks (if compressed)	
4	k	Key (may be compressed)	} Repeats for each key entry
4+k	2	For duplicate key (if compressed)	
6+k	4	Pointer to data (top bit may duplicate flag)	
N-2	1	Index tree number (this is always the second to the last byte in the node)	
N-1	1	Level in tree (this is always the last byte in the node)	

Figure 4-10 displays the byte offsets, the lengths (in bytes) of the offsets, and the description of the meaning of the bytes for the free-list node. The last two bytes in Figure 4-10 identify the node type. In addition, you can identify the node using the position where it resides within the index tables.

Figure 4-10
Free-List Node

Byte Offset	Number of Bytes	Item	Value
0	2	Number of bytes used in this node (n)	
2	4	Index-node number for list continuation (points to next node in the list)	
6	n-8	Space for free index nodes or free data row numbers	
n-2	1	FF indicates this is a data row free list and FE indicates this is an index node free list	FF = data file FE = index file
n-1	1	End of list node flag	7F

Figure 4-11 displays the byte offsets, the lengths (in bytes) of the offsets, and the description of the meaning of the bytes for the audit-trail node. The vertically stacked dots in Figure 4-11 represent the actual log row. Each log row consists of the header, shown in the figure, and the log row whose size depends on the transaction type. In addition, only one audit trail can exist for each table.

Figure 4-11
Audit-Trail Node

Byte Offset	Number of Bytes	Item	Value
0	2	Number of bytes used in this node	
2	2	Flags	0 = audit trail is on 1 = audit trail is off
4	64	Audit trail pathname	
.	.	.	
n-1	1	End of audit trail node	7D

Symptoms and Solutions

Permission Problems	5-3
Corruption Problems	5-4
Operating-System Failures	5-4
Premature Termination of an sqlexec Process	5-5
Physical Disk Corruption	5-5
Lost and Damaged Index and Data Files	5-6
Restoring Index Files	5-7
Restoring Data Files	5-7
Transaction-Log Corruption	5-7
Disk Fragmentation	5-8
Practices to Avoid	5-8
Performance Tuning	5-9

T

his chapter provides information to help you diagnose problems that can occur when you work with a database. This chapter also covers the following topics:

- Permission
- Corruption
- Disk fragmentation
- Practices to avoid
- Performance tuning

Permission Problems

When you cannot gain access to a database or table, check the following areas:

- **The DBPATH environment variable.** This variable must be set to identify the directories that contain databases.
- **The INFORMIXSERVER environment variable.** This variable must be set to identify the correct database server.
- **UNIX permissions.** Permissions on Informix files must be set at 660 (-rw-rw----). Permissions on the database directory must be set at 770 (-rwxrwx---). You must obtain adequate UNIX permissions to access higher level directories.
- **Database ownership.** The database directory and files must belong to group **informix**.
- **SQL privileges.** Your DBA must provide adequate SQL privileges for the operation.

For information on how to set the **DBPATH** and **INFORMIXSERVER** environment variables, refer to the [Informix Guide to SQL: Reference](#). For information on how to set UNIX permissions, consult your UNIX system administrator or a guide on the UNIX operating system. For information on SQL privileges, refer to the **GRANT** and **REVOKE** statements in the [Informix Guide to SQL: Syntax](#).

Corruption Problems

Operating-system failures, premature termination of an **sqlexec** process, and disk corruption can create corruption problems.

A system crash or power failure can corrupt database data and indexes. An INFORMIX-SE database server process that terminates prematurely can also corrupt data and indexes.

You must verify database integrity after a failure occurs. Because SE cannot determine when a failure occurs, make sure your SE administrator stays alert for failures and takes appropriate corrective actions when they occur.

Operating-System Failures

Operating-system failures are generally known to the system administrators. The following events can disable an operating system:

- Power failure
- Central processing unit (CPU) failure
- System crash
- Accidental erasure of some or all operating-system files

When you experience an operating-system failure, you can reload the backup copy of your database and use the **ROLLFORWARD DATABASE** statement to apply the transaction-log file, which recovers the database.

Premature Termination of an `sqlexec` Process

Often, the only way that you can discover that an SE database server process (`sqlexec`) terminated prematurely is to examine the system accounting files. Whenever a process terminates, the accounting system stores information in a UNIX accounting file that is specified on your system. See your system administrator for the name of the accounting file. Look for processes that contain the name of a compiled user-application program or for `sqlexec` database server processes that terminated with a nonzero exit status. Remember that actual status codes can vary, depending on the operating system.

An `sqlexec` database server process can terminate prematurely due to unknown or unavoidable causes. It also can terminate prematurely when you issue a UNIX `kill` command. Do not kill database processes with the UNIX `kill -9` command. An explicit `kill -9` command prevents SE from shutting down the database server process (`sqlexec`) in a controlled manner and can compromise both physical and transaction integrity. To safely terminate a process, kill your client application.

The following sections describe various forms of corruption and what Informix recommends to fix these problems.

Physical Disk Corruption

One symptom of physical disk corruption occurs when users receive UNIX error messages about files that cannot be found. Hard-disk failure or an unintentional shutdown can create physical disk corruption.

Important: *When you use transaction logging or audit trails, you can recover the database up to the point of the last committed transaction. If you forget to establish transaction logs or audit trails, you can recover the database up to the point of the last backup.*



To repair physical disk corruption, restore the database and tables from a backup. To recover the database, you must obtain a backup copy of the database and a transaction-log file. Perform the following steps:

1. Contact your system administrator to load and restore the backup copy of the database data.
2. If your database uses transaction logging, load the transaction-log file and execute the `ROLLFORWARD DATABASE` statement. (For information on the `ROLLFORWARD DATABASE` statement, refer to the [Informix Guide to SQL: Syntax](#).)

If you do not use transaction logging and maintain audit trails on tables, load the audit-trail file and execute the `RECOVER TABLE` statement. (For information on the `RECOVER TABLE` statement, refer to the [Informix Guide to SQL: Syntax](#).)

