ULTRIX

ULTRIX/SQL Operations Guide

Order Number: AA–PBZ9A–TE

ULTRIX

ULTRIX/SQL Operations Guide

Order Number: AA-PBZ9A-TE
June 1990

Software Version:	ULTRIX/SQL Version 1.0
Operating System and Version:	ULTRIX Version 4.0 or higher

This guide discusses ULTRIX/SQL system management. Version 1.0 of ULTRIX/SQL is based on INGRES Release 6.2.

digital equipment corporation
maynard, massachusetts

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital or its affiliated companies.

Table of Contents

Preface

1 Introduction

2 Overview of Installation and Operations Utilities

3 Configuration Decisions

4 Initializing ULTRIX/SQL and Authorizing User Access

5 Maintenance Utilities

6 Installation Tuning Reference Material

7 Troubleshooting

A ULTRIX/SQL Startup Files

B Authorizing User Access to ULTRIX/SQL and Databases

C ULTRIX/SQL Environment Variables

D ULTRIX/SQL System Recovery

E Commands for ULTRIX/SQL System Administration

F Running ULTRIX/SQL Under Network File System

Index

Preface

Purpose of this Document

The *ULTRIX/SQL Operations Guide* describes the initialization and operations procedures for ULTRIX/SQL.

Intended Audience

This manual is intended for the person with overall responsibility for the operation of ULTRIX/SQL running under the ULTRIX operating system in the VAX or RISC environment. The reader should have a thorough understanding of the ULTRIX operating system.

Organization of this Document

The *ULTRIX/SQL Operations Guide* is divided into the following parts:

- Chapter 1 provides an introduction to this guide.
- Chapter 2 describes the initialization utility **iibuild**, the startup utility **iistartup** and the shutdown utility **iishutdown**.
- Chapter 3 discusses various system requirements and configuration decisions.
- Chapter 4 describes system initialization (**iibuild**).
- Chapter 5 discusses the maintenance utilities including:
  - Server monitoring and control utility (**iimonitor**)
  - Name server maintenance utility (**iinamu**)
  - Shared memory and semaphores report utility (**csreport**)
  - Locking facility report (**lockstat**)
  - Logging facility report (**logstat**)

- Chapter 6 discusses miscellaneous installation topics including the database management system server startup utility and the logging and locking facility.
- Chapter 7 discusses troubleshooting of ULTRIX/SQL.
- Appendix A discusses ULTRIX/SQL startup files.
- Appendix B discusses authorizing user access to ULTRIX/SQL and databases.
- Appendix C discusses ULTRIX/SQL environment variables.
- Appendix D discusses recovery of the ULTRIX/SQL master database.
- Appendix E describes the syntax of various ULTRIX/SQL Operating System commands.
- Appendix F discusses running ULTRIX/SQL under the Network File System (NFS).

Compatibility with Remote Access to Rdb/VMS

This document assumes that your installation does not include Remote Access to Rdb/VMS. If your installation includes this option, be sure to check your documentation for Remote Access to Rdb/VMS for information about syntax that may differ from that described in this manual. Remote Access to Rdb/VMS is a VMS layered product installed on a VMS system running Rdb/VMS, which is connected to your ULTRIX/SQL system(s).

Areas that may differ include:

- Length of **varchar** data type
- Legal row size
- Command usage
- Name length
- Table size

Associated Documents

The following manuals are included in your ULTRIX/SQL base system documentation set:

ULTRIX/SQL Database Administrator's Guide
ULTRIX/SQL NET User's Guide
ULTRIX/SQL Operations Guide
ULTRIX/SQL Reference Manual
ULTRIX/SQL Release Notes

Conventions

The following conventions are used to describe syntax in this manual:

- **Boldface** type is used to identify reserved words and required symbols and punctuation in syntax that must be typed as shown when used. Boldface is also used to indicate data types and key names. In sample terminal output, boldface is used to emphasize sections that require further explanation.

- Words in *italics* within text and syntax diagrams represent variable elements of syntax that are to be supplied by the program or the user. Italics are also used within text to introduce new terminology or to show emphasis.

- Double quotes (“ ”) within the general text indicate a specific value of a parameter. Double quotes (" ") and single quotes (' ') within syntax and in code examples have specific meanings within the context of SQL or a host programming language.

- Reserved words are shown in boldface, lowercase letters (except in host language examples, where embedded SQL statements appear in uppercase to distinguish them from the host language code). Although ULTRIX/SQL does not actually distinguish between uppercase and lowercase in reserved words, it does convert any uppercase letters to lowercase. This is true only for reserved words. Variables are case sensitive.

- This documentation uses generic keyboard key names. The key names on your particular keyboard may vary slightly from those used in this documentation. Key names joined by a hyphen (such as **Control-P**) indicate that the user is to press the named keys simultaneously.

- Syntax diagrams may continue over several lines. Line wraps and additional lines in statement and command line syntax are indented under the first line of the statement or command.

- Clauses or arguments enclosed in square brackets ([ ]) within syntax diagrams are optional.

- Clauses enclosed in braces ({ }) within syntax diagrams are optional and can be repeated zero or more times.

- Clauses or reserved words separated by vertical bars (|) within syntax diagrams indicate lists from which one element is to be chosen.

- Examples of code are separated from the text and are shown in a special, constant-width typeface.

- *Pseudocode*, a description of an operation without the actual code, is shown in italics within examples. This generic program code is used to clarify overall syntax structure without unnecessary detail.

References to Products

The ULTRIX/SQL documentation to which this manual belongs often refers to products by their abbreviated names:

- ULTRIX/SQL refers to ULTRIX/SQL database software and to its implementation of the SQL language. (Repetitive occurrences of ULTRIX/SQL have been shortened to SQL.)

- Rdb/VMS refers to VAX Rdb/VMS database software.

Introduction 1

1.1 Overview

ULTRIX/SQL is a relational database management and application development system for the ULTRIX Version 4 operating system running on either a Digital VAX and RISC computer. This manual describes the procedures to maintain and modify the ULTRIX/SQL environment and to authorize user access to specific databases.

This manual also provides information for the ULTRIX/SQL System Administrator. The ULTRIX/SQL System Administrator is responsible for the ULTRIX/SQL installation and the authorization of user access to ULTRIX/SQL. The ULTRIX/SQL System Administrator does not necessarily have responsibility for databases maintained under the system.

A Database Administrator (DBA) is designated for each database. The DBA is usually the user who creates a database. The DBA is responsible for the:

- Creation of shared tables
- Authorization of user access to shared tables
- Backup of the database
- Recovery of the database

Note

This publication assumes that anyone initializing and maintaining ULTRIX/SQL on a VAX or RISC system has a thorough knowledge of the ULTRIX operating system.

If you are in possession of the root password and do not have the minimal knowledge of ULTRIX that this guide presupposes, please be sure that someone with ULTRIX expertise assists you. In the event of an unanticipated error, this will help you to interpret ULTRIX error messages.

We *strongly* recommend that you read the *ULTRIX/SQL Installation Guide*, *ULTRIX/SQL Release Notes*, and this entire manual before using any of the ULTRIX/SQL operations and maintenance tools.

1.2 Introduction to ULTRIX/SQL Architecture

Before beginning the installation procedure, you should become familiar with the following three elements of the ULTRIX/SQL architecture:

- The Logging and Locking System
- The General Communication Facility
- The Database Management System Server

This chapter describes each of these elements.

1.3 The Logging and Locking System

The logging and locking system manages database access. It coordinates the locking, recovery, and journaling of databases. The system is composed of the following components:

- Shared memory locking
- Transaction log file
- Recovery process
- Archiver process

1.3.1 Shared Memory Locking

When you initially configured your ULTRIX kernel, you installed the shared memory and semaphore resources used by the ULTRIX/SQL lock manager. Shared memory is allocated to ULTRIX/SQL components initially during the ULTRIX/SQL initialization procedure. The amount of shared memory that your installation requires depends on the logging and database management system server parameters that you select during the initialization procedure. You may reconfigure the parameters during any subsequent installation startup.

1.3.2 Transaction Log File

The ULTRIX/SQL transaction log file $II_LOG_FILE/ingres/log/ingres_log holds the information about pending transactions and is used to recover active databases after a system failure. It is a circular file of fixed size. Initially, you configured the size of the file during the installation procedure, but you can reconfigure it during any subsequent installation startup. You also have the option of creating your log file as a raw device.

1.3.3 The Recovery Process

Your ULTRIX/SQL installation has one recovery process called the **dmfrcp** (data manipulation facility recovery process). It is a daemon process that automatically uses data from the log file to recover inconsistent databases caused by system crashes or other problems. The recovery process backs out uncommitted transactions and makes the database consistent again without affecting users in the database.

1.3.4 The Archiver Process

Your ULTRIX/SQL installation has one archiver process called the **dmfacp** (data manipulation facility archiver process). It archives transactions for databases and tables that have journaling enabled. It is a daemon process that follows behind the database management system servers in the log file. As transactions are committed the process moves records of them from the log file into their database's journal files. Then the process sleeps until sufficient portions of the log file are ready to be archived again.

1.4 The General Communications Facility

The General Communication Facility (GCF) manages communications between the name server (GCN) and the communications server (GCC).

1.4.1 The Name Server

The name server is a daemon process called **iigcn**. You have a single name server for your ULTRIX/SQL installation. The name server establishes local database management system server and database management system server inter-node communications. It provides the name translation services used to establish local database management system server connections. When you start up a database management system server, it registers with the name server by default.

As an ULTRIX/SQL NET service, the name server provides communications networks with specific information for database management system server inter-node communications.

1.4.2 The Communications Server

Your installation includes ULTRIX/SQL NET, which provides a communications server for your installation. This server is a daemon process named **iigcc** and is the network communication element of the ULTRIX/SQL NET component. The communications server monitors outgoing communication from local user interfaces to remote database management system servers and incoming communication from remote user interfaces to local database management system servers. Your installation can have multiple **iigcc** processes.

1.5 The Database Management System (DBMS) Server

Multiple user interfaces can access databases through a single database manager, the database management system server. The database management system server (**iidbms**) is a daemon process. It has associated slave processes, **iislave**, that execute asynchronous disk and communication I/O for the database management system server.

The database management system server process is divided into the following components:

- Abstract Data Type Facility (ADF)
- Data Manipulation Facility (DMF)
- Optimizer Facility (OPF)
- Parser Facility (PSF)
- Query Execution Facility (QEF)
- Query Storage Facility (QSF)
- Relation Description Facility (RDF)
- System Control Facility (SCF)

The database management system server facilities that you should be familiar with are described in Chapter 7.

Your database management system server is configurable. You may set the number of sessions that may connect to your server, the number of open databases allowed, and other parameters. You can change the database management system server configuration file during any startup.

1.6 Getting Started with the Initialization Procedure

Before you begin the ULTRIX/SQL initialization procedure, read and complete the initialization checklist in Chapter 4.

Overview of Installation and Operations Utilities 2

2.1 ULTRIX/SQL Installation Utilities

As the ULTRIX/SQL System Administrator, you start up, shut down, and modify your ULTRIX/SQL installation using the utilities described in this chapter. They are located in the $II_SYSTEM/sql/utility directory where only the ULTRIX/SQL System Administrator has permission to execute them. Appendix E provides reference material for the commands that run these utilities.

2.1.1 Initialize ULTRIX/SQL—iibuild

To initialize ULTRIX/SQL, use **iibuild** and the initialization procedure described in Chapter 4. The **iibuild** utility performs the following actions during the initialization procedure:

- Ensures that your ULTRIX/SQL installation environment is set up
- Sets up ULTRIX/SQL environment variables for your installation
- Configures and starts up your installation's daemon process and servers
- Creates the ULTRIX/SQL system (**iidbdb**)
- Sets up the ULTRIX/SQL network communications (NET) component

2.1.2 Installation and Server Startup—iistartup

All the elements of your ULTRIX/SQL installation must be initialized and configured, and its daemon processes started in the correct sequence, before any users can connect to their databases. (See Chapter 4 for details.)

To reconfigure elements of your installation, run **iistartup** interactively using its configuration flag. The configuration flag causes it to prompt for configuration parameters.

To restart your ULTRIX/SQL installation using the parameters you supplied during a previous configuration, invoke **iistartup** without a flag. It will automatically start your installation's daemon processes and servers, including one database management system server.

You can use **iistartup** after your installation's startup to start additional database management system servers. You have the option to use the previously specified database management system parameters or to use a flag that allows you to reconfigure the server.

The **iistartup** utility performs the following actions:

- Checks the ULTRIX system for memory, semaphores, and swap space using **syscheck**
- If necessary, sets up ULTRIX shared memory and semaphore resources for ULTRIX/SQL
- If necessary, starts up the name server (**iigcn**)
- If necessary, starts up the communications server (**iigcc**)
- Optionally, reconfigures the transaction log file
- Optionally, reconfigures the logging and locking system parameters
- If necessary, starts up the recovery (**dmfrcp**) and archiver (**dmfrcp**) processes
- Optionally, reconfigures the database management system server (**iidbms**) options
- Starts up a database management system server (**iidbms**) with options saved in the rundbms.opt file

2.1.3 Installation Shutdown—iishutdown

Your ULTRIX/SQL installation's daemon processes will run until you shut them down using **iishutdown**. You need to shut down your ULTRIX/SQL installation's processes in the correct sequence before you bring down your ULTRIX system. You also need to shut down your installation or parts of it to reconfigure.

You can use **iishutdown** to shut down your entire ULTRIX/SQL installation in sequence or select specific elements to shut down. For example, you can use the utility to shut down individual servers. After prompting you for confirmation, **iishutdown** performs the following actions:

- Stops the database management system servers (**iidbms**)
- Stops the recovery (**dmfrcp**) and archiver (**dmfacp**) processes
- Deallocates shared memory resources
- Stops the communications servers (**iigcc**)
- Stops the name servers (**iigcn**)

2.2 Programs Called by the iistartup and iishutdown Utilities

The following programs reside in the $II_SYSTEM/sql/utility directory and are used by **iistartup** and **iishutdown**. They may also be run interactively by the ULTRIX/SQL System Administrator:

- **csinstall**—Executable that allocates ULTRIX shared memory and semaphore system resources based on the value assigned II_LG_MEMSIZE
- **cscleanup**—Utility that deallocates ULTRIX/SQL shared memory resources
- **iirungcc**—Executable that starts up the communications server
- **iirungcn**—Executable that starts up the name server
- **iirun**—Executable that starts up the recovery and archiver processes
- **iirundbms**—Executable that starts up a database management system server
- **mkmakelog**—Script that prompts for transaction log file size parameter and creates the log file
- **syscheck**—Utility that checks your ULTRIX system configuration of memory, semaphores, and swap space

Configuration Decisions 3

3.1 Introduction

Before you initialize ULTRIX/SQL, make sure your machine meets the minimum system hardware and configuration requirements described in this chapter. You must set up your machine correctly because you will need to start up your ULTRIX/SQL installation in order to complete the initialization procedure.

During the procedure, you will be prompted for the paths of directories in which you will install the different elements of your ULTRIX/SQL installation. Use this chapter to help you decide how to configure the ULTRIX/SQL installation on your machine.

3.2 General Guidelines for Avoiding Problems

The $II_SYSTEM/sql directory, normally /usr/lp/sql/sql, may be used as the home directory of the ULTRIX/SQL System Administrator, the **ingres** user. This account should be used only for ULTRIX/SQL administration. No user files or directories should be placed in the $II_SYSTEM directory or its subdirectories.

The files in the $II_SYSTEM directory and subdirectories are critical to ULTRIX/SQL, although many of their functions are undocumented. To delete or change any of these files might corrupt your installation.

ULTRIX/SQL uses operating system permissions to protect your data. Never alter the permissions of any ULTRIX/SQL file, directory or subdirectory.

3.2.1 User ingres Requires kmem Group Membership for Certain Operations

The new user group **kmem** has been added to ULTRIX for purposes of increased system security. User **ingres** does not belong to this group by default but must belong to it when the following database administrative procedures are being performed:

- Initializing ULTRIX/SQL
- Expanding or checking ULTRIX/SQL resources after initialization

In more specific terms, user **ingres** requires **kmem** membership when you are running the **iibuild, syscheck,** and **rcheck** utilities. These utilities reside in the directory /usr/lp/sql/utility.

How you give user **ingres** membership in group **kmem** depends on how services are ordered in the file /etc/svc.conf for the group entry. If local service is listed first in the file, you must modify the /etc/group file. If **yp** or **bind** is listed first, you must add group membership according to documentation for the Yellow Pages (yp) or BIND/Hesiod (bind) system naming services.

In order for user **ingres** to operate with the required privilege enhancements provided by **kmem** membership, you must log in as user **ingres** (or **su** to user **ingres**) *after* adding the **kmem** group membership to the appropriate file.

On completion of the database administrative task that requires enhanced privileges, you can remove kmem membership from user **ingres.**

For more information about system security, refer to the *ULTRIX Security Guide for Administrators.*

3.2.2 ULTRIX/SQL Operation Affected by ULTRIX System Security Levels

ULTRIX/SQL will not allow access if the system is operating at the ENHANCED security level.

At the UPGRADE security level, ULTRIX/SQL will allow proper user access only if the password information has been retained in the /etc/passwd file for each user or, if you are naming the YP or BIND naming service, in the password source file for the naming service. When password information is retained in the file, an asterisk (*) does not appear in the password field, which is the second field delimited by a colon (:).

The ULTRIX security level is determined by the SECLEVEL entry in the /etc/svc.conf file. For more information on security levels, refer to the *ULTRIX Security Guide for Administrators.*

3.3 System Requirements

This section describes the minimum system hardware and configuration requirements that enable you to start up your ULTRIX/SQL installation.

3.3.1 Network Protocols

To run ULTRIX/SQL, your system must run the TCP/IP network protocol. (The TCP/IP subset of ULTRIX is required for loading ULTRIX/SQL subsets.) The General Communication Facility (GCF) uses the TCP/IP protocols for communication between the components of your installation. You can check your TCP/IP installation by doing a loopback test. Use **rlogin** or **ftp** from your local node and connect to that same local node.

The ULTRIX/SQL NET component can interface with either TCP/IP or DECnet. If you want the NET component of ULTRIX/SQL to interface with DECnet protocols, DECnet must be installed on your system.

3.3.2 Disk Space

The *ULTRIX/SQL Installation Guide* indicates the amount of disk space required to read in your ULTRIX/SQL installation. In addition, a minimum of 4096 Kbytes are required for your log file. You must also estimate the space you will need for databases, journals and checkpoints.

3.3.3 Random Access Memory

Your system needs a minimum of 8 Mbytes of random access memory (RAM) to have sufficient memory for database management system server configuration.

3.3.4 Shared Memory

To run ULTRIX/SQL, you must configure your ULTRIX kernel with sufficient shared memory resources to support ULTRIX/SQL installation, locking and server requirements. Your ULTRIX/SQL installation has the following shared memory requirements:

- 8192 bytes for a system segment.
- A minimum of 204800 bytes for a lock segment.
- Each database management system server also requires a shared memory segment. The size is calculated as 16384 + (8708 * *max_connected_sessions)*. The value for *max_connected_sessions* is the same value used for the server parameter of that name. See Chapter 6 for server parameter information.

For example, to configure shared memory for an installation with a single server using the default maximum of 25 connected sessions:

The installation segment requires:

8192 bytes of shared memory

The locking segment requires a minimum of:

204800 bytes of shared memory

The shared memory requirements for the server segment are calculated as follows:

16384 + (8708 * 25) = 234084 bytes of shared memory

The total shared memory requirement for this installation is a minimum of 447076 bytes:

8192 + 204800 + 234084 = 447076

The minimum shared memory configuration possible is 238084 bytes. This would be for an installation with a single server, configured for a maximum of one connected session.

3.3.4.1 Shared Memory—VAX Systems

On ULTRIX VAX systems, you will need to change or introduce the following shared memory kernel parameters:

smbrk—Definition of spacing between private and shared data space

smmax—Maximum size in blocks for a shared memory segment

smseg—Maximum number of shared memory segments per process

If your system has less than 32 Mbytes of internal memory, you will also need to change the value of the **smsat** parameter. This parameter defines the highest attachable address in Mbytes for shared memory segments.

See your ULTRIX system management documentation for a detailed explanation. The following values are recommended for these parameters:

smbrk	30720
smmax	2048
smseg	16
smsat	32

Edit the system configuration file, normally /sys/conf/vax/*HOSTNAME* where *HOSTNAME* is the host name of your machine, to include these kernel parameters. (The *HOSTNAME* must be entered in uppercase letters.) Relink your kernel as instructed in the *ULTRIX Guide to System Configuration File Maintenance.*

3.3.4.2 Shared Memory—RISC Systems

On ULTRIX RISC systems, you will need to change or introduce the following shared memory kernel parameters:

smmax—Maximum size in blocks for a shared memory segment.

smseg—Maximum number of shared memory segments per process.

See your ULTRIX system management documentation for a detailed explanation. The following values are recommended for these parameters:

smmax	256
smseg	16

Edit the file /sys/conf/mips/*HOSTNAME* where *HOSTNAME* is the host name of your machine to include these kernel parameters. (The *HOSTNAME* must be entered in uppercase letters.) Relink your kernel as instructed in your *ULTRIX Guide to System Configuration File Maintenance.*

3.3.5 Semaphores

Your ULTRIX/SQL installation requires a semaphore set of 21 semaphores. This number is equal to the *maximum_number_servers* (a hard-wired limit of 16) plus 5. In addition, each server requires a separate set of 10 semaphores. A minimum system configuration requires 31 semaphores.

The value of **SEMMNS** in the /sys/h/sem.h file determines the maximum number of semaphores available on the system. This is the number of semaphores that must be shared among all applications. The ULTRIX operating system comes configured with a maximum of 60 semaphores.

A typical installation requires a maximum number of 120 semaphores. To raise the number of semaphores for additional application needs, edit the value of **SEMMNS** and remake your kernel as instructed in the *ULTRIX Guide to System Configuration File Maintenance.*

3.3.6 Swap Space

A minimum of 16 Mbytes of swap space is recommended for an ULTRIX/SQL installation with one database management system server. You need to add an additional 3 to 5 Mbytes for each additional database management system server, depending on the configuration of maximum connected sessions.

3.4 Environment Variables

This section contains descriptions of the environment variables that are set during the initialization procedure through **iibuild**. With one exception, the initialization procedure uses the **ingsetenv** command to set ULTRIX/SQL environment variables. ULTRIX/SQL variables set with **ingsetenv** are stored in the ULTRIX/SQL symbol table. They are not available to the ULTRIX environment and can only be displayed with the **ingprenv** command.

The exception to the use of **ingsetenv** is II_SYSTEM. Each user, including the ULTRIX/SQL System Administrator, **ingres**, must set this variable in his/her local ULTRIX environment to use ULTRIX/SQL.

Note

> For each of II_DATABASE, II_CHECKPOINT, II_JOURNAL, and II_LOG_FILE, if an area other than the default II_SYSTEM is going to be specified during the **iibuild** procedure, the named directory must either already exist and be owned by **ingres** or the user **ingres** must have permission to create the named directory using the **mkdir** command. The ULTRIX/SQL installation procedure provides an opportunity to create these alternate areas.

3.4.1 II_SYSTEM

There is no required location for your ULTRIX/SQL installation, so ULTRIX/SQL uses II_SYSTEM to locate components of your installation. Users must set II_SYSTEM in their own environments before they can run ULTRIX/SQL. ULTRIX/SQL superusers must also set II_SYSTEM in their own environments before initializing ULTRIX/SQL.

Set II_SYSTEM in the ULTRIX environment to the full pathname of a directory. Underneath that directory, the installation procedure creates an sql and an ingres subdirectory. The **setld** installation process loads ULTRIX/SQL startup files and subdirectories into these directories during the installation procedure. The default location for the ULTRIX/SQL installation is /usr/lp/sql. Local data directories for the installation are linked into /usr/var/lp/sql.

For your convenience, four startup files are created during the installation of ULTRIX/SQL. The person who is the administrator for ULTRIX/SQL and ULTRIX/SQL users can include the appropriate startup files in their .cshrc, .login or .profile files to set up the proper ULTRIX/SQL environment variable and execution path. For more information on these files, refer to Appendix A.

We recommend that you use the $II_SYSTEM directory only for ULTRIX/SQL administration.

3.4.2 II_INSTALLATION (Installation Code)

You may have more than one ULTRIX/SQL installation on a machine, so the ULTRIX/SQL environment variable II_INSTALLATION is used to uniquely identify each installation. During the ULTRIX/SQL initialization procedure, you assign a unique two-character identification code for the installation. The two characters that you select must be alphanumerics (such as "xx" or "x4"), and the first character must be a letter. II_INSTALLATION is stored in the ULTRIX/SQL symbol table as the two-character code that you choose. The environment variable uses **ingsetenv** to set II_INSTALLATION. II_INSTALLATION should *never* be defined in the local ULTRIX environment.

The default value for II_INSTALLATION, provided during the **iibuild** procedure, is "II." Note that you may have the same installation code on different hosts connected through NET, as long as each installation on a host has a unique code for that host.

3.4.3 II_LOG_FILE (Transaction Log File Location)

ULTRIX/SQL uses an installation-wide transaction file. This file is used to handle all of the installation's concurrent ULTRIX/SQL transactions. The logging system writes pending transactions to the log file and the archiver moves completed transactions to the journal files when necessary. The full pathname of the log file is $II_LOG_FILE/ingres/log/ingres_log.

The default value for II_LOG_FILE is the value of II_SYSTEM. Sites must determine the best place for this file to reside. Make sure *not* to place the logging file on an I/O bound disk. This increases the data acquisition times of the recovery and archiver routines, which will slow down all ULTRIX/SQL users.

The size of the log file is another important factor. It *must* be large enough to handle all concurrent transactions without reaching saturation. It is designed as a circular file that wraps when the physical end-of-file is reached. If the log file reaches the FORCE-ABORT-LIMIT, it backs out the oldest transactions dynamically. If it is not successful in freeing enough space and the LOG-FULL-LIMIT is reached, the transaction system stops and ULTRIX/SQL backs out the oldest transactions. This could have severe performance implications and should be avoided.

To calculate the required size, examine queries, calculate the number of ULTRIX/SQL pages they touch, and multiply by the number of concurrent users. As with the log and locking system parameters, it is recommended that you gauge these values on the high side.

During initialization, II_LOG_FILE is set to the full pathname of a directory. Under that directory, the installation procedure makes, if necessary, an ingres directory and in the ingres directory makes a login subdirectory.

3.4.3.1 Choosing Between Regular and Raw Log File Options

You may configure the installation-wide log file either as an ordinary filesystem file (regular file) or as a raw device. The raw device option can provide better performance. However, you will probably have to reconfigure your system to configure for a raw log (see the following section, "Configuring the Raw Log File"). Using the regular log file option allows for much easier installation and reconfiguration of size.

Performance benefits of a raw log file are related to:

- Directly writing to disk, bypassing the ULTRIX buffer cache
- Writing of larger disk blocks

ULTRIX/SQL guarantees data integrity by ensuring that all necessary log file records are on disk before indicating to the user process that a transaction is complete. The same is also true of the files that ULTRIX/SQL uses to store data, except when you use the **fast_commit** server option (see "iirundbms and Database Management System Server Startup Parameters" in Chapter 6). This writing to disk involves both file data and the file **inode** (file descriptor) in the case of a regular log file, but only involves the file data in the case of a raw log file, as there is no filesystem on the device that is configured for the ULTRIX/SQL raw log file. Thus, using the raw log option reduces the number of system I/Os required per log-file write.

Use of the raw log file option is of significant performance benefit only if the disk with the log file becomes saturated. You may wish to use the regular log file option until you have more experience working with ULTRIX/SQL and you know what size to make the log file. Then you can reconfigure the system to create a suitably sized raw device. The sections called "Changing the ULTRIX/SQL Log File Size" and "Changing the Log File Type" in this chapter describe the procedures for changing the log file size and type.

3.4.3.2 Configuring the Raw Log File

Once you know what size log file you need for your installation, create a raw device of that size on your system. The device must not contain a filesystem and therefore does not show up in the output of the ULTRIX **df** command, just as raw devices used for swap space on your system do not show up. Note that the ULTRIX/SQL log file must be at least 4096 Kbytes in size.

Take an example of partition "d" on disk "sd1" being configured as a raw device. The relevant special device files in "dev" are shown below.

The character or raw block device:

```
crw-r-----  1  root     17,    3 Dec 16 12:52 /dev/rsd1d
```

The block device:

```
brw-r-----  1  root      7,    3 Dec 16 12:52 /dev/sd1d
```

The raw block device is the one that ULTRIX/SQL uses when accessing the raw log file.

By entering $II_SYSTEM/sql/utility/mkrawlog, the ULTRIX root user creates the raw log which prompts for a special device file name (in the example above, this would be /dev/rsd1d). After checking that the device name is a raw device file, that it does not contain a filesystem, and that it is large enough, this command creates a special device file in $II_LOG_FILE/ingres/log with the same major and minor device numbers as the selected device:

```
crw-------  1  ingres     17,   3 May 15 16:43 ingres_log
```

The owner of this special device is **ingres,** which permits ULTRIX/SQL processes to write to this file.

Once you run the **mkrawlog** script, your ULTRIX/SQL system is properly configured for using the raw log file option.

3.4.3.3 Changing the ULTRIX/SQL Log File Size

Caution

> The procedures described here include ensuring a clean shutdown of all ULTRIX/SQL processes with **iishutdown** (which calls **rcpconfig -shutdown** when the option to shut down the archiver and recovery processes is taken). If these procedures are followed *without* using this method to ensure a clean shutdown of ULTRIX/SQL processes, there is a risk of data loss.

If the log file is a regular file, take the following steps to change the log file size:

1. Ensure that all ULTRIX/SQL processes are shut down with **iishutdown.**

2. Run **iistartup -init**, which offers the opportunity to specify a new size for the log file.

3. If the log file is a raw device, you cannot directly change the size, as the log file size is the same as the raw device's. If you require a larger log file, but it is not convenient or possible to configure a larger raw device, change the log file type to a regular file. This allows simple size reconfiguration.

To change the raw log file size:

1. Ensure that all ULTRIX/SQL processes are shut down with **iishutdown**.

2. Reconfigure your system for a larger raw device or select an alternate dedicated raw device for the log.

3. Delete the $II_LOG_FILE/ingres/log/ingres_log special device file.

4. Run $II_SYSTEM/sql/utility/mkrawlog as the ULTRIX root user.

5. Run **iistartup -init**.

3.4.3.4 Changing the Log File Type

Caution

> The procedures described here include ensuring a clean shutdown of all ULTRIX/SQL processes with **iishutdown** (which calls **rcpconfig -shutdown** when the option to shut down the archiver and recovery processes is taken). If these procedures are followed *without* using this method to ensure a clean shutdown of ULTRIX/SQL processes, there is a risk of data loss.

To change from using a raw log to a regular log file, follow these steps:

1. Ensure that all ULTRIX/SQL processes are shutdown with **iishutdown**.

2. Delete the $II_LOG_FILE/ingres/log/ingres_log special device file.

3. Run **iistartup -init**, selecting the **ordinary filesystem file** option for the log file. This offers the opportunity to specify the log file size.

To change from using a regular log file to a raw log, follow these steps:

1. Ensure that all ULTRIX/SQL processes are shutdown with **iishutdown**.

2. Reconfigure your system as necessary (see the section "Configuring the Raw Log File" earlier in this chapter) so that your system has a suitable raw device available for ULTRIX/SQL log usage.

3. Delete $II_LOG_FILE/ingres/log/ingres_log.

4. As the ULTRIX root user, run $II_SYSTEM/sql/utility/mkrawlog to create the special device file $II_LOG_FILE/ingres/log/ingres_log. This allows ULTRIX/SQL processes to directly write to the raw partition.

5. Run **iistartup -init**, which will automatically detect the presence of the raw log.

3.4.4 II_DATABASE

The ULTRIX/SQL default directory for database files is, by default, in the ii_database location. The directory that ii_database maps to is defined by the ULTRIX/SQL environment variable II_DATABASE.

II_DATABASE, first set during the ULTRIX/SQL system initialization procedure, determines the location of the ULTRIX/SQL database database, **iidbdb**. Because II_DATABASE location information is written into a configuration file associated with the **iidbdb**, the assignment is permanent. II_DATABASE cannot be changed later, not even during ULTRIX/SQL updates. Note, however, that any database except the **iidbdb** may be created in an alternate location or unloaded and re-created in an alternate location.

During initialization, II_DATABASE is set to the full pathname of a directory. Underneath that directory, the installation procedure makes, if necessary, an ingres subdirectory. Under ingres it makes a data subdirectory, and beneath that a default subdirectory. After initialization, each time a DBA creates a new database and does not identify an alternate location, ULTRIX/SQL will locate it in a subdirectory that it made under default. The subdirectory will have the same name as the database it holds. Then, by default, tables and certain temporary files associated with the database will be created in the database's directory.