Lost and Damaged Index and Data Files

Experiencing symptoms such as abnormally sluggish performance or missing data indicates that damaged or corrupted `.dat` or `.idx` files exist. A corrupted index file affects queries that use the index. The following situations can create that type of corruption:

- Users abnormally terminating the creation or alteration of an index
- Users aborting a batch insert

Error messages that ISAM or the UNIX operating system send also indicate a problem with the data and index files. When a corrupted file exists, users receive an ISAM error message. When you lose a file, users receive a UNIX error message indicating that the file cannot be found or cannot be opened. The following situations can generate these error messages:

- Hardware problems
- Users editing or deleting files
- Power fluctuations
- Physical problems in the mass storage system

To determine whether a corrupt table exists, you can use the CHECK TABLE statement to compare the data in a table with its indexes. You can use the REPAIR TABLE statement to repair damaged indexes or data in a table. For more information on the CHECK TABLE and REPAIR TABLE statements, refer to the [Informix Guide to SQL: Syntax](#).

Restoring Index Files

Use the **secheck** utility to check index files for damage. When damage exists, you can use **secheck** to repair or rebuild the index files. Refer to “[The secheck Utility](#)” on page 6-3. You can also check or repair tables within INFORMIX-SQL or DB-Access using specific SQL statements such as CHECK TABLE and REPAIR TABLE.

System catalog tables contain index files that can become corrupted. When you have the DBA privilege or when you are logged on as user **informix**, you can use the **secheck** utility on these index files if you suspect problems.

Restoring Data Files

To restore a data file, reload both the data file and its associated index file from your backup.

Transaction-Log Corruption

An ISAM error message stating that you cannot open the transaction log or that an unusable transaction log exists indicates transaction-log corruption problems. Possible causes for transaction-log corruption include removing the log file, improper permissions, or editing the file.

When the transaction log becomes corrupted you might want to continue using transactions.



To continue using transactions

1. Back up the database.
2. Invoke DB-Access and execute the following SQL statement to obtain the pathname of the transaction log:

```
SELECT dirpath FROM systables WHERE tabid = 0
```

3. Empty the transaction-log file by entering the following UNIX command at the operating system prompt, substituting the full pathname of the transaction-log file for *dirpath*:

```
cat /dev/null > dirpath
```

Important: Make sure that no users can access the database when you perform the preceding steps.

You can now begin logging transactions again. Use the START DATABASE statement to start a log file.

Disk Fragmentation

Disk fragmentation occurs when the blocks that make up an ISAM file become scattered throughout the partition. Disk fragmentation can slow retrieval times, making searches less efficient than when you store data in contiguous blocks. To correct disk fragmentation, unload the data and re-create the table in an unfragmented partition.

Practices to Avoid

To avoid many types of corruption, follow these few basic rules:

- Do not edit your ISAM files. Do not edit any files in the **.dbs** directory.
- Do not kill database server processes. Instead, shut down the application to make the SE database server terminate in a controlled manner.
- Do not remove or rename the transaction log while logging is active.
- Do not change the UNIX owner, group, or permissions for the **.dbs** directory or any files within that directory.

- Do not use the **dbexport** utility for database backups. Rely on proper UNIX backups.
- Do not create backups with a named pipe as the destination.
- Do not use audit trails to update your tables.

Performance Tuning

You can significantly enhance SE performance by making the following improvements:

- Cluster indexes for faster retrieval.
- Drop indexes for bulk inserts and updates that update keys.
- Save report jobs for times when all users are off-line (such as holidays, weekends, and evenings). Reports use I/O intensively.
- Update statistics during low system-load times to reduce contention.
- Use stored procedures to enhance speed. Executing a stored procedure allows you to bypass repeated parsing, validity checking, and query optimization. You can also use a stored procedure to perform frequently executed tasks.

INFORMIX-SE Utilities

The secheck Utility	6-3
Choosing Not to Specify the -n or -y Option	6-5
Simple Example	6-6
Output.	6-6
Parenthetical Values	6-7
Printing a Long List of Index Key Values	6-8
An Example Using the -l Option	6-10
Converting Index-Node Size with the -s Option	6-11
Causes for secheck Failure	6-11
The selog Utility.	6-12
Displaying the Contents of a Transaction Log	6-12
Specifying the Table Info Option	6-13
Specifying the User Info Option.	6-13
Specifying the Time-Period Info Option	6-14
Specifying the Log-Range Info Option	6-15
Specifying the Header Info Option.	6-16
Use and Output	6-20
Displaying the Contents of a Transaction Log	6-20
Displaying Activity Logged About a Specific Table	6-20
Displaying Activity Logged About a Specific User	6-20
Displaying Activity Occurring During a Time/Date Range	6-21
Displaying Activity Logged About a Specific Byte Range	6-21
Displaying Header Information.	6-21

This chapter describes the following administrative utility programs that are included with INFORMIX-SE:

- The **secheck** utility checks and restores the integrity of your index files.
- The **selog** utility displays the contents of an SE transaction log.

The SE utilities accept multibyte characters in the parameters for filenames, log file names, and table names. For information on multibyte character support for **secheck** and **selog**, see Chapter 5 of the [Guide to GLS Functionality](#). ♦

The secheck Utility

The **secheck** utility checks your index for corruption. If your index has been corrupted, **secheck** can repair it.

When you create data, you often create an index to access your data more efficiently. You can create a data file in which to store your table and data by using the SQL CREATE TABLE statement. Data files have a **.dat** extension. To create a file in which to store an index, you can use the SQL CREATE INDEX statement. Index files have a **.idx** extension. The index that you create must be associated with a data file that contains a table and its data.