3.4.5 II_CHECKPOINT (Default Checkpoint Location)

II_CHECKPOINT, first set during the ULTRIX/SQL initialization procedure, determines the default checkpoint location for the installation, ii_checkpoint. Because II_CHECKPOINT location information is written into a configuration file associated with the **iidbdb**, the assignment is permanent. It cannot be changed later, not even during ULTRIX/SQL updates.

Unless a DBA specifies an alternate checkpoint location when creating a database, ULTRIX/SQL stores checkpoints (static database backups) in the ii_checkpoint location.

During initialization, II_CHECKPOINT is set to the full pathname of a directory. Under that directory, the installation procedure makes, if necessary, an ingres directory and in the ingres directory made a ckp subdirectory.

3.4.6 II_JOURNAL (Default Journal Location)

II_JOURNAL, first set during the ULTRIX/SQL initialization procedure, determines the default journal location for the installation, ii_journal. Because II_JOURNAL location information is written into a configuration file associated with the **iidbdb**, the assignment is permanent. II_JOURNAL cannot be changed later, not even during ULTRIX/SQL updates, so select your default journal location carefully.

Unless a DBA specifies an alternate journal location when creating a database, ULTRIX/SQL stores journals (up-to-the-minute database backups), in the ii_journal location.

During initialization, II_JOURNAL is set to the full pathname of a directory. Under this directory, the installation procedure makes, if necessary, an ingres directory and a jnl subdirectory.

Initializing ULTRIX/SQL and Authorizing User Access 4

4.1 Introduction

This chapter contains the ULTRIX/SQL initialization procedures and an initialization checklist. The initialization procedure is identical for ULTRIX VAX and ULTRIX RISC systems. It must be used along with the initialization checklist located at the end of this chapter.

Note

If your system supports Network File System (NFS) and you plan to use NFS-mounted file systems with ULTRIX/SQL, read Appendix F before you begin the initialization procedure.

4.2 ULTRIX/SQL Initialization Procedure

Note

Read the checklist at the end of this chapter and fill in the blanks before you begin the initialization procedure.

1. Make sure you have loaded the ULTRIX/SQL software as documented in the *ULTRIX/SQL Installation Guide* before proceeding.

2. Log in as or **su** to the **ingres** account and set the II_SYSTEM environment variable to the directory you selected for the installation. This variable is used by the ULTRIX/SQL programs to locate the database management system installation. II_SYSTEM/sql is where the subdirectories containing ULTRIX/SQL executables and libraries are installed. The following examples set the II_SYSTEM environment variable:

   C shell example:

   ```
   setenv  II_SYSTEM  /install/r1
   ```

Bourne shell example:

```
II_SYSTEM=/install/r1
export  II_SYSTEM
```

Each user must define this variable in his or her environment to run ULTRIX/SQL.

4. Include II_SYSTEM/sql/bin and $II_SYSTEM/sql/utility in the search path of the **ingres** user.

   C shell example:

   ```
   set path = ($II_SYSTEM/sql/{bin,utility} $path)
   ```

   Bourne shell example:

   ```
   PATH=/install/r1/sql/bin:/install/r1/sql/utility:$PATH
   export  PATH
   ```

   Each user must include $II_SYSTEM/sql/bin in his or her search path to run ULTRIX/SQL. Users other than **ingres** do not include $II_SYSTEM/sql/utility in their search paths.

5. Execute the **iibuild** command to begin the initialization of your ULTRIX/SQL software:

   ```
   iibuild
   ```

   A readable log of the output of **iibuild** is maintained in the $II_SYSTEM/sql/files/install file. This file may be viewed or printed at any time during or after the execution of iibuild. Once you have successfully run **iibuild**, the initialization procedure is complete. If **iibuild** is run again, the log of the previous **iibuild** session is saved in the $II_SYSTEM/sql/files/.install file. Now, authorize the appropriate users to access ULTRIX/SQL with the **accessdb** command, as explained in Appendix B of this guide.

   Note

   C shell users will need to invoke the **rehash** command prior to invoking **iibuild** so that the files in the bin and utility directories can be found on the path.

4.3 Initialization Checklist

You need to collect all the following information before you begin the ULTRIX/SQL initialization procedure. Please fill in the blanks on this checklist. Retain this checklist for future reference. Have it completed and available in the event that you need to contact Digital about an ULTRIX/SQL problem.

- Your ULTRIX system configuration meets the minimum recommended ULTRIX/SQL resource requirements?

  If desired for interface with the NET component, the system is running DECnet protocol. (TCP/IP must also be running even if you answer yes to this question.) (y/n) __________

  RAM of at least 8 Mbytes (y/n) __________

  Shared memory of 238084 bytes or more (y/n) __________

  31 or more semaphores (y/n) __________

  Swap space of 16 Mbytes or more (y/n) __________

  If you answered "no" to any of these questions, please read Chapter 3 before continuing.

- Read the chapter "Initializing ULTRIX/SQL NET" in the *ULTRIX/SQL NET User's Guide* and determine the maximum number of connected sessions.

  Maximum number of outbound connected sessions: __________

  Maximum number of inbound connected sessions: __________

- *Your two-letter ULTRIX/SQL installation code is:* __________

- Read the configuration discussion in Chapter 3 and choose default locations for your databases, checkpoints, journals, and logging file.

  For your databases, set II_DATABASE to: __________

  For your checkpoints, set II_CHECKPOINTS to: __________

  For your journals, set II_JOURNAL to: __________

  For your logging file, set II_LOG_FILE to: __________

- *Do you want the log file for this installation to be a raw device (y/n)* __________

  Read Chapter 3 for information on the log file. If you answered "yes" to this question, read the instructions for creating a raw log device in Chapter 3.

- Read the discussion in Chapter 6 for an explanation of logging and locking facility configuration, logging and locking facility parameters, and primary database management system server information.

  Logging parameters

  Enter the transfer block size for the log file.
  Legal block sizes are 8, 16, or 32 Kbytes. Default is 8. __________

  Enter the number of log buffers in shared memory.
  The value must be greater than or equal to 4. Default is 4. __________

  Enter the maximum number of open databases in the logging system.
  The value must be greater than or equal to 10. Default is 32. __________

Enter the maximum number of transactions in the logging system.
The value must be greater than or equal to the maximum number of open databases in the logging system. Default is (maximum number of open databases in the logging system * 4).

Enter the log-full-limit as a percentage of the log file.
Acceptable values are in the range 90 through 99. Default is 95.

Enter the percentage of log file to be used for each consistency point.
Acceptable values are in the range 1 through 75. Default is 5.

Enter the number of consistency points to be taken before invoking the archiver.
Acceptable values are in the range 1 to “LOG-FULL-LIMIT”/“consistency point percentage.” Default is “LOG-FULL-LIMIT”/“consistency point percentage” or 4, whichever value is less.

Enter the force-abort-limit as a percentage.
Acceptable range is from 1 to (“LOG-FULL-LIMIT” - 1). Default is 80.

Locking Parameters

Enter the size of the locks hash table.
The minimum acceptable value is 1. Default is 257.

Enter the size of the resources hash table.
The acceptable range is from 1 to “size of the locks hash table.” Default is “size of the locks hash table.”

Enter the maximum number of locks in the locking system.
The minimum acceptable value is 100. Default is 2000.

Enter the maximum number of lock lists in the locking system.
Value must be greater than or equal to “maximum number of transactions in logging system.” Default is “maximum number of transactions in logging system.”

Enter the maximum number of locks allowed per transaction.
The minimum acceptable value is 10. Default is 80.

- See Chapter 6 for an explanation of the options used in the startup of the database management system server.

Maximum number of server connections allowed:

Maximum number of open cursors per session:

Maximum number of open databases per server:

The per session stack size in bytes:

- Choose the ULTRIX editor to be used by ULTRIX/SQL. The default is the current setting of the ULTRIX $EDITOR variable. If $EDITOR is not defined, the default is /usr/ucb/vi.

Set ING_EDIT to the pathname for your editor:

Maintenance Utilities 5

5.1 Overview

You can use the maintenance and report utilities in this chapter to monitor and administer your ULTRIX/SQL installation. Notice, however, some utility options are restricted to use by an ULTRIX/SQL superuser.

5.2 The Server (iidbms) Maintenance Utility—iimonitor

Database access in ULTRIX/SQL is controlled by the database management system server. You can use the **iimonitor** utility to examine the status of a server and the sessions connected to it. The utility can control the server's execution, including shutting down sessions or the database management system server itself. Notice that administrative options such as stopping the server are restricted to an ULTRIX/SQL superuser. You can use **iimonitor** to:

- Display database management system server information
- Display active sessions for the database management system server
- Stop the database management system server
- Display session information
- Disconnect a session
- Suspend a session

To start the **iimonitor** utility, at the operating system prompt type:

> **iimonitor** *server_name*

Note that *server_name* is the GCF address number supplied by the **iinamu show** operation.

Table 5-1 describes the commands available to the **iimonitor** utility at the IIMONITOR > prompt.

Table 5-1: iimonitor Commands

Command	Function
help	Lists the available commands.
show server	Displays information about the server, including the number of sessions currently active or connected to it, the state of the server, and the CPU usage in terms of quanta used.
show sessions	Displays a list of active sessions and their current state. Explanations of session states are: **CS_EVENT_WAIT** Session is waiting for an event. The event type is shown in parentheses: (LOCK), (DIO) - disk I/O, (BIO) - application communication. **CS_MUTEX** Awaiting a semaphore (access to a system data structure). **CS_COMPUTABLE** Runnable and waiting for a chance to run. **CS_INTERRUPT** Interruptable, either a lock or application I/O request.
set server shut	Disallows additional connections and shuts the server down when currently connected sessions finish. This command may only be run by an ULTRIX/SQL superuser.
stop server	Stops the server immediately. Use this command only if absolutely necessary (for example, if an application is hanging). This command may only be run by an ULTRIX/SQL superuser.

Table 5-2 describes the commands that use the *session_id* to perform actions on a specific server session. The *session_id* is displayed in the **iimonitor** utility with the **show sessions** command.

Table 5-2: Session-Specific iimonitor Commands

Command	Function
format *session_id*	Gives a synopsis of the ULTRIX/SQL information about a session.
remove *session_id*	Disconnects a particular session. This command may only be run by an ULTRIX/SQL superuser.
suspend *session_id*	Suspends a compute-bound session to allow technical support personnel to trace the problem.
quit	Terminates the **iimonitor** session.

5.3 The Name Server (iigcn) Maintenance Utility—iinamu

You can use the Name Service Management Utility (**iinamu**) to display server information and administer the name server. Only an ULTRIX/SQL superuser may execute the administrative options, such as adding or deleting entries, or stopping the **iigcn** process. The following functions are available:

- Show a list of registered servers
- Add a server to the list of registered servers
- Delete a server from the list of registered servers
- Stop the name server process

To invoke the **iinamu** utility, at the operating system prompt type:

```
iinamu
```

When you see the IINAMU> prompt, choose one of the commands from Table 5-3.

Table 5-3: iinamu Commands

Command	Function
show [*svr_type]*	Shows the list of currently registered servers. *svr_type* can be ULTRIX/SQL or COMSVR. ULTRIX/SQL is the ULTRIX/SQL database management system server type and the default. COMSVR is the GCC ULTRIX/SQL NET process type. The following is an example of **show** command output: `ingres * 1201 ingres * 1243` The first column is the server type, the second is a list of databases registered to be served by this database management system server, and the third is the GCF_ADDRESS. The database name entry "*" means that the server has registered to service requests for any database. The GCF_ADDRESS column contains the GCF-specific address for access to this server.
add ***svr_type obj_name gcf_address***	Manually adds to the list of registered servers. This command may only be run by an ULTRIX/SQL superuser.
delete ***svr_type obj_name gcf_address***	Manually deletes a server from the list of registered servers.

Command	Function
stop	Stops the GCF Name Service. This is the correct way to stop the GCN. If GCN is stopped while database management system servers are running no users can connect to those servers. Connected users will function undisturbed until they disconnect their ULTRIX/SQL sessions. This command may only be run by an ULTRIX/SQL superuser.
help	Displays command information.
quit	Quits the **iinamu** utility.

5.4 Shared Memory and Semaphores Report Utility—csreport

You can use the **csreport** utility to display shared memory and semaphore information for your installation. Some information that **csreport** displays is:

- The maximum number of servers configured for your installation
- Shared memory and semaphore information

To invoke the **csreport** utility, at the operating system prompt type :

csreport

The following example illustrates sample output from the **csreport** utility.

```
!Installation version 610008
!Max number of servers 16
!Description of shared memory for control system:
!key 0x490A3007: size 8192 attach 00000000
!Description of shared memory for logging & locking system:
!key 0x490A300A: size 573440 attach 08000000
!Semaphore information for installation:
!      sysV semid 10, num sems 21, used sems 19
! 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
!Event system: used space 4424, length space 8192
!Server info:
!server 0:
!inuse 1, pid 1022, connect id 0, id_number 0, semid 0
!shared memory:
!key 0x00000000: size 0 attach 00000000
!server 1:
!inuse 1, pid 1189, connect id 0, id_number 1, semid 0
!shared memory:
!key 0xFFFFFFFF: size 0 attach 00000000
!server 2:
!inuse 1, pid 1201, connect id 0, id_number 2, semid 0
!shared memory:
!key 0xFFFFFFFF: size 0 attach 00000000
!server 3:
!inuse 1, pid 1235, connect id 1118, id_number 3, semid 11
!shared memory:
!key 0x490A3033: size 131072 attach 00000000
!server 4:
!inuse 0, pid 16696, connect id 0, id_number 4, semid 0
```

```
!shared memory:
!key 0xFFFFFFFF: size 0 attach 00000000
!server 5:
!inuse 0, pid 14174, connect id 0, id_number 5, semid 0
!shared memory:
!key 0xFFFFFFFF: size 0 attach 00000000
!server 6:
!inuse 0, pid 0, connect id 0, id_number 0, semid 0
!shared memory:
!key 0x00000000: size 0 attach 00000000
!server 7:
!inuse 0, pid 0, connect id 0, id_number 0, semid 0
!shared memory:
!key 0x00000000: size 0 attach 00000000
!server 8:
!inuse 0, pid 0, connect id 0, id_number 0, semid 0
!shared memory:
!key 0x00000000: size 0 attach 00000000
!server 9:
!inuse 0, pid 0, connect id 0, id_number 0, semid 0
!shared memory:
!key 0x00000000: size 0 attach 00000000
!server 10:
!inuse 0, pid 0, connect id 0, id_number 0, semid 0
!shared memory:
!key 0x00000000: size 0 attach 00000000
!server 11:
!inuse 0, pid 0, connect id 0, id_number 0, semid 0
!shared memory:
!key 0x00000000: size 0 attach 00000000
!server 12:
!inuse 0, pid 0, connect id 0, id_number 0, semid 0
!shared memory:
!key 0x00000000: size 0 attach 00000000
!server 13:
!inuse 0, pid 0, connect id 0, id_number 0, semid 0
!shared memory:
!key 0x00000000: size 0 attach 00000000
!server 14:
!inuse 0, pid 0, connect id 0, id_number 0, semid 0
!shared memory:
!key 0x00000000: size 0 attach 00000000
!server 15:
!inuse 0, pid 0, connect id 0, id_number 0, semid 0
!shared memory:
!key 0x00000000: size 0 attach 00000000
```

5.5 The Locking Facility Report—lockstat

You can use the **lockstat** utility to display information about installation lock status.