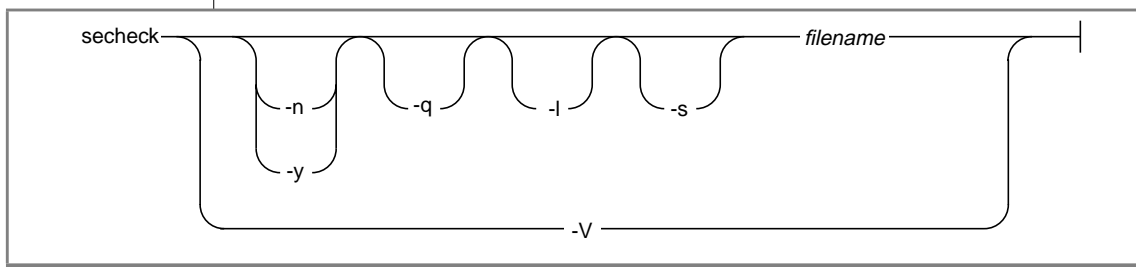
In the process of working with data files, indexes are sometimes corrupted. The **secheck** utility helps to ensure consistency between an index file and its associated data file. The **secheck** utility takes a table name as input and compares the data and index files. When **secheck** identifies corrupt indexes, it prompts you to delete the corrupt indexes and build new indexes to replace them.

You can perform the following tasks when you check and repair indexes with **secheck**:

- Print a long list of index-key values
- Convert an index file from its existing node size to the current computer hardware node size, after you migrate data

The **.dbs** directory contains the data and index files that are used to create database tables. You must run the **secheck** utility from within the **.dbs** directory.

For information about index organization and index-table structure and format, see [Chapter 4, “INFORMIX-SE Indexing”](#) in this manual.



Element	Purpose	Key Considerations
-l	Directs secheck to check and repair indexes and lists all index-key values.	References: For specific details on this option, see “An Example Using the -l Option” on page 6-10.
-n	Responds negatively to all prompts.	Additional Information: When you know in advance that all your responses to secheck prompts are negative, use the -n option. References: For specific details on this option, see “Choosing Not to Specify the -n or -y Option” on page 6-5.
-q	Suppresses banner display.	None.
-s	Converts an index file from its existing node size to the current computer hardware node size.	References: For specific details on this option, see “Converting Index-Node Size with the -s Option” on page 6-11.

(1 of 2)

Element	Purpose	Key Considerations
-V	Displays software version information.	None.
-y	Responds affirmatively to all prompts.	Additional Information: When you know in advance that all your responses to secheck prompts are positive, specify the -y option. References: For specific details about this option, see “Choosing Not to Specify the -n or -y Option” below.
<i>filename</i>	Specifies the table name associated with the data and index files that secheck evaluates.	Restrictions: The <i>filename</i> must match a value of a table listed in the database directory. References: For information about using multibyte character filenames with secheck , see the Guide to GLS Functionality .

(2 of 2)

To determine the correct value for *filename*, list the contents of the database directory. For example, the **stores7** database identifies the **customer** table in the database directory as **custo00100**, not as **customer**. System catalog tables, on the other hand, do not contain a numerical component. Do not include the **.dat** or **.idx** extension as part of the filename.

To execute the **secheck** utility, type the **secheck** syntax at the UNIX command line and then press RETURN. The following command line shows the minimum syntax that you need to successfully run **secheck**:

```
secheck filename
```

In the preceding command line, *filename* is the name of the table that is associated with the data and index files that **secheck** evaluates.

Choosing Not to Specify the -n or -y Option

When you do not know your responses to **secheck** prompts in advance, you can choose not to specify the -n or -y option. However, when you do not specify the -n or -y option, responding to prompts might take a long time.

When you do not specify the -n or -y option, **secheck** prompts you interactively. The prompts from **secheck** request confirmation that you want to re-create the index when **secheck** finds bad entries. To repair indexes, **secheck** reads all data from the **.dat** file and re-creates the index in the **.idx** file.

Simple Example

The following command checks and repairs corrupt indexes on the **customer** table:

```
secheck custo00100
```

Output

Figure 6-1 shows the output that the preceding command generates.

Figure 6-1
Example of secheck Output

```
SECHECK C-ISAM B-tree Checker version 7.20
Copyright (C) 1981-1995 Informix Software, Inc.
Software Serial Number RDS#N000000

C-ISAM File: custo00100

Checking dictionary and file sizes.
Index file node size = 1024
Current C-ISAM index file node size = 1024
Checking indexes and key descriptions.
Index 1 = unique key
    0 index node(s) used -- 1 index b-tree level(s) used
Index 2 = unique key (0,4,2)
    1 index node(s) used -- 1 index b-tree level(s) used
Index 3 = duplicates (111,5,0)
    1 index node(s) used -- 1 index b-tree level(s) used
Checking data record and index node free lists.
4 index node(s) used, 0 free -- 28 data record(s) used, 4 free
```



Tip: The first index listing in the **secheck** output displays the height of the B+ tree and the number of index nodes used.

Parenthetical Values

In the **secheck** output, one set of parenthetical values (values enclosed within parentheses) appears for each column that is named as part of the index. The following example from Figure 6-1 shows a line of output that contains parenthetical values:

```
Index 2 = unique key (0,4,2)
```

The three enclosed values from the preceding example define the following characteristics for each column component of the index:

1. The starting byte value of this component (within the row of the table upon which you are building the index) is the first value in the parentheses.
2. The length of the value in bytes is the second value in the parentheses.
3. The data type of this component column, expressed as a digit, is the third value in the parentheses. Figure 6-2 shows **secheck** data types and related digit values.

Figure 6-2
secheck Data Types and Related Digit Values

Data Type	Digit
CHAR	0
SMALLINT	1
INTEGER	2
FLOAT	3
SMALLFLOAT	4
DECIMAL	5
SERIAL	6
DATE	7
MONEY	8

(1 of 2)

Data Type	Digit
DATETIME	10
INTERVAL	14
NCHAR	15

(2 of 2)

Printing a Long List of Index Key Values

Figure 6-3 describes the content of each node.

Figure 6-3
Descriptions of Node Contents

Node Content	Description
flag	Tells you whether the key is a duplicate: 0 (unique) 1 (duplicate) 2 (following key duplicates current key) 3 (duplicate and the following key duplicates current key)
totln	Is the total byte length of the key: keyln + recptr (4 bytes) + leadc (1 byte) + tailc (1 byte) + when duplicate (2 bytes)
keyln	Is the field length - (leadc + tailc), where field length equals the total length of the columns that make up the index.
dupnm	Is the duplicate counter. Represents the number of duplicates that secheck encountered. 0 represents the first occurrence.