To invoke the **lockstat** utility, at the operating system prompt type:

lockstat

The following example shows typical output from the **lockstat** utility.

```
============ 8-AUG-1989 14:02:17.17 Locking System Summary=========
 Create lock list           42        Release lock list          23
 Request lock              157        Re-request lock             4
 Convert lock               68        Release lock              103
 Escalate                    0        Lock wait                   1
 Convert wait                0        Convert Deadlock            0
 Deadlock Search             1        Deadlock                    0
 Cancel                      0
------------------------Locks by lock list-------------------------
Id: 0001001E Tran_id: 0000009287AB931A R_llb: 00000000 R_cnt: 0
Wait: 00000000 Locks: (0,0/128) Status: NONPROTECT,EWAIT,ESET
Id: 0001001F Tran_id: 0000000000000010 R_llb: 00000000 R_cnt: 0
Wait: 00000000 Locks: (0,0/128) Status: NONPROTECT,NOINTERRUPT
Id: 00010022 Tran_id: 000000000000000E R_llb: 00000000 R_cnt: 0
Wait: 00000000 Locks: (8,0/128) Status: NONPROTECT
    Id: 00030066 Rsb: 0001002C Gr: IS  Req: IS  State: GR PHYS(1)
KEY(BM_DATABASE,DB=00000001)
    Id: 0003006E Rsb: 00010029 Gr: IS  Req: IS  State: GR PHYS(1)
KEY(SV_PAGE,DB=00000001,TABLE=[1,0],PAGE=13)
    Id: 0003000D Rsb: 00010027 Gr: IS  Req: IS  State: GR PHYS(1)
KEY(SV_PAGE,DB=00000001,TABLE=[1,0],PAGE=4)
    Id: 0002005C Rsb: 0001001D Gr: IS  Req: IS  State: GR PHYS(1)
KEY(SV_PAGE,DB=00000001,TABLE=[3,0],PAGE=4)
    Id: 00040062 Rsb: 0002001F Gr: IS  Req: IS  State: GR PHYS(1)
KEY(BM_TABLE,DB=00000001,TABLE=[44,0])
    Id: 00010057 Rsb: 00010058 Gr: IS  Req: IS  State: GR PHYS(1)
KEY(SV_PAGE,DB=00000001,TABLE=[44,0],PAGE=10)
Id: 00010027 Tran_id: 000000000000000B R_llb: 00000000 R_cnt: 0
Wait: 00000000 Locks: (0,0/128) Status: NONPROTECT,NOINTERRUPT
Id: 00010028 Tran_id: 000000000000000A R_llb: 00000000 R_cnt: 0
Wait: 00000000 Locks: (8,0/128) Status: NONPROTECT

    Id: 00010020 Rsb: 00010021 Gr: IS  Req: IS  State: GR PHYS(1)
KEY(SV_PAGE,DB=00000001,TABLE=[1,0],PAGE=20)
    Id: 0001001C Rsb: 0001001D Gr: IS  Req: IS  State: GR PHYS(1)
KEY(SV_PAGE,DB=00000001,TABLE=[3,0],PAGE=4)
    Id: 0002001E Rsb: 0002001F Gr: IS  Req: IS  State: GR PHYS(1)
KEY(BM_TABLE,DB=00000001,TABLE=[44,0])
    Id: 00010018 Rsb: 00010019 Gr: IS  Req: IS  State: GR PHYS(1)
KEY(SV_PAGE,DB=00000001,TABLE=[44,0],PAGE=3)
Id: 00010029 Tran_id: 0000000000000009 R_llb: 00000000 R_cnt: 0
Wait: 00000000 Locks: (0,0/128) Status: NONPROTECT,NOINTERRUPT
------------------------------Locks by resource--------------------
Id: 00010001 Gr: IS  Conv: IS  Value: 00000000000
KEY(SV_PAGE,DB=00000001,TABLE=[1,0],PAGE=15)
    Id: 00010070 Llb: 0001002C Gr: IS  Req: IS  State: GR PHYS(1)
Id: 00010003 Gr: IS  Conv: IS  Value: 00000000000
KEY(SV_PAGE,DB=00000001,TABLE=[1,2],PAGE=2)
    Id: 00010002 Llb: 0001002C Gr: IS  Req: IS  State: GR PHYS(1)
Id: 00010019 Gr: IS  Conv: IS  Value: 00000000000
KEY(SV_PAGE,DB=00000001,TABLE=[44,0],PAGE=3)
    Id: 00010010 Llb: 0001002C Gr: IS  Req: IS  State: GR PHYS(1)
    Id: 00010018 Llb: 00010028 Gr: IS  Req: IS  State: GR PHYS(1)
Id: 0003006D Gr: IS  Conv: IS  Value: 00000000000
KEY(BM_TABLE,DB=00000001,TABLE=[151,0])
    Id: 0003006C Llb: 0001002C Gr: IS  Req: IS  State: GR PHYS(1)
===================================================================
```

5.5.1 Interpreting the Locking System Summary

The first portion of the output is a summary listing of locking activity for this installation. All values are cumulative from the time **iistartup** was run for this iteration of the system. Table 5-4 describes the meaning of each entry.

Table 5-4: Locking System Summary Values

Value	Function
Create lock list	Indicates the number of times a lock list was created on behalf of a server, session, or transaction.
Release lock list	Indicates the number of times a release of a lock list occurred on behalf of a server, session, or transaction.
Request lock	Indicates the number of new lock requests that the ULTRIX/SQL locking system processed.
Re-request lock	Gives the number of times an implicit lock conversion request was issued on a resource that the lock list already had locked. Implicit lock conversion requests can occur when a request is made on a page for update that was previously requested for read.
Convert lock	Shows the number of times an explicit lock conversion request is made to change a lock mode on a physical lock from one mode to another. These types of requests occur as a result of a physical lock being converted during an existing transaction to lower or higher modes.
Release lock	Indicates the number of times a specific ULTRIX/SQL logical lock is released, as opposed to a full, partial, or physical lock release.
Escalate	Gives the number of times a partial release occurred to allow escalation of lock granularity from page to table level.
Lock wait	Shows the number of times a new lock request had to wait to be granted.
Convert wait	Gives the number of times an existing lock waited for conversion to a different lock mode.
Convert Deadlock	Indicates the number of times a request for conversion turned into a deadlock.
Deadlock Search	Shows the number of times a deadlock search was initiated.
Deadlock	Shows the number of times that deadlock existed.
Cancel	Gives the number of times a lock request was cancelled due to a timeout or interrupt.

Table 5-5 describes lock list values.

Table 5-5: Lock List Values

Value	Function
Id	Internal lock list identifier (lock list block).
Tran_id	Transaction identifier associated with this lock list. This value correlates to a transaction identifier in the logstat utility output.
R_llb	Related lock list identifier, if not a transaction lock list.
R_cnt	Number of related lock list identifiers that this lock list must assure are released before this lock list can be released.
Wait	Internal resource block identifier of the lock that is currently blocked.
Locks	The three values that make up this entry are, in order: The total number of locks on the list currently The number of logical locks on the list currently The total number of locks allowed to be on this list
Status	Indicates the state of the lock list at the present time. The possible values are: **WAIT** Waiting for lock **NONPROTECT** Can be released without going through a recovery (system lock lists) **ORPHAN** Lock list remaining without transaction **EWAIT** Waiting on system event **RECOVER** Lock list taken over by the recovery process **MASTER** Lock list owned by the recovery process **ESET** Lock list set on wait queue for event **EDONE** Event that lock list is waiting on is done **NOINTERRUPT** Lock requests on this list are non-interruptable

The values indented under individual lock lists are lock block values. Table 5-6 describes these values.

Table 5-6: Lock Block Values

Value	Function
Id	Internal lock block identifier.
Rsb	Internal resource block identifier.
Gr	Granted lock mode.
Req	Requested lock mode.
State	Current state of lock (GR = granted, WT = waiting).
KEY	Information used to identify the resource being locked. When checking contention on data pages, the key will contain PAGE, the database id that can be traced back to logstat output, the table reltid & reltidx, and the page number.

5.5.2 Interpreting the "Locks by resource" Portion

The third portion of **lockstat** output groups the individual locks by resource block and will show any contention that can lead to query performance problems. Table 5-7 describes resource block values.

Table 5-7: Resource Block Values

Value	Function
Id	Internal resource block identifier
Gr	Granted mode of the resource
Conv	Conversion mode requested on the resource
Value	Lock value associated with the resource
KEY	Byte string identifying the resource

The indented portions of the resource blocks show the individual lock blocks that are contending for the resource. Table 5-8 describes these lock block values.

Table 5-8: Individual Lock Block Values

Value	Function
Id	Internal lock block identifier
Llb	Lock list identifier that this lock resides on
Gr	Granted mode of the lock
Req	Requested mode of the lock
State	Current state of the lock. GR indicates granted and WT indicates waiting.

5.6 The Logging Facility Report—logstat

You can use the **logstat** utility to display information about the logging system. The information includes:

- The FORCE-ABORT-LIMIT set during installation
- The percent of the log file used
- The number of open transactions
- A list of open databases

To invoke the **logstat** utility, at the operating system prompt type:

logstat

The following example is typical output from the **logstat** utility.

```
========= Tue Jan 9 16:56:13 1990 Logging System Summary =========
Database add         1198         Database removes      1191
Transaction begins   31529        Transaction ends      31522
Log read i/o's       17782        Log write i/o's       26656
Log writes           151412       Log forces            38300
Log waits            32956        Log splits            0
Log group commit     24508        Log group count       24529
Check commit timer   17117275     Timer write           54
Inconsistent db      0            Kbytes written        120456
---- Current log file header -------------------------------------
Block size:  16384   Block count  :2048       Buffer count:8
CP interval: 1331    Logfull interval :2007   Abort interval:1945
Last Transaction Id:000025609D6A25B2
Begin: <627088820:585:84>CP:<627088820:1075:84>    End:
<627088820:1875:304>
Journal Window:<0,0,0>..<0,0,0>
Status:  ONLINE,ECPDONE

---- List of active processes -------------------------------------
ID          PID         TYPE        OPEN_DB     WRITE FORCE     WAIT        BEGIN       END
-----------------------------------------------------------------------------------------
00010015    00000000                7  17246    235   1849      3706        3701
00010018    00000000                1      0      0   123          2        1
```

```
0001001B  00000000  MASTER          1     3  0         122        1         0
---- List of active databases ------------------------------------
Id:FFFF0001   Database:($recovery, $ingres)   Status:NOTDB
    Tx_cnt:7   Begin:8  End:1  Read:0  Write:3  Force:37  Wait:372
    Location:  None
    Journal Window:<0,0,0>..<0,0,0>
    Id:001C0014   Database:(prod,kimman)   Status:FAST_COMMIT
    Tx_cnt: 0  Begin:8  End:41  Read:0  Write:5  Force:0 Wait:2
    Location:   install/rl/01u/ingres/data/default/prod
    Journal Window:,0,0..,0,0   <0,0,0>
    Id:00210015   Database:(market,kimman)   Status:FAST_COMMIT
    Tx_cnt:0   Begin:9   End:9   Read:0   Write:5   Force:0  Wait:2
    Location:   install/rl/01u/ingres/data/default/mark
    Journal Window:,0,0..,0,0   <0,0,0>
----- List of active transactions---------------------------------
Id:0001000B Tran_id:000025609D6A173D
Database:FFFF0001 Process: 00010015   Session: 00000000
First: <0,0,0>   Last: <0,0,0>   Cp:<627088820,585,84>
    Write: 0   Split:0   Force:11   Wait:3
    Status: INACTIVE
Id:0001000D Tran_id:000025609D6A173C
Database:FFFF0001 Process: 00010015   Session: 00000000
First: <0,0,0>   Last: <0,0,0>   Cp:<627088820,585,84>
    Write: 0   Split:0   Force:9   Wait:0
    Status: INACTIVE
Id:0001000F Tran_id:000025609D6A173B
Database:FFFF0001 Process: 00010015   Session: 00000000
First: <0,0,0>   Last: <0,0,0>   Cp:<627088820,585,84>
    Write: 0   Split:0   Force:9   Wait:1
    Status: INACTIVE
Id:00010011 Tran_id:000025609D6A173A
==================================================================
```

5.6.1 Determining Proximity to FORCE-ABORT-LIMIT

To determine how close you are to reaching the FORCE-ABORT-LIMIT, refer to the section of the **logstat** utility output entitled "Current log file header."

- The first boldfaced number highlights the "Abort interval" of 1945. This figure refers to the number of blocks in the log file that must be filled before the FORCE-ABORT-LIMIT is reached.

- The second highlighted number is part of a numeric series following the word "Begin." The important figure here (585) refers to the block marking the log file's Beginning of File (BOF).

- The third highlighted number is part of a numeric series following the letters "CP." The important figure (1075) refers to the block marking the last consistency point. This consistency point contains a list of all open transactions and open databases between the BOF (585) and the CP (1075).

- The fourth highlighted number is part of a numeric series following the word "End." The important figure (1875) refers to the block marking the log file's End of File (EOF). This is the last block that contains transaction information.

To calculate the number of blocks in use, you need only subtract the BOF from the EOF (1875 - 585). Currently, 1290 blocks are in use in the log file.

To determine how close you are to the FORCE-ABORT-LIMIT, you need only subtract the total blocks used (1290) from the abort interval (1945):

```
1945 - 1290 = 655
```

- Also highlighted in this section is an area called "Status," which in this case states: "ONLINE, ECPDONE". The status provided here is of the logging and recovery systems. "ONLINE" indicates that everything is fine. "ECPDONE" indicates that a consistency point was taken and the status is fine. Table 5-9 describes the options that might appear as the status.

Table 5-9: logstat Utility Status Options

Option	Description
CPNEEDED	The logging system is about to take a consistency point.
ECPDONE	A consistency point was taken and the status is fine.
LOGFULL	The log file is full and transactions will stall until the archiver process catches up.
FORCE_ABORT	The FORCE-ABORT-LIMIT has been reached; the oldest open transaction will be aborted.
ONLINE	Everything is fine.
RECOVER	The recovery process is performing recovery.
ARCHIVE	The archiver process is archiving journaled transactions to the journal files.
ACP_SHUTDOWN	The archiver is preparing to shutdown (this status is displayed when you enter **rcpconfig -shutdown**).
IMM_SHUTDOWN	The logging system has been told to shutdown immediately (this is displayed when you enter **rcpconfig -imm_shutdown**).
START_ARCHIVER	This is an important status that indicates the archiver has died and must be restarted by the DBA. The archiver will not restart automatically; if the DBA does not restart it, the log file will eventually reach full capacity.
PURGEDB	This status appears when a database has been closed by the last user who had it open; the archiver is archiving transactions that belong to this database.

5.6.2 Determining the Active Databases and Users

In order to determine active databases and their users, facilitating notification of users prior to shutting down an installation, refer to the section of the **logstat** report entitled "List of active databases." The first database listed will always be owned by **$ingres**. In this case, **$recovery** is displayed as the database, but the status on this line indicates that this is not a database. This entry shows that ULTRIX/SQL is running fine.

The second database shown here is called "prod" and is owned by "kimman." Note that the ID for this database is FFFF0001. Refer now to the section entitled "List of active transactions." You will see here that the first transaction listed belongs to "Database: FFFF0001." The status of this database (r1) is ACTIVE. Were you about to shut down the installation, you could determine whom to notify by running the logstat utility.

The third database shown is called "market,kimman." Since its ID number does not appear on the list of active transactions, "market" is not currently active. Both other transactions shown in this list are also inactive, and belong to **$ingres**.

5.6.3 Calculating the Number of Blocks in Use

Because the log file is circular, it is not uncommon for the block marking the file's beginning to be a higher number than the block marking the file's end. In the following figure you can see the "Current log file header" from the results of another **logstat** execution.

- The first boldfaced number highlights the "Block count" of 8000, the number of 4K blocks in the log file.
- The "Abort interval" here is 6400.
- The log file's Beginning of File (BOF) is 7662.
- The consistency point (CP) is the same as the BOF (7662). The reason for this is that the amount of space occupied in the log file by open transactions since the last consistency point has not reached the 5% threshold at which a new consistency point is taken.
- The End of File (16) is far smaller than the BOF (7662). Such a discrepancy is possible because the log file is circular. In this case, to calculate the number of blocks in use, you must subtract the BOF from the block count (8000), and then add the EOF to the total, as in the following example:

  ```
  (8000 - 7662) + 16 = 354
  ```

  To determine how close you are to the FORCE-ABORT-LIMIT, subtract the total blocks used from the abort interval:

  ```
  6400 - 354 = 6046
  ```

The log file header

```
------Current log file header------------------------------------
Block size: 16384   Block count: 8000   Buffer count: 1
CP interval:400   Archive interval: 7600   Abort interval: 6400
Last Transaction Id:   0000009114A0049F
Begin: 8495:7662:100   Cp:8495:7662:100   End:8496:16:96
Journal Window: ,0,0..,0,0
Status:   ONLINE, ECPDONE
------ List of active processes ---------------------------------
```

Installation Tuning Reference Material 6

6.1 iirundbms and Database Management System Server Parameters

Access to a database is controlled by the database management system server (**iidbms**). The **iistartup** command invokes **iirundbms** to configure the server, using parameters saved in the $II_SYSTEM/sql/files/rundbms.opt file. If you choose the server configuration option, you are prompted with a selected list of **iirundbms** parameters that will be written to rundbms.opt and used for server startup. If you choose parameters other than those prompted for, you must edit the **iirundbms** with the required parameter list.