(1 of 2)

Node Content	Description
recptr	Is the rowid of the row in the table when the node is a leaf node. When the node is a root node or a branch node, the recptr points to the next index node that contains records less than or equal to the key value of this entry. When this is the last entry of the root node or the right-most branch node, the recptr points to the next index node that contains records greater than the key value of this entry.
leadc	Is the number of bytes saved due to leading-character compression.
tailc	Is the number of bytes saved due to trailing-character compression. SE creates indexes with full compression applied when the total length of the CHAR type key exceeds seven.
key	Is the actual key value.

(2 of 2)

An Example Using the -I Option

The following command checks and repairs corrupt indexes and lists all index key values on the **customer** table:

```
secheck -I custo00100
```

Figure 6-4 shows the output that is generated when you use the -I option with the **secheck** command.

Figure 6-4
Example of secheck Output for List of Key Values

```
SECHECK C-ISAM B-tree Checker version 7.20
Copyright (C) 1981-1995 Informix Software, Inc.
Software Serial Number RDS#N000000

C-ISAM File: custo00100

Checking dictionary and file sizes.
Index file node size = 1024
Current C-ISAM index file node size = 1024
Checking data file records.
Checking indexes and key descriptions.
Index 1 = unique key
      0 index node(s) used -- 1 index b-tree level(s) used
Index 2 = unique key (0,4,2)

btree level: 0, node: 3, used: 144
flag totln keyln dupnm recptr leadc tailc key
0      8      4      0      1      0      0 101
0      8      4      0      2      0      0 102
0      8      4      0      3      0      0 103
0      8      4      0      4      0      0 104
0      8      4      0      5      0      0 105
0      8      4      0      6      0      0 106
.
.
.
1 index node(s) used -- 1 index b-tree level(s) used
Index 3 = duplicates (111,5,0)
.
.
.
```

Notice the following portion of the output in Figure 6-4:

```
btree level: 0, node: 3, used: 144
flag totln keyln dupnm recptr leadc tailc key
```

The rightmost column heading, **key**, contains very useful data. The complete index-key value for each data row in the table appears under the **key** heading. When the index has more than one piece (a composite index can contain up to eight pieces), the value of each piece is displayed, separated by a space. (Informix designates all other columns for internal use and does not document them.)

Converting Index-Node Size with the -s Option

The index-node size is a number that is a multiple of 512 bytes and is fixed for each computer. The node size affects how many keys you can store in a node before it must be split.

An incorrect index-node size can occur for the following reasons:

- You moved a database from one computer to another.
- You upgraded to a newer version of SE.

Use **secheck** with the **-s** option after you move a table to a computer with a different node size or upgrade to a newer version of SE. When you are running an application on an incompatible computer, an error message indicating a wrong node size appears.

Causes for secheck Failure

The **secheck** utility fails when it:

- encounters an invalid filename.
- cannot resize the index file.
- cannot allocate a new file descriptor.
- cannot lock the index file.
- encounters an unstable lock for the index file.
- cannot read the dictionary.
- cannot check the consistency of the dictionary.
- encounters the wrong GLS collation sequence.
- cannot allocate a temporary record.

The selog Utility

The **selog** utility displays the contents of an SE transaction-log file. A transaction-log file keeps an automatic record of activity associated with a specific database. For information on transaction-log files, refer to “[Transaction-Log Files](#)” on page 2-6. In addition, refer to the *Informix Guide to SQL: Syntax* for information about creating SE databases with transaction logging.

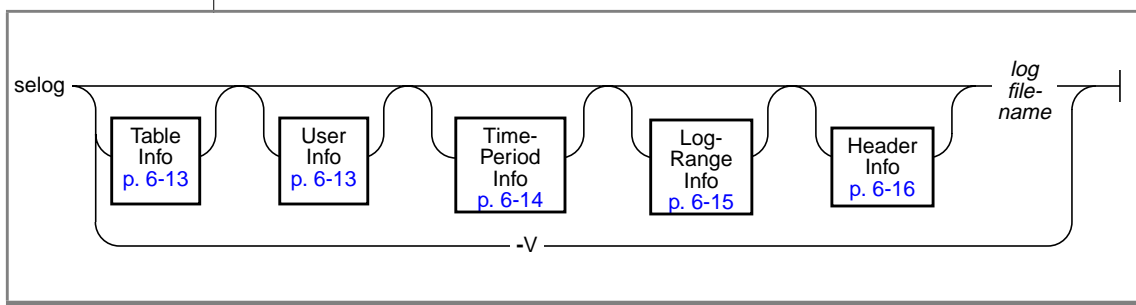
For information on multibyte character support for SE utilities, see Chapter 5 of the *Guide to GLS Functionality*. ♦

Displaying the Contents of a Transaction Log

You can display the complete contents of the transaction-log file, or you can specify transaction records based on the following criteria:

- Activity on a specific table
- Activity that a specific user initiates
- Activity within a specific time period
- Activity within a specific log range

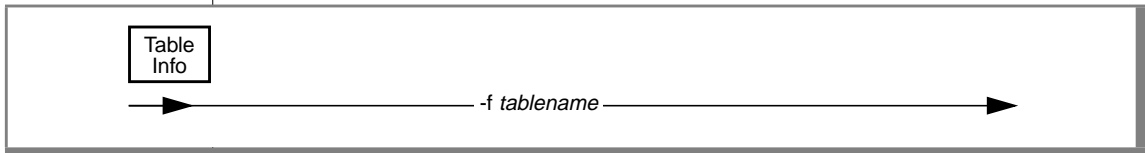
Other options allow you to choose the type and frequency of the header display.



Element	Purpose	Key Considerations
-V	Displays software version information.	None.
<i>log filename</i>	Specifies the name of the transaction-log file.	Restrictions: You can specify the <i>log filename</i> as a file in the current directory or as a complete pathname.

Specifying the Table Info Option

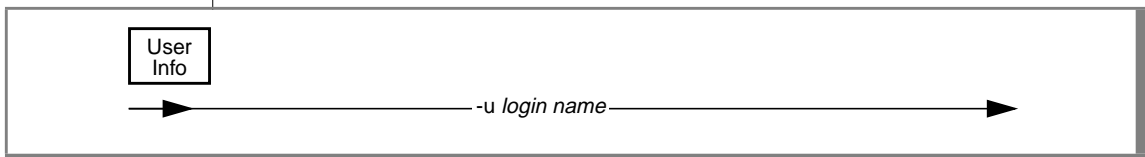
The Table Info option directs **selog** to display all activity on a specified table.



Element	Purpose	Key Considerations
-f tablename	Directs selog to display all activity on the specified table, where <i>tablename</i> represents the name of the table as it appears in the systables system catalog table.	Restrictions: Table must exist when you execute the utility. To obtain the correct value for <i>tablename</i> , invoke DB-Access and run the following SQL statement to list the directory paths and the names associated with all the tables in your database: <pre>SELECT dirpath, tablename FROM systables</pre>

Specifying the User Info Option

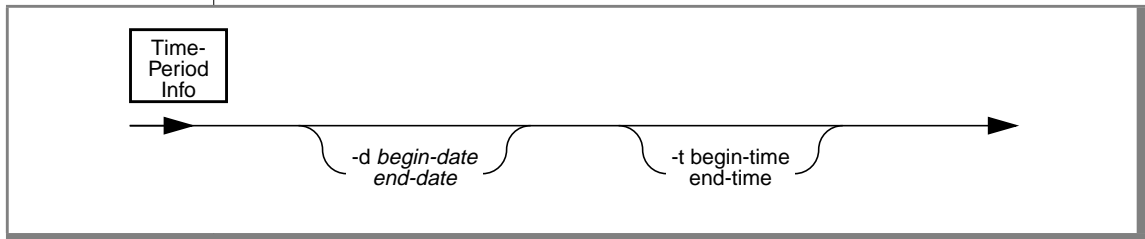
The User Info option directs **selog** to display all activity that is associated with a specific user.



Element	Purpose	Key Considerations
-u <i>login name</i>	Directs selog to display all activity that the specified login name initiates.	Restrictions: Must be an existing login name. The login name must conform to operating-system-specific rules for login name.

Specifying the Time-Period Info Option

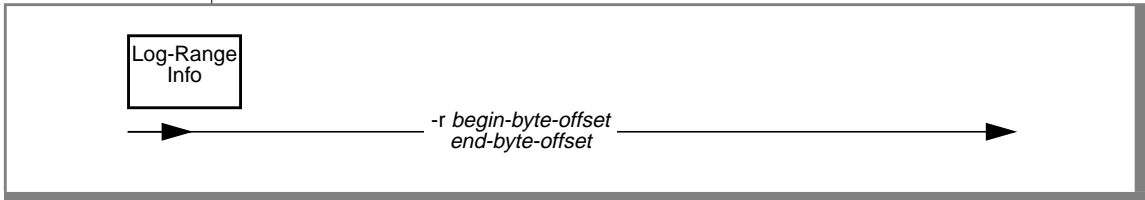
The Time-Period Info option directs **selog** to within a certain date range and time range that you specify.



Element	Purpose	Key Considerations
-d <i>begin-date end-date</i>	Directs selog to display all activity that occurred within the specified date range.	Restrictions: When you specify a <i>begin-date</i> without an <i>end-date</i> , an error occurs. Specify dates in <i>mm/dd/yyyy</i> format. Make sure that you include a space between the two date values.
-t <i>begin-time end-time</i>	Directs selog to display all activity that occurred within the specified time range.	Restrictions: When you specify a time without a date, selog assumes the current day. When you specify a <i>begin-time</i> without an <i>end-time</i> , an error results. Make sure that you include a space between the two time values. Specify time in <i>hh:mm:ss</i> format. Make sure that you include a space between the two time values.

Specifying the Log-Range Info Option

The Log-Range Info option directs **selog** to display all transaction records that are located in the log within a byte range that you specify.



Element	Purpose	Key Considerations
-r <i>begin-byte-offset</i> <i>end-byte-offset</i>	Directs selog to display all transaction records located in the log within the specified byte range, inclusive.	Restrictions: Make sure that you include a space between the two offset values.

To obtain the byte offset for a specific record, use the **-l** option described in “[Specifying the Header Info Option](#)” (You do not need any other parameter.) See [Figure 6-8 on page 6-22](#) for an example of output that the **-l** option generates.

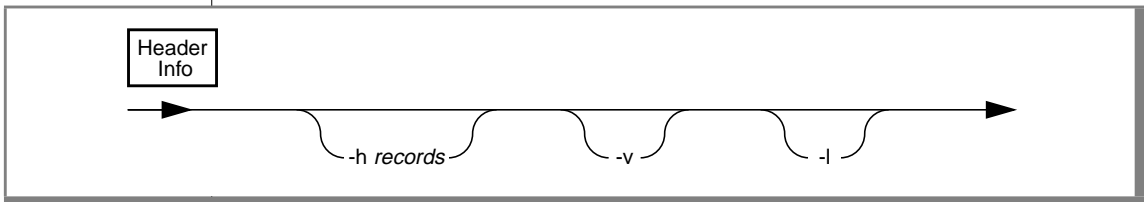
Specifying the Header Info Option

The Header Info option displays header information. A header is a title row that precedes data rows. Typically, header columns identify the columns of the rows of data that follow them.

When you specify no Header Info options, the default display for each transaction-log record begins with a header that contains the information that is listed in Figure 6-5. For corresponding information on the transaction-log record header, see [“Transaction-Log Files” on page 2-6](#).