The following **iirundbms** parameters are recognized:

Table 6-1: Iirundbms Parameters

Parameters	Description
-connected_sessions *n*	The maximum number of user sessions that are allowed to connect to a database management system server. The default is 24, and *n* may not exceed 50.
-cursors_per_session *n*	The number of simultaneously open cursors per session. The default is 16.
-dmf.*option n*	The Data Manipulation Facility (DMF) manages the interface between the database management system and stored data. The following **dmf** options are available: **cache_size** — The number of individual buffers (single data pages) in the buffer manager. The default is (128 + 4 * sessions). **count_read_ahead** — The number of group buffers (multiple data pages) in the buffer manager. The default is (4 + sessions/4). **size_read_ahead** — The size of the group buffers used in read_ahead operations. The default is 8. **tcb_hash** — The size of tcb hash table used in lookups of table control blocks. The default is 255. For optimal hashing operation, set this option to a prime number.

wbstart

The threshold for modified pages in the cache, after which write-behind starts. When there are wbstart modified pages in the cache, the *write behind* threads wake up and write modified pages out of the cache until the number of modified pages drops below the **wbend** limit. The default is (**mlimit** - 15% of the cache size).

wbend

The lower limit for modified pages. When it is reached, *write behind* threads go to sleep and wait for the **wbstart** limit to be reached. The default is (1/2 of the buffer manager (cache) size). If 1/2 of the cache is not less than **wbstart**, the size defaults to (1/2 **wbstart**).

flimit

The minimum number of free pages that the buffer manager will try to keep available in the cache. When this limit is reached, the buffer manager will begin doing synchronous writes of dirty pages whenever a new page is needed. The default is 1/32 of the buffer manager (cache) size.

mlimit

The maximum number of modified pages that can be left in the buffer manager. When this limit is reached, the buffer manager will begin writing pages out of the cache each time a dirty page is unfixed. The mlimit must be greater than **wbstart.** If **wbstart** is specified and **mlimit** is not, then **mlimit** will default to halfway between **wbstart** and the size of the cache. The default is 3/4 of the buffer manager (cache) size.

memory

The maximum memory usage by the DMF for control blocks and caching. Calculate requirements based on total cache size (single and group buffers) and shade enough for control blocks. Keep in mind that each cache buffer requires 2K (ULTRIX/SQL pages). The default is (500K, Value input * 1024).

-database_count *n*

The maximum number of open databases for the server. The default is the value of the **connected_session** parameter.

-fast_commit

Transaction I/O optimized by using a delayed write algorithm. Used only with **-sole_server** option.

-nopublic

Creates a private database management system server. To access a private database management system server, the II_DBMS_SERVER environment variable must be set to the value reported at startup. The global value of II_DBMS_SERVER is not updated when this option is used.

-qef.*option n*

The Query Execution Facility (QEF) manages query plans and execution. The only **qef** option is **qep_size**. This option translates to the QEF data size used to calculate the maximum memory that QEF will use to build data segment headers. The data segment headers contain all the runtime information needed for the query to be executed. **dsh_maxmem** = 2048 + ((sessions * cursors) * **qep_size**).

-qsf.*option n*

The Query Storage Facility (QSF) manages shared memory between facilities. The only **qsf** option is **pool_size.** This option increases or resets the QSF memory pool by allocating *n* bytes of memory to it. The pool is used to store all the query objects, regardless of type. It serves as the facility that manages a session's memory use. The default size for the QSF pool is ((number of connected sessions * 40K) + 60K).

-scf.*option n*

The System Control Facility (SCF) is the central controlling facility. The only **scf** option is **row_estimate.** This specifies the number of rows to expect on the first pass of a select or retrieve. This is used to build message blocks to send MDEs (message data elements) across GCA to the application programs. This reduces the amount of formatting done before queries are executed. If not enough buffers to send the data from the query are formatted, the additional buffers needed are built after the query completes.

-sole_server

Forces databases accessed by this server to refuse connection requests from other servers. This option should be used when possible to reduce the overhead in managing multiple server connections to common databases.

-stack_size *n*

The per-session stack size in bytes. The default is 32768 bytes.

-write_behind *n*

Allows the creation of *write behind* threads in a database management system server. The write behind threads write dirty pages out of the buffer manager when the cache starts getting full. This keeps user sessions from having to do synchronous writes to free up space in the cache to read in a new page. They also wake up when consistency points are taken to help the *fast commit* thread flush all the dirty pages out of the cache. There are no rules to determine the optimal number of write behind threads to allocate and no way to dynamically add new write behind threads to the server once it is invoked. As a guideline, look at the load on the system and determine the number of writes per second that need to be done (figuring in the number of devices to be written to). Each write behind thread can perform approximately 20 I/Os per second up to the capacity of the operating system. The value for *n* must be greater than 0.

The following options control ULTRIX process parameters:

-maximum_working_set *n*

Sets the server's resident size (**rlimit_rss**) in bytes.

-priority *priority*

Adjusts the server process priority (**prio_process**).

These options may be abbreviated to the minimal unique prefix. For example, **-con** can replace **-connected_sessions.**

6.2 rcpconfig Logging and Locking System Parameters

ULTRIX/SQL builds a single circular log file per installation. The transaction log file contains records used in backing out aborted or incomplete transactions. It also contains records of completed transactions. The archiver takes these records and places them in the corresponding journal files. The log file, ingres_log, is located in the directory defined during the installation procedure by the ULTRIX/SQL environment variable II_LOG_FILE.

For information on configuring the log file as a regular file or raw device, refer to the section entitled "II_LOG_FILE (Transaction Log File Location)" in Chapter 3.

6.2.1 rcpconfig Logging Parameters

This section contains information about the parameters you use to configure the logging system. When you set these values, it is important for you to understand the dependencies between the database management system server configuration and the configuration of the logging system. Before setting these values, read through this entire chapter.

The transfer block size for the log file

The default size of the log buffer transfer block is 8 Kbytes. Performance may improve for large transactions if larger transfer block sizes are used. For small individual transactions, configure a larger number of log buffers with smaller transfer block size; for larger individual transactions, configure a relatively smaller number of log buffers of larger size. Legal block sizes are 8, 16 or 32 Kbytes. The default is 8.

The number of log buffers in shared memory

The number of log buffers corresponds to the number of outstanding I/Os waiting to be placed in the log file. For small individual transactions, configure a larger number of log buffers with smaller transfer block size; for larger individual transactions, configure a relatively smaller number of log buffers of larger size. The value must be greater than or equal to 4. The default is 4.

The maximum number of open databases in the logging system

You must configure the ULTRIX/SQL logging system for at least 10 open databases. Overhead for open database blocks is minimal. When you set this value, keep in mind that every server and connected session requires an open database. Running out of these blocks forces a reconfiguration. The value must be greater than or equal to 10. The default is 32.

The maximum number of transactions in logging system

Every database management system server and connected session requires that a transaction block be available. Do not be conservative when setting this value, as these blocks are not expensive in terms of system resources. If transaction blocks are exhausted, however, you must reconfigure the ULTRIX/SQL system. The minimum value given here is based on the value entered for open databases in the logging system. The value must be greater than or equal to the value of the "maximum number of open databases in the logging system" C parameter. The default is (maximum number of open databases in the logging system * 4).

The LOG-FULL-LIMIT as a percentage of the log file

The LOG-FULL-LIMIT is a hard stopping point. Once the limit is reached, additional transactions will not be processed until sufficient space in the logging file can be cleared. Acceptable values are in the range of 90-99. The default is 95.

The percentage of log file to be used for each consistency point

The logging system uses consistency points to keep track of all active databases, transactions, and lock lists. Consistency points are taken periodically as the log file is used. The latest consistency point is where the recovery process will resume processing after a system crash. How often consistency points are written is determined by the number of consistency points taken before invoking the archiver. The larger the value, the longer recovery will take. Larger log file sizes should use a smaller percentage. There is some performance cost associated with writing consistency points. Acceptable values are in the range of 1-75. The default is 5.

The number of consistency points to be taken before invoking the archiver

This value tells the archiver that a certain number of consistency points are to be written before waking and archiving applicable data involved in that range. If each consistency point involves 5% of the log file, a value of 4 here would wake the archiver each time 20% of the log file is available to be archived. Acceptable values are in the range of 1 to (LOG-FULL-LIMIT/"consistency point percentage" value). The default is the (LOG-FULL-LIMIT/"consistency point percentage" value) or 4, whichever value is less.

The FORCE-ABORT-LIMIT in percentage

The FORCE-ABORT-LIMIT is the soft failure point and aborts the oldest pending transactions. This limit minimizes the chances of reaching the LOG-FULL-LIMIT and halting all transactions in process. The FORCE-ABORT-LIMIT should be set far enough below the LOG-FULL-LIMIT to minimize the probability that the LOG-FULL-LIMIT will be reached. Acceptable range is from 1 to (LOG-FULL-LIMIT - 1). The default is 80.

6.2.2 rcpconfig Locking Parameters

Use the parameters below to configure shared memory for the locking system. When you set these values, it is important for you to understand the dependencies between the database management system server configuration and the locking configuration. Before setting values, read through this entire chapter.

The size of the locks hash table

The ULTRIX/SQL lock manager uses two types of lock lookup tables. The locks hash table is used to locate information about locks held by users. The value that the resources hash table may be given must be less than or equal to the value for the locks hash table. The minimum acceptable value is 1. The default is 257.

The size of the resources hash table

Information about specific resources and associated locks can be determined by using the resources hash table. This value sets the size of the hashed lookup table. The value that the resources hash table may be given must be less than or equal to the value for the locks hash table. The acceptable range is from 1 to "size of the locks hash table value." The default is "size of the locks hash table value."

The maximum number of locks in locking system

The value for the maximum number of locks in the locking system should be based on the sum of all resources and locks required by the ULTRIX/SQL system. A reasonable estimate is 100 to 120 locks per connected session. Set this value based on the expected configuration of the database management system server **connected_session** value. The minimum acceptable value is 100. The default is 2000.

The maximum number of lock lists in locking system

The maximum number of lock lists in the locking system can be determined with the following formula: 50 + 2 * (number of connected sessions for all database management system servers). The default is based on the maximum number of transactions allowed in the logging system. Set this value based on the expected configuration of the database management system server **connected_session** value. The value must be greater than or equal to the value of "the maximum number of transactions in logging system parameter." The default is "the maximum number of transactions in logging system value."

The maximum number of locks allowed per transaction

Specify the maximum number of locks a transaction can take. If this limit is reached, locking will escalate to table level. The minimum acceptable value is 10. The default is 80.

Troubleshooting 7

7.1 Overview

This chapter includes general information about ULTRIX/SQL and your installation that will help you troubleshoot any problems. There is also a section that identifies a few specific problems and gives suggestions for resolving them.

If you experience a problem with your ULTRIX/SQL installation, you can use the following sections to help isolate your trouble:

- **Your Installation's ULTRIX/SQL Processes**—If you have trouble connecting to a server, you may want to examine your installation to verify that all its processes are running and were started up in the correct sequence.

- **Log Files**—If you have trouble starting up an installation process, you will want to check the ULTRIX/SQL log files for error messages. ULTRIX/SQL error, warning, and informational messages are written into log files along with the date and time they occurred. The log file section of this chapter discusses the various ULTRIX/SQL logs and their contents.

- **Error, Warning, and Informational Messages**—Use the message section to help you to understand the ULTRIX/SQL messages that you find in your log files.

- **Facility Descriptions**—ULTRIX/SQL error messages refer to ULTRIX/SQL facilities. Information about the facility associated with an error might help you to isolate your problem.

- **Troubleshooting Your ULTRIX/SQL Installation**—After you determine that your installation is correctly installed and you have checked the ULTRIX/SQL log files, here are some common problems, along with possible solutions.

7.2 Your Installation's ULTRIX/SQL Processes

The installation processes that you start with **iistartup** should continue to run until you use **iishutdown**. Generally, you do not need to shut down your installation except to reconfigure or when your operating system must be brought down. The sequence in which you bring your installation's processes up and down is significant. For example, the name server must be started up before the database management system server processes because database management system processes register with the name server. Using **iistartup** and **iishutdown** to administer your installation will ensure that your installation is started and stopped correctly. The processes necessary for your ULTRIX/SQL installation are brought up in this sequence by **iistartup**:

- **iigcn** (name server) process
- **iigcc** (communications server) process
- **dmfrcp** (recovery) process
- **dmfacp** (archiver) process
- **iidbms** (database management system server) process
- one or more **iislave** processes (associated with the database management system server)

7.2.1 Examine Your Installation's Processes

If your ULTRIX/SQL installation has a problem you can examine it and verify that all the necessary processes are running with the ULTRIX **ps** command. Since you may have more than one ULTRIX/SQL installation, ULTRIX/SQL processes are identified by the two-character installation code, II_INSTALLATION, which is described in chapter 3. To see your installation's processes, log in as or **su** to the **ingres** user and type:

ps -x

Assuming that your installation code is "II" (the default), the following display appears when you examine it with the ULTRIX **ps** command. Output from **ps** varies slightly between machines and your PID and TIME values will be different.

```
PID   TT   STAT  TIME   COMMAND
123   co   IW    0:01   iigcn II
126   co   IW    0:01   iigcc II
129   co   R     1:06   /dmfrcp II
133   co   IW    0:05   /dmfacp II
135   co   S     7:28   iidbms II 1105 3 6
149   co   IW    0:34   iislave 1 0 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
150   co   IW    0:31   iislave 1 0 4 1 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

The four digit number to the right of the **iidbms** process is the communications address of the server. (This number appears in boldface for emphasis.) The communications address is the *server_name* that is used as the argument to the **iimonitor** utility.

When you examine your installation, verify that all the processes necessary for your installation are running and they have your installation code.

7.2.2 Emergency Manual Shut Down of Processes

The following describes an emergency manual procedure to stop an ULTRIX/SQL process. Normally your installation's servers, archiver and recovery, and other processes should be stopped by using **iishutdown.** If this utility is unsuccessful, use the specific utility for the servers/processes:

- Database management system server—**iimonitor**
- Recovery and archiver processes—**rcpconfig**
- Communications server—**netu**
- Name server—**iinamu**

Caution

> Your ULTRIX/SQL installation's processes must be started and stopped in the correct sequence. You can use the order of the preceding list as your shutdown sequence.

You should stop an ULTRIX/SQL process using the ULTRIX **kill** only if the preceding utilities fail. In that case, you will determine its ULTRIX process ID number (PID) and kill it with the SIGQUIT signal. For example, to **kill** a database management system server you would:

1. Log in as the **ingres** user.
2. Determine your ULTRIX/SQL installation code:

   ```
   ingprenv | grep  II_INSTALLATION
   ```

   The two letter installation code will be displayed:

   ```
   II_INSTALLATION II
   ```

3. Determine the installation's servers and their ULTRIX process IDs:

   ```
   ps -ax | grep iidbms
   ```

   The process ID, installation code, and server name will be displayed:

   ```
   135  co  S     7:28  iidbms II 1105               3 6
   ```

4. Kill the database management system server. For example, if you want to kill database management system server 1105, use **kill**, the **QUIT** signal, and the database management system server's ULTRIX process ID 135:

   ```
   kill -QUIT 135
   ```

Note

The following list of ULTRIX signals and their expected effects on a server is for your information only. QUIT is the recommended signal to use.

HUP Terminate the server if there are no active sessions.

TERM Terminate the server when all sessions finish.

QUIT Terminate the server immediately.

KILL Terminate the server immediately. Since the server process terminates abnormally, run **cscleanup**.

If a server process was terminated abnormally with **KILL** instead of **QUIT**, run **cscleanup**. This program will attempt to release global system resources that the server might have owned. Do not run **csinstall** again.

If this facility fails to release shared memory or semaphore resources, use the ULTRIX command **iperm**.

7.3 ULTRIX/SQL Log Files

ULTRIX/SQL creates log files where it writes information about your installation. The files described in this chapter are English text log files that you can use for troubleshooting.

See Chapter 3 for information on the logging transaction associated with the logging and locking facility.

7.3.1 The Error Log

When you have an ULTRIX/SQL problem, check the $II_SYSTEM/sql/files/errlog.log file. Messages about your installation are appended to this log along with their date and time. The errlog.log file contains the following information:

- Archiver shut down
- Database management system server startup and shut down
- Error messages
- Warning messages

Your ULTRIX/SQL System Administrator must maintain the errlog.log file. It will continue to grow until the System Administrator shuts down the installation and manually truncates the log.

7.3.2 The Archiver (dmfacp) Log

The file $II_SYSTEM/sql/files/II_ACP.LOG is overwritten each time the archiver process is started up. ULTRIX/SQL writes information about the current archiver process in this log, such as:

- Archiver startup
- Error messages
- Warning messages

7.3.3 The Recovery (dmfrcp) Log

The file $II_SYSTEM/sql/files/II_RCP.LOG is overwritten each time the recovery process is started up. ULTRIX/SQL writes information about the current recovery process in this log, such as:

- Current logging and locking parameter values
- Error messages
- Recovery operations information
- Warning messages

7.4 Error, Warning and Information Messages

ULTRIX/SQL messages consist of a code segment and message text. The code segment gives you the message type, the facility which generated the message, the error number and abbreviated text. The message text is a one-line summary of the error.

The first two characters of the message code segment identify the message type:

- An error message code segment begins with “E_.”
- Warnings begin with “W_.”
- An “I_” or “S_” prefix indicates an informational message.

The second one or two characters of the code segment point to the ULTRIX/SQL facility that generated the error. Following that is a hexadecimal number and an abbreviated description. The message text is a one-line English text description.

In the following message, the “E_” indicates this is an error message. The second two letters in the error code, “QE,” are the first two letters of the Query Execution facility acronym (QEF). The short error text “_NO_MEM” indicates a memory problem. Finally, the message text refers to another facility, SCF.