Figure 6-5
Header Information for Default Display

Header Column	Contents
TY	Record type code
TrxID	Transaction number
User	User id associated with the record
User Name	User name associated with the user id
Date/Time	Time the record was written to the log
Lngth	Record length in bytes
FD	Descriptor of affected ISAM file
Recno	Record number affected by the log record
Filename	Filename contained in the log record (for record types open, close, build, erase, and rename)



Element	Purpose	Key Considerations
-h <i>records</i>	Specifies the number of transaction records separating each header display.	Restrictions: An integer. When you omit the -h option, the default value for <i>records</i> is 20. When you set <i>records</i> to 0, the header prints only once at the beginning of the output.
-l	Displays additional location information, including the byte offset for each transaction record.	Additional Information: When you use the -l option, the selog output displays three additional fields. The first, Location , displays the byte offset for each transaction record. The other two fields, Prev Loc and Prev Leng , hold additional internal information provided for updated records. The information is used internally during ROLLBACK WORK procedure.
-v	Displays additional fields from the transaction records, depending on the transaction record type.	Restrictions: You must enter the -v option as a lowercase character. In this context, when you enter -v as an uppercase character, you generate an error message. Do not confuse the lowercase -v option with the uppercase -V option, which displays version information. References: For specific details on this option, see “Displaying Additional Fields with the -v Option” below.

Displaying Additional Fields with the -v Option

When you specify the **-v** option, **selog** displays additional fields for certain record types. Figure 6-6 shows all record types and the additional fields that the **-v** option displays for each.

Figure 6-6
Additional Header Fields Displayed by the -v Option

Record Type	Code	Additional Fields Displayed
Build an ISAM file	BU	Row length (ISAM record length); build mode
Close an ISAM file	FC	None
Open an ISAM file	FO	None

(1 of 3)

Displaying the Contents of a Transaction Log

Record Type	Code	Additional Fields Displayed
Delete a record from an ISAM file	DE	Image (hexadecimal and ASCII display of the log record)
Insert a record from an ISAM file	IN	Image (hexadecimal and ASCII display of the log record)
Rename an ISAM file	RE	New filename (new filename contained in the log record)
Update a record in an ISAM file	UP	Pre-image (hexadecimal and ASCII display of the ISAM file before-image contained in the log record) Post-image (hexadecimal and ASCII display of the after-image contained in the log record)
Get a unique ID for a particular ISAM file	UN	None
Get a unique ID for a particular ISAM file	UN	None
Set a unique ID for a particular ISAM file	SU	None
Erase an ISAM file	ER	None
Begin work	BW	None
Commit work	CW	None
Rollback work	RW	None

(2 of 3)

Record Type	Code	Additional Fields Displayed
Create an index on an ISAM file	CI	Key-description information
Delete an index on an ISAM file	DI	Key-description information
Change the physical order of an ISAM file to key sequence	CL	Key-description information

(3 of 3)

Three record types that are associated with the indexes (CI, DI, and DL) display detailed key information. Key information is derived from a key structure. The key structure describes an index. The following example shows the format and contents of the important elements of the key structure (or key display):

```
key.k_nparts = n key.k_flags = n key.k_len = n
key.k_part:  kp_start kp_leng kp_type
```

<code>key.k_nparts</code>	specifies the number of component parts in the index key description contained in the log record.
<code>key.k_flags</code>	specifies the key flags associated with the key description.
<code>key.k_len</code>	specifies the length of the log record key description.
<code>key.k_part</code>	describes the following items for each component of the index key:
<code>kp_start</code>	starting byte of this part of the key
<code>kp_leng</code>	length of this part of the key
<code>kp_type</code>	type associated with this part of the key

For details about the parts that are used to make an index key, refer to the discussion on [“Printing a Long List of Index Key Values”](#) on page 6-8.

Use and Output

This section shows you how to use **selog** syntax and how to interpret some examples of **selog** output. To execute the **selog** utility, type the **selog** syntax at the UNIX command line and then press RETURN.

Displaying the Contents of a Transaction Log

The following command displays all records in the **allcall.log** transaction log using the default format:

```
selog allcall.log
```

Figure 6-7 shows the output generated by the preceding command. Refer to [Figure 6-5 on page 6-16](#) for definitions of each header field.

Figure 6-7

selog Output for Displaying All Records in a Transaction Log

```
SELOG: Transaction Log File Display C-ISAM Version 7.20
Copyright (C) 1981-1995 Informix Software, Inc.
Software Serial Number INF#R000000
```

TY	Trx ID	User	User Name	Date/Time	Lngh	FD	Recno	Filename
BW	23614	1817	stevek	12/ 5 17:29:20	20			
FO	23614	1817	stevek	12/ 5 17:29:20	74	1		custo00100
IN	23614	1817	stevek	12/ 5 17:29:20	160	1	17	
IN	23614	1817	stevek	12/ 5 17:29:20	160	1	16	
IN	23614	1817	stevek	12/ 5 17:29:20	160	1	15	

Displaying Activity Logged About a Specific Table

The following command displays all of the activity logged about the **customer** table:

```
selog -f custo00100 allcall.log
```

Displaying Activity Logged About a Specific User

The following command displays all the activity logged about the user **billj**:

```
selog -u billj allcall.log
```

Displaying Activity Occurring During a Time/Date Range

The following command displays all transactions that were logged for the time range of 10:00:00 to 11:30:00 in the date range from 12/12/94 to 12/15/94:

```
selog -d 12/12/94 12/15/94 -t 10:00:00 11:30:00 allcall.log
```

Displaying Activity Logged About a Specific Byte Range

The following command displays all transaction records beginning at byte offset 100 and ending at byte offset 500, inclusive:

```
selog -r 100 500 allcall.log
```

See “Displaying Header Information” below for information on obtaining the byte offset for a specific record.

Displaying Header Information

The following command displays location and byte offset information for headers associated with transaction records:

```
selog -l allcall.log
```

Figure 6-8 shows output generated when you use the -I option.

Figure 6-8
Example of Output When Using the -I Option

```
SELOG: Transaction Log File Display C-ISAM Version 7.20
Copyright (C) 1981-1995 Informix Software, Inc.
Software Serial Number RDS#N000000
```

TY	Trx ID	User	User Name	Date/Time	Lngh	FD	Recno	Filename	Location	Prev	Loc	Pr	Ln
BW	9647	9658	stevek	1/14 15:16:27	20				0		0		0
FO	9647	9658	stevek	1/14 15:16:27	43	0		sysstables	20		0		0
UN	9647	9658	stevek	1/14 15:16:27	26	0	100		63		0		0
BU	9647	9658	stevek	1/14 15:16:27	52			junk.dbs/tab1_00100	89		0		0
FO	9647	9658	stevek	1/14 15:16:27	43	3		sysstables	141		0		0
IN	9647	9658	stevek	1/14 15:16:27	205	3	25		184		0		0
FO	9647	9658	stevek	1/14 15:16:27	44	4		syscolumns	389		0		0
.													
.													
.													
Program over.													

The area under the **Location** header of the -I option display provides the byte offset for each transaction record output. When you attempt to estimate a location, enter a decimal value, not a hexadecimal value.

Index

A

ALTER TABLE statement 3-3
 ANSI standard SQL 1-3
 ANSI-compliant logging 3-4
 Audit-trail file
 compared to transaction-log file 2-8
 considering growth of 1-5
 creating 3-6
 description of 2-8, 3-6
 format for a table with variable-length rows 2-9
 priority in disk placement 1-4

B

Backing up the transaction-log file 3-5
 Backups, creating 3-6
 Block, definition of 4-12
 B+ tree
 adding to 4-6
 deleting from 4-11
 growth of 4-8
 levels 4-5
 maximum keys per node 4-5
 nodes 4-3
 organization 4-3
 pointers 4-3
 searching 4-5
 sequential addition to 4-8

C

Checking and repairing corrupt indexes 6-3
 Client/server configurations
 compatibility with local SE server 1-7
 compatibility with remote SE server 1-8
 local, definition of 1-7
 remote, definition of 1-7
 Communication files
 discussed 1-10
 /etc/hosts 1-9
 /etc/services 1-9
 for IPX/SPX connections 1-10
 for network security 1-9
 for TCP/IP connections 1-9
 \$INFORMIDIR/etc/sqlhosts 1-9
 network 1-10
 network security 1-10
 overview 1-9
 relationships between 1-17
 sqlhosts
 description of 1-12
 example of fields 1-13
 file 1-9
 format of nettype field 1-13
 hostname field 1-15
 nettype field 1-13
 servicename field 1-16
 Connecting to a database server, example of 1-38
 Connection
 4.11 client with 7.2 server 1-26
 5.x client with 7.2 server 1-26

- 7.2 client with 7.2 server, example 1-20
- 7.2 Relay Module 1-32
- client/server configurations 1-7
- local loopback, example 1-23
- local, with unnamed pipes 1-19
- network 1-32
- network, description of 1-13
- preparing 1-6
- using INFORMIX-NET 1-28
- using INFORMIX-NET Relay Module 5.x 1-29
- with different client/server versions 1-26

Connectivity file, sqlhosts 1-12

Conventions

- command-line syntax Intro-11
- example code Intro-14
- icon Intro-10
- overview Intro-8
- railroad diagram Intro-11
- typographical Intro-9

Corruption

- avoiding 5-8
- data and indexes 5-4
- indexes 6-3
- lost index and data files 5-6
- physical disk 5-5
- transaction log 5-7

CREATE AUDIT statement 3-6

CREATE DATABASE statement 3-4

Creating backups 3-6

D

- .dat file 1-5, 2-5

Data

- intermediate transfer method 1-4
- transferring 1-4

Data file 2-5

Data language, description of 1-3

Data row storage 2-6

Database management system (DBMS) 1-3

Database server, description of 1-3

DBPATH environment variable 1-8, 5-3

Default UNIX permissions 2-10

Demonstration database

- copying Intro-6
- installation script Intro-5
- overview Intro-5

Differences between audit trails and transaction logs 2-8

Disk

- data and head movement 1-5
- fragmentation 5-8
- minimizing input/output 1-4
- partitioning 1-4
- reducing competition for access 1-4
- using separate devices 1-4

Displaying contents of a transaction log 6-12

Documentation notes Intro-17

Documentation, other useful Intro-15

E

Environment variables

- DBPATH 1-8, 5-3
- GLS-related 1-9
- INFORMIXDIR 1-8
- INFORMIXSERVER 1-8, 5-3
- INFORMIXSQLHOSTS 1-8, 1-12
- overview 1-8
- PATH 1-9
- setting 1-8
- SQLEXEC 1-9
- SQLRM 1-9
- SQLRMDIR 1-9
- TERM 1-9
- TERMCAP 1-9
- TERMINFO 1-9

Error messages Intro-16

- /etc/hosts file
 - example of hostname field 1-15
 - host-alias field 1-15
 - hostname field 1-15
 - net address field 1-15
- /etc/services file
 - port number/protocol field 1-16
 - service-alias field 1-16
 - servicename field 1-16

G

Global Language Support (GLS)

- description of Intro-8
- environment variables 1-9
- using with SE 1-9

Group informix 2-11

H

Hardware configuration 1-4

hostname field 1-15

I

.idx file 2-6, 1-5

Index

- checking 6-3
- dictionary node 4-12
- file 2-5, 2-6
- fixing when corrupt 6-3
- free-list node 4-13
- ISAM table 4-3
- key description node 4-13
- organization 4-3
- repairing 6-3
- table formats 4-15
- table organization 4-12
- table structure 4-12
- using secheck utility on 6-3

INFORMIXDIR environment variable 1-8

INFORMIX-SE

- compatibility with client tools 1-7
- considerations before installing 1-4
- creating the demonstration database Intro-5
- improving performance 1-5
- installing 1-6
- limits 1-5
- Machine notes file 1-14
- maximum for open tables 1-5
- maximum locks per table 1-5
- maximum row size 1-5
- starting a process 1-20
- system files 2-4

INFORMIXSERVER environment variable 1-8, 5-3
INFORMIXSQLHOSTS environment variable 1-8, 1-12
INFORMIXTERM environment variable 1-8, 1-9
Inodes, increasing number of 1-5
Installation
 definition of 1-6
 instructions for 1-6
Interprocess communication (IPC), enabling with sqlhosts file 1-14
IPX/SPX protocol, enabled with sqlhosts file 1-14
ISAM table 4-3

K

Kernel parameters, increasing value of 1-5

L

Limits
 INFORMIX-SE Intro-17, 1-5
 number of locks per table 1-5
 number of open tables 1-5
 row size 1-5
 UNIX 1-5
Lost index and data files 5-6

M

Machine notes file 1-14
Maximum limits
 number of locks per table 1-5
 number of open tables 1-5
 row size 1-5
Message files, error messages Intro-16
Monitoring
 data integrity 3-4
 disk space usage 3-3

N

nettype field
 examples of 1-15
 valid values for 1-15
Network File System (NFS), databases residing on 1-37
Network interface protocol, description of 1-14
Network security files and communication files 1-9
 overview 1-10
 /etc/host.equiv 1-10
 /etc/passwd 1-10
 /etc/shadow 1-10
 /.netrc 1-10
 /.rhosts 1-10
Notes
 Documentation Intro-17
 Machine Intro-17, 1-14
 Release Intro-17

O

On-line files Intro-16
Operating system
 block size 2-6
 failure 5-4
 permissions 2-4, 2-11
Organization of C-ISAM files, index file 4-12
Ownership and group settings 2-11

P

Partition
 avoiding overflow 1-5
 for disks 1-4
 increasing size of 1-5
 moving table to larger 1-5
 spanning 1-5
 using innermost for fast access 1-5
PATH environment variable 1-8, 1-9
Performance
 improving 1-5
 multiple indexes 4-13
 tuning 5-9

Permissions
 database files 2-10
 directories 2-10
 problems 5-3
Physical disk corruption 5-5
Placement of active tables and files 1-4
Planning for growth 1-4
Printing index key values 6-8
Program files 2-3
Protecting data integrity 3-4
Protocols, network, enabled with sqlhosts file 1-14

R

Railroad diagrams
 conventions used in Intro-11
 example of syntax conventions Intro-13
RECOVER TABLE statement 5-6
Relay module
 using version 5.x 1-29
 using version 7.2 1-32
Release notes Intro-17
Renaming the transaction-log file 3-5
Repairing and checking corrupt indexes 6-3
Restoring
 data files 5-7
 index files 5-7
Restricting access 2-4
ROLLFORWARD DATABASE statement 5-6

S

secheck utility
 converting index node size 6-11
 description of 6-3
 printing index key values 6-8
Security files, for network 1-9
selog utility
 additional header fields 6-17
 default header information 6-16
 description of 6-12

- displaying contents of a
 - transaction log 6-12
- key display 6-19
- options
 - specifying a header 6-16
 - specifying activity initiated by user 6-13
 - specifying activity on a table 6-13
 - specifying activity within a log range 6-14
- servicename field
 - description of 1-16
 - IPX/SPX connection 1-22
 - TCP/IP connection 1-16
- Setting environment variables 1-8
- Sockets, in nettype field 1-14
- Space allocation
 - for transaction-log files 2-8
 - for .dat and .idx files 2-6
- SQL API 1-3
- SQL privileges 5-3
- SQL statements
 - ALTER TABLE 3-3
 - CREATE AUDIT 3-6
 - CREATE DATABASE 3-4
 - RECOVER TABLE 5-6
 - ROLLFORWARD DATABASE 5-6
 - START DATABASE 3-4
- SQLEXEC environment variable 1-9
- sqlxec process 5-5
- sqlxecd daemon
 - logfile options 1-37
 - setting 1-35
- sqlhosts file
 - building 1-12
 - dbservername field 1-13
 - dbservername field illustrated 1-13
 - editing 1-12
 - enabling IPX/SPX protocol 1-14
 - enabling TCP/IP protocol 1-14
 - enabling unnamed pipes 1-14
 - entries for 1-13
 - entries to avoid 1-12
 - examples of nettype fields 1-15
 - field for database servename 1-12

- field for hostname 1-12
- field for network protocol 1-12
- field for servicename 1-12
- hostname field 1-15
- hostname field illustrated 1-13
- nettype entry
 - for INFORMIX-Gateway with DRDA 1-14
 - for INFORMIX-SE 1-14
 - for interprocess communication 1-14
 - for OnLine 1-14
 - for socket network interface protocol 1-14
 - for transport-level interface 1-14
- nettype field format 1-13
- nettype field illustrated 1-13
- nettype field internal programming interface connection entry 1-14
- nettype field subfields 1-14
- overview 1-12
- servicename field 1-16
- servicename field illustrated 1-13
- table of fields 1-13
- table of valid nettype values 1-15
- SQLRM environment variable 1-9
- SQLRMDIR environment variable 1-9
- START DATABASE statement 3-4
- stores7 database
 - copying Intro-6
 - creating Intro-5
 - overview Intro-5
- Stray locks 5-7
- System catalog tables 5-7
- System files 2-4

T

- TCP/IP protocol, enabled with sqlhosts file 1-14
- TERM environment variable 1-9
- TERMCAP environment variable 1-9
- termcap file 1-9
- terminfo directory 1-9

- TERMINFO environment variable 1-9
- Transaction log
 - considering growth of 1-5
 - priority in disk placement 1-4
- Transaction record header format 2-7
- Transaction-log file
 - compared to audit-trail file 2-8
 - corruption 5-7
 - creating 3-4
 - description of 2-6, 3-4
 - displaying contents of 6-12
 - header information for 6-16
 - log range information for 6-15
 - maintaining 3-5
 - table information for 6-13
 - time-period information for 6-14
 - user information for 6-13
 - using selog utility on 6-12
- Transaction, definition of 3-4
- Transferring data 1-4
- Transport-level interface, enabling with sqlhosts file 1-14

U

- Unbuffered logging 3-4
- UNIX
 - increasing size of kernel parameters 1-5
 - kernel parameters 1-5
 - limitations 1-5
 - permissions 5-3
- Unnamed pipes, enabled with sqlhosts file 1-14
- /usr/informix directory 1-6
- Utilities
 - secheck 6-3
 - selog 6-12