```
E_QE001E_NO_MEM  Unable to allocate memory from SCF.
```

By reading the facility descriptions for QEF and SCF, you would learn that these are database management system server facilities that can be configured using **rcpconfig** parameters and options. Identifying the source of this error message might enable you to isolate and correct the problem.

7.5 ULTRIX/SQL Facility Descriptions

Some of the elements of your ULTRIX/SQL installation and its database management system servers are described as facilities. This section contains a list of facilities, their acronyms, and brief descriptions. It is intended to help you isolate ULTRIX/SQL installation problems.

7.5.1 Database Utility Facility (DUF)

DUF includes database utilities such as those that create and destroy databases and modify tables.

7.5.2 Database Management System Server Facilities

The following are database management system server facilities. You may find this section helpful to use with the database management system server tuning chapter in this guide. Some of the database management system server parameters and options that are used with **rcpconfig** to configure database management system server facilities.

7.5.2.1 The Abstract Data Type Facility (ADF)

ADF does all the work that has to do with data types. It manipulates floating point numbers, character strings, integers, all the conversions and comparisons between them, and so on. This facility can be executed independently from the server so that ULTRIX/SQL user interfaces also use ADF.

7.5.2.2 The Data Manipulation Facility (DMF)

DMF is the interface between the database management system and the stored data. It communicates with the logging and locking facility processes and log file. DMF is responsible for data access and transaction services: recovery, locking, and journaling. You have the option to configure DMF for your database management system server using an **rcpconfig** parameter.

7.5.2.3 The Optimizer Facility (OPF)

The optimizer selects the optimal plan for performing a query. It also converts the query tree that comes out of the parser into the query execution plan (QEP). OPF uses a memory pool.

7.5.2.4 The Parser Facility (PSF)

Queries are converted from text that you write to an internal form by PSF, the parser. The parser adds data from the system catalogs to the query, such as information about the table structure and keys that the optimizer needs to make a useful query plan.

7.5.2.5 The Query Execution Facility (QEF)

QEF executes query plans and database utilities. It provides internal query services that other facilities use. QEF manages repeat queries, transactions, and cursors. You have the option to configure QEF for your database management system server using a **rcpconfig** parameter.

7.5.2.6 The Query Storage Facility (QSF)

QSF provides shared memory facilities with a temporary or permanent place to store query trees and query plans. You have the option to configure QSF for your database management system server using a **rcpconfig** parameter.

7.5.2.7 The Relation Description Facility (RDF)

RDF is a central caching point for information about relations. It is used by PSF and OPF.

7.5.2.8 The System Control Facility (SCF)

SCF is the central controlling facility that manages sessions. It coordinates the other facilities executing a query for a user interface. SCF schedules the multiple user interfaces using the database management system server. It manages access to system resources such as memory.

You have the option to configure SCF for your database management system server using a **rcpconfig** parameter.

7.5.3 General Communication Facility (GCF)

GCF manages communication among all the components of ULTRIX/SQL. GCF has three elements whose acronyms you will encounter:

- Name server (GCN)
- Communications server (GCC)
- General Communications Area (GCA)

7.5.3.1 Name Server (GCN)

GCN is the name server (**iigcn**). It assists communications between local and remote database management system servers. It provides the name translation services used to establish local database management system server connections. GCN provides communications networks specific information for ULTRIX/SQL NET.

7.5.3.2 Communications Server (GCC)

GCC is the communications server (**iigcc**). It is the network (NET) component of ULTRIX/SQL.

7.5.3.3 General Communications Area (GCA)

GCA maintains communication connections between ULTRIX/SQL processes on the same local ULTRIX/SQL installation. It is a subroutine library that is a part of all ULTRIX/SQL user interface utilities, database management system servers and the ULTRIX/SQL libraries associated with embedded ULTRIX/SQL.

7.6 Troubleshooting Your ULTRIX/SQL Installation

The following sections contain suggestions for items to check and activities that may help specific problems. Use the information from the previous sections to check processes, log files, and error messages.

7.6.1 Installation Startup Problems

If you have problems starting up your ULTRIX/SQL installation:

- Check the log files for error messages.
- Ensure that the file $II_LOG_FILE/ingres/log/ingres_log exists.
- Ensure that you have the II_SYSTEM variable set in your local environment.
- Ensure that you are the ULTRIX/SQL System Administrator, the **ingres** user.
- Using **accessdb**, ensure that the **ingres** and **$ingres** users are ULTRIX/SQL superusers.

7.6.2 Database Management System Server Connection Problems

If you can start up the database management system server but cannot connect to your databases:

- Ensure that TCP/IP has been correctly installed for your machine.
- Using **ps**, ensure that all your installation's processes are running.
- Verify that the name server (**iigcn**) PID number is lower than that of the database management system server (**iidbms**) process. The name server must start before the database management system server so the database management system server can register with it.
- Check your local environment and the ULTRIX/SQL symbol table using **ingprenv**. If the ULTRIX/SQL environment variable II_DBMS_SERVER is set, verify that it points to a valid server name (communications address).
- Use **iishutdown** and **iistartup** to stop and restart your database management system server.

7.6.3 After You Have Checked Everything

After you have checked each of the applicable possibilities in the preceding sections, use **iishutdown** and **iistartup** to stop and restart your installation.

Note

If the installation cannot be restarted from this point, you can shut it down again and reinitialize the log file. *This should only be done as a last resort.*

ULTRIX/SQL Startup Files A

A.1 Introduction

Each time that a user invokes the ULTRIX/SQL commands, the Terminal Monitor can read up to three different startup files. These can contain either ULTRIX/SQL commands or macro definitions. Startup files can be installation-wide, database-specific, or user-specific depending upon how they are invoked.

A.2 Installation-Wide Startup Files

The appropriate system startup files are automatically read when ULTRIX/SQL is invoked with the Terminal Monitor. These files are included with your ULTRIX/SQL installation and can be edited by the ULTRIX/SQL System Administrator to suit the requirements of your site.

The startup files created by the ULTRIX/SQL installation procedure are:

sqladmin.csh—Administrator startup file for C shell

sqluser.csh—General user startup file for C shell

sqladmin.profile—Administrator startup file for Bourne shell

sqluser.profile—General user startup file for Bourne shell

Include a *.csh file by inserting a source command line in either the ~/.cshrc or ~/.login file (whichever you use to define your path variable) after the line that defines the path variable. For example:

```
source /usr/lp/sql/sqladmin.csh
```

Include a *.profile file by inserting a source (.) command in the $HOME/.profile file, after the line that defines your PATH variable. For example:

```
. /usr/lp/sql/sqladmin.profile
```

The following list specifies the contents of each ULTRIX/SQL startup file:

- sqladmin.csh

```
#
# ULTRIX/SQL administrator's startup file for C shell login
# Include the following line in your ~/.cshrc
#
#   source /usr/lp/sql/sqladmin.csh
#
setenv II_SYSTEM /usr/lp/sql
set path=($II_SYSTEM/sql{bin,utility} $path)
```

- sqladmin.profile

```
#
# ULTRIX/SQL administrator's startup file for Bourne shell login
# Include the following line in your $HOME/.profile
#
#   . /usr/lp/sql/sqladmin.profile
#
II_SYSTEM=/usr/lp/sql
export II_SYSTEM
PATH=$II_SYSTEM/sql/bin:$II_SYSTEM/sql/utility:$PATH
```

- sqluser.csh

```
#
# ULTRIX/SQL user's startup file for C shell login
# Include the following line in your ~/.cshrc:
#
#   source /usr/lp/sql/sqluser.csh
#
setenv II_SYSTEM /usr/lp/sql
set path=($II_SYSTEM/sql/bin $path)
```

- sqluser.profile

```
#
# ULTRIX/SQL user's startup file for Bourne shell login
# Include the following line in your $HOME/.profile
#
#   . /usr/lp/sql/sqluser.profile
#
II_SYSTEM=/usr/lp/sql
export II_SYSTEM
PATH=$II_SYSTEM/sql/bin:$PATH
```

The $II_SYSTEM/sql/files/startsql file can be edited by the ULTRIX/SQL System Administrator to include ULTRIX/SQL commands that will be executed each time the Terminal Monitor is invoked.

A.3 Database-Specific Startup File

Database-specific environment variables can be set to contain the full pathname of a system startup file to be read when the ULTRIX/SQL Terminal Monitor is invoked for a particular database.

In ULTRIX/SQL, the *DBNAME*_SQL_INIT environment variable can be set to the full pathname of a file containing ULTRIX/SQL commands. The filename is specified by the user. For *DBNAME*, substitute the name of the database for which the file is to be read. The database name must be specified in uppercase.

The ULTRIX/SQL System Administrator can set these environment variables installation-wide by using the **ingsetenv** command as follows:

> **ingsetenv** *DBNAME*_SQL_INIT *path_name*

The definitions for environment variables set with the **ingsetenv** command are stored in the $II_SYSTEM/sql/files/symbol.tbl file. Type the command **ingprenv** to see the ULTRIX/SQL environment variables and their values.

These definitions can also be set locally as user-specific environment variables. A good place to set them locally is in the user's .login or .profile file. To set the database-specific environment, use one of the following commands:

C shell example:

```
setenv DBNAME_SQL_INIT path_name
```

Bourne shell example:

```
DBNAME_SQL_INIT=path_name
export DBNAME_SQL_INIT
```

A.4 User-Specific Startup File

An environment variable can be set by the user to contain the full pathname of a user-specific startup file to be read when ULTRIX/SQL is invoked with the Terminal Monitor. This startup file can be tailored by the user.

In ULTRIX/SQL, the II_SQL_INIT environment variable can be set to the full pathname of a file containing ULTRIX/SQL commands. The file name is specified by the user.

A good place to set user-specific environment variables is in the user's .login or .profile file. To set user-specific environment variables, use one of the following commands:

C shell example:

```
setenv II_SQL_INIT path_name
```

Bourne shell example:

```
II_SQL_INIT=path_name
export II_SQL_INIT
```

Authorizing User Access to ULTRIX/SQL and Databases B

B.1 The ULTRIX/SQL System Database (iidbdb)

ULTRIX/SQL maintains a special database named **iidbdb**. It is used to store information about databases and authorized ULTRIX/SQL users, and to specify the databases that can be accessed by specific users. The information in the **iidbdb** is updated by ULTRIX/SQL when a database is created or destroyed. The **iidbdb** is consulted to:

- Validate a user's request to use ULTRIX/SQL
- Validate a user's request to use a particular database
- Determine where a particular database is stored

The **iidbdb** is an ULTRIX/SQL database owned by the ULTRIX/SQL System Administrator.

The information in the **iidbdb** regarding the databases and where they are located is automatically maintained by ULTRIX/SQL. This information is updated by the **createdb** and **destroydb** commands. This information is further affected by commands that relocate tables in a database into different directories or file systems.

The information in the **iidbdb** about authorized ULTRIX/SQL users and which databases may be accessed by them is maintained by the ULTRIX/SQL System Administrator with the **accessdb** program, described in this appendix.

ULTRIX/SQL users other than the ULTRIX/SQL System Administrator can use the **catalogdb** command to view this information. This command is described in the *ULTRIX/SQL Reference Manual.*

B.2 Database Access

For user Bob to access database "xyz," at least one of the following conditions must be true:

- Bob is the Database Administrator (DBA) for "xyz."
- "xyz" is globally accessible.
- Bob has been explicitly authorized by the "xyz" DBA to use database "xyz."
- Bob has the ULTRIX/SQL superuser flag set in his user entry and uses the **-u** (user) option on the command line.

By default, databases are globally accessible when they are created. The **-p** (private) option of the **createdb** command must be used to create a private database. Operations such as changing the global access status of a database, extending a database to different locations, or authorizing a particular user to access a specific private database, may be performed by the ULTRIX/SQL System Administrator with the **accessdb** command. Authorizing a user to access a database can be accomplished by adding the user to the table of users authorized to access the database or by adding the database to the list of databases the user is authorized to access.

Remember that authorizing access to a database does not automatically authorize access to tables in the database. Only the DBA for a database may authorize user access to shared tables with the ULTRIX/SQL **grant** command.

B.3 Defining the Terminal

The **accessdb** command must be executed on a cursor-addressable terminal whose description is contained in the $II_SYSTEM/sql/files/termcap file. The environment variable TERM (or TERM_INGRES) sets the terminal definition for use by the forms utilities. Standard terminal types may be identified with the TERM variable, but some additional terminal definitions are available with the TERM_INGRES variable. For example, if you are using a VT100 terminal with function keys active in the C shell environment, you can identify your terminal, in your .login or .cshrc file, to **accessdb** by including the following command:

```
setenv TERM_INGRES vt100f
```

To identify your terminal in the Bourne shell environment, in your .profile file, include the following commands:

```
TERM_INGRES=vt100f
export TERM_INGRES
```

For more information on ULTRIX/SQL terminal types and termcap entries, refer to the *ULTRIX/SQL Reference Manual.*

B.4 Invoking accessdb

After you define your terminal, start up **accessdb** by typing the following command at the operating system level with no parameters:

```
accessdb
```

Remember that only the ULTRIX/SQL System Administrator, root, and other accounts you set up in the **iidbdb** with ULTRIX/SQL superuser privilege are allowed to use **accessdb**.

B.5 Using accessdb

Because **accessdb** is a forms-based program, it is important to understand how to enter data into forms and select menu items before using it. For more information on using forms-based programs, refer to the *ULTRIX/SQL Reference Manual.*

When **accessdb** starts up, it displays a main menu of commands. The process of using **accessdb** consists of selecting a command from this main menu, moving on to another display or menu, optionally selecting another command, and exiting by selecting the **Quit** command. The execution of most commands causes **accessdb** to display a new screen with menu items appropriate to the function chosen. To return to the main menu from any screen display, enter the **End** menu item. Other menu items may also return you to the main menu after performing their sub-function.

All menus within **accessdb** contain an entry for **Help**. If you are in doubt about the meaning of an option that you have selected, select the **Help** menu item to get a quick reminder.

B.6 Functions in accessdb

The main menu in **accessdb** contains the following commands:

Table B-1: Accessdb Functions

Command	Function
Catalog	Submenu of additional operations (discussed in Chapter 3).
Database	Summarize information about a database.
ExtendDB	Extend a database to a new volume.
LocationName	Create/Examine a *locationname*-area mapping.
User	Create/Modify/Delete an ULTRIX/SQL user or specify the databases a user may access.
Help	Print help about the top level menu.
Quit	Exit from the **accessdb** program.

B.6.1 Database Function

The **Database** function is used to examine and update information about a single database. When the **Database** option is selected from the main menu, you are prompted for the name of an existing database. After you enter the database name and press **Return**, a form describing the database is displayed or an error message is generated if the database does not exist. To return to the main menu, select the **End** menu item. Help is available by selecting the **Help** menu item. The information displayed on the form includes the fields listed in the following table.

Only the user table and the global access flag may be updated using the Database command.

Table B-2: Database Functions

Field Name	Mode	Description
Database name	read	Name of database.
Owner	read	Owner name.
Database location	read	*Locationname* where database is stored.
Checkpoint location	read	*Locationname* where database checkpoints are stored.
Journal location	read	*Locationname* where database journals are stored.
Access	read/write	Set "private" for private databases, and "public" for globally accessible databases.
Authorized users	read/write	For private databases, a list of users explicitly authorized to access database. This is a table field, which can be edited. To delete a user, move to the row and press **Return**. To add another user, move to an empty row and enter the new name.

The **Database** form allows you to change both the global access status of a database and its list of explicitly authorized users. To do so, once the database form is on the screen, edit the form to reflect the changes, then select the **Save** menu item. If you don't want to keep the changes you have made, select the **End** menu item instead.

B.6.2 LocationName Function

The **LocationName** function is used to add or examine a single *locationname.* A locationname is a label that is mapped to an ULTRIX directory.

The locationname is chosen by its creator and must follow ULTRIX/SQL naming conventions. Each locationname must be alphanumeric and begin with a letter or an underscore. The maximum length of a locationname is 24 characters. The full pathname can be a maximum of 128 characters, including slashes, and must follow the ULTRIX syntax for directory names.

Each ULTRIX/SQL installation has a set of default locationnames. These defaults are ii_database, ii_journal and ii_checkpoint. The locationnames map to directories that are values for the environment variables II_DATABASE, II_JOURNAL, and II_CHECKPOINT, respectively. These defaults are mapped to specific directory paths during the installation procedure. They *cannot* be changed.

The locationname is used by the **createdb** and **finddbs** utilities. If a locationname is not specified, the default locationname is used.

B.6.2.1 Setting Up Directories

Before you create a locationname, you must create the directory to which it points. By default, a locationname maps to a directory within the ULTRIX/SQL installation area. To point to a directory outside the ULTRIX/SQL installation area, you must specify the full pathname, beginning with the slash (/) character.

For example, to create the new_area directory within the ULTRIX/SQL installation area, type:

```
$II_DATABASE/ingres/data/new_area
```

To create the new_area directory outside the ULTRIX/SQL installation area, include the full pathname as follows:

```
/otherplace/new_area
```

The following table illustrates where ULTRIX/SQL places database files when a locationname points to an area within the ULTRIX/SQL installation.

Table B-3: Database Locations Within an ULTRIX/SQL Installation Area:

Designated *locationname*	**Designated Area Name**	**Where ULTRIX/SQL places data, checkpoints, or journals, after you set up subdirectories**
altloc2	new_area	$II_DATABASE/ingres/data/new_area $II_CHECKPOINT/ingres/ckp/new_area $II_JOURNAL/ingres/jnl/new_area

The following table illustrates where ULTRIX/SQL places database files when a locationname points to an area outside the ULTRIX/SQL installation.

Table B-4: Database Locations Not Within an ULTRIX/SQL Installation Area:

Designated ***locationname***	**Designated Area Name**	Where ULTRIX/SQL places data, checkpoints, or journals, after you set up subdirectories
altlocl	/otherplace/new_area	/otherplace/new_area/ingres/data/default /otherplace/new_area/ingres/ckp/default /otherplace/new_area/ingres/jnl/default

You must set up a directory and subdirectories in an area that will contain a new location. The examples below place only the data in the new area. To place journals or checkpoints in the new area, follow the same procedure, except substitute "jnl" or "ckp" for "data." You may place data, checkpoints, and journals in separate locations (recommended) or all in one location (not recommended).

- New area within the ULTRIX/SQL installation:

  In this illustration, the area is called new_area. $II_SYSTEM/sql/data already exists. Within this normal ULTRIX/SQL installation, you must create the new area. The new area must be owned by the ULTRIX/SQL System Administrator and must have the permissions specified below. You can follow the script below to create the new area.

  Log in as the ULTRIX/SQL System Administrator, **ingres**. Start from the $II_SYSTEM/sql/data directory.

  ```
  mkdir new_area
  chmod 777 new_area
  ```

- New area in a completely different location from the ULTRIX/SQL installation:

  In the following illustration, the area that will hold the new location is called /otherplace/new_area. Let's say that /otherplace already exists. You will create the directory new_area. Within that area, you must create the subdirectory structure ingres/data/default. The new_area directory and each of the subdirectories must be owned by the ULTRIX/SQL superuser and must have the permissions specified below. You can follow the script below to create the proper subdirectory structure in your new area.

  Log in as the ULTRIX/SQL System Administrator, **ingres**. Start from the /otherplace directory.

  ```
  mkdir /otherplace/new_area
  mkdir /otherplace/new_area/ingres
  mkdir /otherplace/new_area/ingres/data
  mkdir /otherplace/new_area/ingres/data/default

  chmod 755 /otherplace/new_area
  chmod 755 /otherplace/new_area/ingres
  chmod 700 /otherplace/new_area/ingres/data
  chmod 777 /otherplace/new_area/ingres/data/default
  ```

B.6.2.2 Creating Location Names and Mapping them to Directory Paths

After setting up the area with subdirectories, you are ready to add the locationname. Select the **LocationName** function from the main menu of **accessdb**. The system prompts you for a locationname. Enter a new location that is undefined by ULTRIX/SQL. You will be asked if you want to create a new location. Say **y** (yes) and a blank form appears with the default values filled in. Fill in the appropriate data fields on the form.

The new locationname appears on the top of the form. Below it, a table-field displays the databases currently using that location. (This should be empty now.) Fill in the **Area** field with the full pathname for the new area that locationname represents in your environment. Leave off the ingres/data/default part of the pathname; this is assumed, and will cause errors if noted here. Next, fill in the usage permissions. The privileges are as follows:

- **Databases** allows database relations to reside on the specified locationname. The default is "y(yes).
- **Journals** allows journals to reside on the specified locationname. The default is **n** (no).
- **Checkpoints** allows checkpoints to reside on the specified locationname. The default is **n** (no).

When the form accurately describes the new locationname, select the **Save** menu item. If you decide not to create the locationname, select the **End** menu item.

On each execution of the **LocationName** function, you may add a new locationname.

B.6.2.3 Examining an Existing Locationname

When the **LocationName** function is selected from the main menu, the system prompts you for a locationname. Enter an existing location. A form appears that displays the current attributes of the locationname. You cannot change anything on this form.

B.6.2.4 Modifying a Locationname

Once the form that created the locationname has been exited, only the usage permissions can be modified. The mapping between the locationname and area is permanent.

B.6.3 ExtendDB Function

The **ExtendDB** function is used to extend a database to other locations. To locate tables outside the default area or the area corresponding to the locationname specified on the **createdb** command line, the database must be extended to a previously defined location.

Select the **ExtendDB** menu item from the main menu. When the system prompts for the login of the DBA, enter the login of the DBA of the database to be extended. A form then displays two tables: one listing existing locationnames and database names, and one in which you can add new database extensions. The names currently in the first table are those databases that have been extended to the specified locationname. One database may be extended to many locations.

For a database to be extended, both the database and locationname must exist. The database must be owned by the specified DBA, and the corresponding locationname must be available for databases as defined in the **LocationName** function in the main menu. To extend a database to a location, move the cursor to the table of new database locations, and enter the database name and the corresponding locationname in the proper columns. When you are done extending databases for a particular DBA, select the **Save** menu item. The data in the table is then checked. If any of the data in the table are invalid, or if a database extension or restriction is not allowed, the cursor is positioned on the row containing the offending data. You can then correct the data and select the **Save** menu item. When the changes are accepted, you are returned to the main menu.

B.6.4 User Function

The **User** function is used to add, modify, or delete access to ULTRIX/SQL by a user and to identify the databases that a user may access. The **User** function may be used to display an existing user's authorization information. Each implementation of the **User** function operates on a single user.

When the **User** function is selected from the main menu, you are prompted for a user name. The name you enter may be either the login of an existing ULTRIX/SQL user or the login of a user to be added. After entering the user name, a form describing the user is displayed.

B.6.4.1 Adding a New User

If you enter a new user name, you are asked to verify that you really intend to create a new user. If your response is positive, a form with the new user name and default permissions is displayed.

To create a new user, you must first know the ULTRIX login for the user you wish to add to the user list.

Once you have the login, the user may be created. First, select the **User** function in the main menu, responding with the login when prompted. After entering the login, respond with **y** (yes) when asked if you really intend to create the user. Next, fill in the user information on the form that appears. The privileges are as follows:

- **Create database** permission allows a user to create new databases. The default is **y** (yes).

- **Update system catalog** permission allows a user to directly update system catalogs (attribute, relation, and so on) using ULTRIX/SQL. This is rarely needed. The default is **n** (no).

- **Set trace flag** permission allows a user to set the debugging trace flags within ULTRIX/SQL. The default is **n** (no).

- **ULTRIX/SQL superuser** permission allows a user to impersonate any other user in the system or to run the **accessdb** program. Few users other than root and **ingres** should be allowed this permission. The default is **n** (no).

Change the default privileges provided by entering a **y** (yes) or **n** (no) next to the privilege shown. The first table field shown, of the databases owned by the user, is a read-only table field and is empty for new users because a new user does not own any databases. The other table field, of the databases that the user may access, can be filled in with the names of private databases you wish the user to access. Enter a database name, press **Return**, enter another database, name and so on. When finished, press the **Menu** key to bring up the menu and select the **Save** menu item to create the new user. If you decide not to create the user after filling in part of the form, select the **End** menu item instead of the Save item.

B.6.4.2 Modifying an Existing User

To examine or modify an existing user's permissions, select the **User** function from the main menu and respond with the user's login when prompted. Edit the form that appears to reflect the changes you wish, including changes to the set of databases the user is explicitly authorized to access in the May Access table field. The Owns table field is read-only, because only the called **createdb** and **destroydb** commands can be used to add and delete databases from that list. Select the **Save** menu item to save the changes. If you only want to examine the user's permissions or decide not to save the changes, select the **End** menu item instead.

B.6.4.3 Deleting a User

To delete the authorization entry for an ULTRIX/SQL user, select the **User** function from the main menu and respond with the user's login when prompted. When the user's information form appears on the screen, select the **Delete** menu item. If you decide not to delete the user, select the **End** menu item instead.

Note

Deleting a user is not permitted if that user owns any databases.

B.6.5 Catalog Function

In addition to establishing database extensions, locationname-area mappings and ULTRIX/SQL users, the **accessdb** utility enables you to view current database access entries. The **Catalog** operation in the main **accessdb** menu calls a submenu of functions that displays the data in separate frames.

The **Catalog** submenu contains the following commands:

Table B-5: Catalog Functions

Command	Description
Databases	Displays a read-only table of databases.
LocationNames	Displays the read-only table of locationname-area mappings.

Command	Description
Users	Displays a read-only table of users.
Help	Provides information about the operations in this menu.
End	Returns to the **accessdb** main menu.

B.6.5.1 Databases Catalog Function

The **Databases** function displays a read-only table field containing information for each ULTRIX/SQL database. For each database, the database name, owner, database type, and global access flag are displayed. To change information about a database, return to the **accessdb** main menu and select the **Database** function. To scroll up and down in the table field, use the techniques described in the *ULTRIX/SQL Reference Manual.*

To return to the **Catalog** menu, select the **End** menu item. To return to the main menu, select **End** again.

B.6.5.2 LocationNames Catalog Function

The **LocationNames** function displays a read-only table field containing each locationname-area mapping. Scroll through the table field to view its entire contents.

To return to the main menu, select the **End** menu item.

B.6.5.3 Users Catalog Function

The **Users** function displays a read-only table field containing information about each ULTRIX/SQL user. For each user, the login and user permissions are displayed. The display is a table and may be scrolled up and down to view its full contents. You cannot modify this information. To change information about a user you must return to the **accessdb** main menu and select the **User** function.

To return to the main menu, select the **End** menu item.

B.7 Summary of accessdb

The **accessdb** command can *only* be used by the ULTRIX/SQL System Administrator or ULTRIX/SQL superusers to modify, add, delete or list ULTRIX/SQL users and database access permissions. It uses a forms-based interface, so **accessdb** must be run on a supported video terminal. Other ULTRIX/SQL users cannot use **accessdb**, but they may use the **catalogdb** command, which is described in the *ULTRIX/SQL Reference Manual.* The **catalogdb** command displays *in read-only mode* information similar to **accessdb.**

ULTRIX/SQL Environment Variables C

C.1 Overview

You must define environment variables for your ULTRIX/SQL installation. Certain installation-wide parameters should only be defined by the ULTRIX/SQL System Administrator in symbol tables. Other variables can be defined or redefined by individual users to customize their local ULTRIX/SQL environment.

C.2 Setting Installation-Wide Environment Variables

Environment variables are made installation-wide by the ULTRIX/SQL System Administrator, who can use the **ingsetenv** command to write them to the ULTRIX/SQL symbol table.

For example, to set the ING_EDIT environment variable for an installation to the vi editor, you would log in as the ULTRIX/SQL System Administrator and type:

```
ingsetenv  ING_EDIT  /usr/ucb/vi
```

You can display all installation wide variables by typing:

```
ingprenv
```

The variables and the values they are set to in the symbol table will be displayed.

C.3 Setting User-Defined Environment Variables

Some ULTRIX/SQL environment variables may be set or reset by individual users in their local environment using ULTRIX commands. For example, one variable often set in the user's environment is TERM_INGRES. It defines the ULTRIX/SQL termcap definition to be used by the Forms System. To reset it in your local environment use one of the following:

C shell:

```
setenv  TERM_INGRES  vt100f
```

Bourne shell:

```
TERM_INGRES=vt100f
export  TERM_INGRES
```

Users can display the values set in their environment with the **printenv** command:

```
printenv
```

Environment variables set in a user's local environment supersede the variables set in the symbol table. A good place to set user defined variables is the user's .login or .profile file.

C.4 Environment Variable List

Following are the environment variables you can define for your ULTRIX/SQL installation:

- *DBNAME*_SQL_INIT is an environment variable that affects any user who connects to the specified database through the Terminal Monitor. This environment variable may be set installation-wide or locally. *DBNAME*_SQL_INIT points to **set** commands within a file. This environment variable applies only to the ULTRIX/SQL Terminal Monitor. Setting this variable is equivalent to users executing **\i filename** in the Terminal Monitor each time they connect to *DBNAME*.

  The syntax for defining this environment variable is:

  C shell:

  ```
  setenv DBNAME_SQL_INIT "path_to_file"
  ```

  Bourne shell:

  ```
  DBNAME_SQL_INIT="path_to_file"
  export DBNAME_SQL_INIT
  ```

  The *DBNAME* must be in uppercase. The file contains the **set** commands. If you have several **set** commands, separate the commands with semicolons (;). You must end the entire file with \g.

  To affect both the Terminal Monitor and applications, use the more general environment variable ING_SET*DBNAME*.

- II_CHECKPOINT is set to the full pathname for the default checkpoint location, ii_checkpoint. This variable is set by the ULTRIX/SQL System Administrator during the **iibuild** procedure, and it may not be changed, even during installation updates. Specific databases may designate alternate locations for checkpoints as a parameter to the **createdb** command.

- II_CONFIG is set to the full pathname of the ULTRIX/SQL files directory during the ULTRIX/SQL installation procedure.

- II_DATABASE is set to the full pathname for the default database location, ii_database. This variable is set by the ULTRIX/SQL System Administrator during the **iibuild** procedure and it may not be changed, even during installation updates. Specific databases may designate alternate locations as a parameter to the **createdb** command.

- II_DATE_FORMAT defines the format style for date output. Default is the US value with an output format of *dd-mmm-yyyy*. Default legal input formats for dates are:

  ```
  dd-mmm-yyyy
  mm/dd/yy
  mmddyy
  ```

  If the environment variable is set, it replaces one of these formats with an alternative format. Table C-1 lists the valid settings for II_DATE_FORMAT.

Table C-1: II_DATE_FORMAT Settings

Value	Alternative Input Format	Replaces Input
US	*dd-mmm-yyyy*	
ISO	*yymmdd*	*mmddyy*
SWEDEN or FINLAND	*yyyy-mm-dd*	*dd-mmm-yyyy*
MULTINATIONAL	*dd/mm/yy*	*mm/dd/yy*
GERMAN	*dmmyy*	*xxxxx*
	ddmmyy	*xxxxxx*
	dmmyyyy	*xxxxxxx*
	ddmmyyyy	*xxxxxxxx*

  In all cases, the *alternative* input format becomes the *default* output format. The three US default input formats listed previously are unaffected by alternative settings and remain valid date input formats.

- II_DBMS_SERVER points to the database management system server to which the user's process will automatically connect.

- II_DECIMAL specifies the character used to separate fractional and nonfractional parts of a number. The default value is the period (.), as in 12.34. Alternatively, the comma (,) can be used, as in 12,34. Only the period and the comma are allowed.

- II_ERSEND is set during the ULTRIX/SQL installation procedure to the full pathname of the errlog.log file.

- II_FILES is set to the full pathname of the ULTRIX/SQL files directory during the ULTRIX/SQL installation procedure.

- II_GCN*xx*_PORT contains the connect address of an installation's name server, where *xx* is its two letter installation code (II_INSTALLATION).

- II_INSTALLATION is a two-character code used to define a particular ULTRIX/SQL installation. It should be defined at the installation level using the **ingsetenv** command. It should *never* be defined at the user level. This variable is set by the ULTRIX/SQL System Administrator during the **iibuild** procedure.

- II_JOURNAL is set to the full pathname for the default journal location, ii_journal. This variable is set by the ULTRIX/SQL System Administrator during the **iibuild** procedure and it may not be changed, even during installation updates. Specific databases may designate alternate locations for journals as a parameter to the **createdb** command.

- II_LG_MEMSIZE is set to a value that will be used for the size, in bytes, of the locking and logging shared memory segment created when **csinstall** is called by **iistartup**. The size of the segment is fixed until the next time **csinstall** is executed. II_LG_MEMSIZE is calculated from the locking and logging parameters you selected for your ULTRIX/SQL installation, when **iistartup** calls **iirun** to start the recovery process. The value of II_LG_MEMSIZE should never be set below the default of approximately 200K.

  In the current release of the system, the size of the LG/LK shared memory segment defaults to approximately 200K. This segment's size is fixed once it is created by **csinstall**. Its size may increase the next time **csinstall** is called. When the **dmfrcp** process starts up, it reads the locking and logging parameters that the user has specified during the **iibuild** and calculates the maximum amount of memory that may be needed to support those parameters. The RCP sets a system level environment variable II_LG_MEMSIZE to this value (represented in number of bytes). Until shutdown, the RCP will continue to use the default size segment, possibly returning "out of (locking/logging) memory" error conditions on some queries. Subsequently, whenever the system is restarted (using **iistartup** and thus calling **csinstall** to recreate the locking/logging segment) the LG/LK segment will be created with whatever size the variable II_LG_MEMSIZE is set to. II_LG_MEMSIZE should never be set below the default of approximately 200K.

- II_LOG_FILE is set to the parent directory of ingres/log/ingres_log. It determines the location of the installation-wide logging file. The **iibuild** procedure prompts you for this.

- II_MSG_TEST is set to "false" during the ULTRIX/SQL installation procedure, and should not be changed.

- II_MONEY_FORMAT defines the format of money output. You can change output by setting the variable to a string with two symbols separated by a colon (:). The symbol to the left of the colon must be "L" for a leading currency symbol, or "T" for a trailing symbol. To the right of the colon, put the currency symbol you want prepended or appended to the amount. Examples follow:

Environment Variable Definition	Result
L:$	$100
T:DM	100DM
T: FF	100 FF

- II_MONEY_PREC shows the number of digits of precision to be used in the default representation of money data. The default is two digits (for decimal currency). Valid choices are 0, 1 and 2.

- II_NUM_SLAVES specifies the number of slave processes that a database management system server will create to do disk operations. The default is two, but systems with many or faster disk drives will want to increase the number of slave processes. For example,

  ```
  ingsetenv  II_NUM_SLAVES  4
  ```

  allows each new server that is started to have four slaves. You must stop and restart servers to reset the number of slaves. The maximum supported value is 10, and the minimum is 0. Setting this variable to 0 will degrade performance substantially.

- II_PATTERN_MATCH determines the set of pattern matching characters for layered product qualification functions. These can be set to be ULTRIX/SQL-like or not according to the following parameters.

  Pattern qualification functions use the percent (%), underscore (_), left bracket ([), and right bracket (]) pattern matching characters. The percent character is equivalent to the ULTRIX/SQL asterisk (*) and the underscore is equivalent to the ULTRIX/SQL question mark (?).

- II_SQL_INIT may be set locally to the full pathname of a file containing ULTRIX/SQL commands. When a user with this variable set connects to the ULTRIX/SQL Terminal Monitor, the commands in the named file are processed. Setting this variable is equivalent to executing **\i file** in the Terminal Monitor each time a user connects to a database.

- II_SYSTEM specifies the parent directory of sql, which is the root of the ULTRIX/SQL installation. This environment variable should not be changed unless ULTRIX/SQL is reinstalled. The value of II_SYSTEM is normally /usr/lp/sql.

- II_TEMPLATE is set during the ULTRIX/SQL installation procedure to the full pathname of the database "template" directory.

- II_TEMPORARY can be set to an area that will supersede the default area, specified by II_SORT, where ULTRIX/SQL creates temporary sort files. First, you must create a new location using **accessdb**. Finally, set II_TEMPORARY to the name of the new location. Sort files are created during the processing of ULTRIX/SQL commands like **modify**, and **create index**.

- II_TERMCAP_FILE specifies an alternative termcap file. This file *must* be in termcap file format.

- II_THOUSANDS is set to a one-character symbol indicating the separator between thousands in numbers. Choices are the comma (,) which is the default, or the period (.).

- II_TMPDIR is used by ULTRIX/SQL to locate temporary sort files. Sort files are created during the processing of many ULTRIX/SQL commands: queries (for instance, retrieval with joins, **sort by** clauses), the **modify** and **create index** commands. II_TMPDIR is not for temporary files created by the ULTRIX/SQL applications. Use II_TEMPORARY to locate these files elsewhere.

- ING_EDIT specifies the default editor spawned by various editor commands. The default for the entire installation is set during the ULTRIX/SQL installation procedure. Users can also set this in their local environment.

- ING_SET may be set installation wide or locally, to a quoted string. The string must be 64 characters or less, or it will be invalid. It may contain either **set** commands separated by semicolons, or the word **include** followed by the full pathname for a file of **set** commands. If the variable is defined, the **set** commands are executed when any ULTRIX/SQL interactive utilityconnects to a database management system server. For example:

  C shell:

  ```
  setenv ING_SET "set qep;set joinop notimeout"
  setenv ING_SET "include /usr/abc/sqlsetfile"
  ```

  Bourne shell:

  ```
  ING_SET="set qep;set joinop notimeout""
  export ING_SET
  ```

- ING_SET_*DBNAME* is an environment variable that affects any user who connects to the specified database through an application or interactive utility. This environment variable may be set installation-wide or locally. ING_SET_*DBNAME* points to set commands within a quoted string or within a specified file. The quoted string must be 64 characters or fewer.

  The syntax for defining this environment variable is:

  C shell:

  setenv ING_SET_*DBNAME* "set *command*"

  setenv ING_SET_*DBNAME* "set *command*; set *command*"

  setenv ING_SET_*DBNAME* "include *path_to_file*"

Bourne shell:

> **ING_SET**_*DBNAME*="**set** *command*"
> **export ING_SET**_*DBNAME*
>
> **ING_SET**_*DBNAME*="**set** *command*; **set** *command*"
> **export ING_SET**_*DBNAME*
>
> **ING_SET**_*DBNAME*="**include** *path_to_file*"
> **export ING_SET**_*DBNAME*

The *DBNAME* must be in upper case. The file can contain the **set** commands. If you have more than one **set** command in the file, use a semicolon (;) between the commands. Also, each set command must be on a separate line in the file.

- ING_SHELL, if defined, contains the pathname of the shell that ULTRIX/SQL front ends use when the **shell** operation is invoked.

- ING_SYSTEM_SET may be set installation-wide, to a quoted string. The string must be 64 characters or less, or it will be invalid. It may contain either **set** commands separated by semicolons, or the word **include** followed by the full pathname for a file of **set** commands. If the variable is defined, the **set** commands are executed when an interactive utility connects to a database management system server.

- TERM is the terminal description for the terminal. This environment variable is most conveniently defined in the user's .login or .profile file.

- TERM_INGRES contains the terminal designation for the terminal. This environment variable is most conveniently set in a .login or .profile file. TERM_INGRES takes precedence over TERM in defining terminal type to ULTRIX/SQL, and allows TERM to be defined differently for use by other ULTRIX programs.

ULTRIX/SQL System Recovery D

D.1 Overview

The recovery tools **rollforwarddb** and **finddbs** are provided by ULTRIX/SQL to recover from system failures that can lead to database or installation corruption. This section briefly discusses when to use each of these tools and describes the **finddbs** command. See the *ULTRIX/SQL Reference Manual* for information about **rollforwarddb** and additional information about **finddbs.**

D.2 Deciding Which Recovery Method to Use

The **rollforwarddb** command can regenerate a database from a static backup called a checkpoint. If your installation keeps dynamic records of changes to the database (journals), they can be used to restore the database up to the time of the system failure.

If a system failure causes corruption of the information in the **iidbdb,** the **finddbs** command can be used to recover the locations of databases. This information is stored in the **iidbdb** system catalog, **iidatabase.** The **finddbs** command allows the user to search any directories for databases and enter them into the **iidatabase** table.

D.3 Using finddbs

The syntax of the **finddbs** command is as follows:

finddbs [-a | -r] [-p]

Parameter	Description
-a	Runs **finddbs** in analyze mode (the default). Scans the **iidbdb,** but simply writes intended updates to the terminal, not to the **iidbdb.** Use this option if you suspect the **iidbdb** database catalog is out of order.
-r	Runs **finddbs** in replace mode, actually inserting the locations of databases found by the program into the **iidbdb.**
-p	Makes all located databases private, except for the **iidbdb.** By default, the **-r** parameter makes all databases public.

The **finddbs** command is intended to help ULTRIX/SQL recover when the **iidbdb** has been destroyed. Only the ULTRIX/SQL System Administrator can use it. The **finddbs** command runs in analyze mode, which is the default, or replace mode. Analyze mode is selected with the **-a** option, while replace mode is selected with the **-r** option. Analyze mode is a read-only mode that informs you about possible errors in the **iidbdb iidatabase** table with a new table formed by scanning the ULTRIX/SQL directories on a set of devices for databases. The use of replace mode is not recommended unless the **iidbdb iidatabase** table is in error.

When **finddbs** starts, it first builds a list of found databases. The list is formed by scanning the $II_SYSTEM/sql/data/default directory. You are then prompted for any other directories to search. If you respond with a directory name not preceded by a slash, **finddbs** looks in $II_SYSTEM/sql/data/*name*, where *name* is the name you specify. If you specify a directory name beginning with a slash, **finddbs** looks in /*name*/sql/data/default. Enter an empty line to exit this scan phase. It is important to search all directories that contain databases. If your installation creates all its databases in the default directory, then the default search will suffice. If non-default directory names are specified on the **creatdb** command line at your installation, or you have extended databases to alternate locations *using* **accessdb**, then you must specify all such directory names so that **finddbs** can find the databases they contain.

Use the analyze mode of **finddbs** if you suspect the **iidatabase** table of the **iidbdb** is in error. The output of analyze mode consists of two lists. The first lists names of databases present on disk but not contained in the **iidatabase** table. If analyze mode reports differences between the found database list (built during the scan phase) and the **iidbdb iidatabase** table, you may decide to run **finddbs** in replace mode.

The **finddbs** command in replace mode replaces the contents of the **iidbdb iidatabase** table with a new table consisting of the databases found during the directory scan phase. Before running **finddbs** in replace mode, you should first run it in analyze mode to see what changes would be performed by replace mode. By default, replace mode causes all databases to be made globally accessible. The **-p** option causes all databases to be made private, except for the **iidbdb.**

Examples of the **finddbs** command follow.

Run finddbs in analyze mode to examine directories and iidatabase table in the iidbdb:

```
finddbs -a
```

Replace the contents of the iidatabase system catalog with databases found running this command.:

```
finddbs -r
```

In both cases, **finddbs** prompts for the directories to check for ULTRIX/SQL databases.

Caution

When the **finddbs** command is run in replace mode, the entire contents of the **iidatabase** table are destroyed and replaced. All directories containing databases must be scanned, since only the directories which are scanned will have their databases included in the new table.

Commands for ULTRIX/SQL System Administration E

E.1 Introduction

This appendix provides quick reference information for ULTRIX/SQL operating system commands used by the ULTRIX/SQL System Administrator.

E.2 csinstall

E.2.1 Purpose

Installs shared memory and semaphore resources.

E.2.2 Syntax

csinstall

E.2.3 Description

The **csinstall** command allocates ULTRIX shared memory and semaphore resources for use by ULTRIX/SQL.

It uses the value stored in the ULTRIX/SQL environment variable II_LG_MEMSIZE to install ULTRIX/SQL shared memory and semaphore resources. If II_LG_MEMSIZE is not set in the ULTRIX/SQL symbol table, the size of the locking and logging facilities' shared memory segment defaults to approximately 200K. This memory segment is created when **csinstall is called** and its size is fixed until the next time **csinstall** is called. II_LG_MEMSIZE is set at **dmfrcp** startup. It reads the locking and logging parameters that you specified during configuration and calculates the maximum amount of memory needed to support them. It sets II_LG_MEMSIZE to that amount in bytes.

The **csinstall** command is run during **iistartup** if you choose to reconfigure shared memory and semaphore resources using the **-init** option. It can also be run interactively by the ULTRIX/SQL System Administrator.

E.3 cscleanup

E.3.1 Purpose

Deallocates ULTRIX/SQL shared memory and semaphore resources.

E.3.2 Syntax

cscleanup

E.3.3 Description

The **cscleanup** command deallocates the ULTRIX shared memory and semaphore resources that were allocated by **csinstall** for use by ULTRIX/SQL. This command is called by **iishutdown.** It can be run interactively by the ULTRIX/SQL System Administrator.

E.4 csreport

E.4.1 Purpose

Displays shared memory and semaphore information.

E.4.2 Syntax

csreport

E.4.3 Description

The **csreport** command displays shared memory and semaphore information for your installation.

E.5 iibuild

E.5.1 Purpose

Initializes and starts up ULTRIX/SQL.

E.5.2 Syntax

iibuild

E.5.3 Description

The **iibuild** utility manages the initialization and startup of your ULTRIX/SQL system. You should use this command only during the ULTRIX/SQL initialization procedure. During **iibuild** you will configure your ULTRIX/SQL installation and start up your installation. If you are updating an existing ULTRIX/SQL installation, it must be shut down before you run **iibuild**. Shut it down using **iishutdown**. The **iibuild** command can be used only by the ULTRIX/SQL System Administrator.

E.6 iimonitor

E.6.1 Purpose

Monitors and administers database management system servers.

E.6.2 Syntax

iimonitor *server_name*

E.6.3 Description

The **iimonitor** utility is an operating system level utility that allows an ULTRIX/SQL System Administrator to monitor the operation of a database management system server or to shut down a server. In addition, the System Administrator can use **iimonitor** to monitor or shut down a particular server session.

E.7 iinamu

E.7.1 Purpose

Monitors and administers the name server.

E.7.2 Syntax

iinamu

E.7.3 Description

You can use the Name Service Management Utility (**iinamu**) to display database management system server information and administer the name server. Only an ULTRIX/SQL superuser may execute the administrative options, such as adding or deleting entries, or stopping the **iigcn** process.

When you see the IINAMU> prompt, enter one of the following commands:

Command	Function
show [*svr_type*]	Shows the list of currently registered servers. *svr_type* can be INGRES or COMSVR. INGRES is the ULTRIX/SQL database management system server type and the default. COMSVR is the GCC ULTRIX/SQL NET process type. The following is an example of **show** command output: `INGRES * 1201 INGRES * 1243` The first column is the server type, the second is a list of databases registered to be served by this database management system server, and the third is the GCF_ADDRESS. The database name entry "*" means that the server has registered to service requests for any database. The GCF_ADDRESS column contains the GCF address for access to this server.
add *svr_type obj_name gcf_address*	Manually adds to the list of registered servers. This command may only be run by an ULTRIX/SQL superuser.
delete *svr_type obj_name gcf_address*	Manually deletes a server from the list of registered servers. This command may only be run by an ULTRIX/SQL superuser.
stop	Stops the GCF name server. This is the correct way to stop the GCN. If GCN is stopped while database management system servers are running
quit	Quits the **iinamu** utility.

E.8 iirun

E.8.1 Purpose

The **iirun** command invokes a utility to start up the recovery or archiver process.

E.8.2 Syntax

iirun dmfrcp | dmfacp

E.8.3 Description

The **iirun** program starts up the recovery or archiver process for your ULTRIX/SQL installation. It is called by **iistartup**. It can also be run interactively, but only by the ULTRIX/SQL System Administrator.

The options for **iirun** have the following meanings:

Option	Function
dmfrcp	Starts the recovery process.
dmfacp	Starts the archiver process.

E.9 iirungcc

E.9.1 Purpose

Starts up the communications server, **iigcc**.

E.9.2 Syntax

iirungcc

E.9.3 Description

The **iirungcc** command invokes a program that starts up the communications server, **iigcc**.

E.10 iirungcn

E.10.1 Purpose

Starts up the name server.

E.10.2 Syntax

iirungcn

E.10.3 Description

The **iirungcn** command invokes a program that starts up the name server, **iigcn**.

E.11 iishutdown

E.11.1 Purpose

Shuts down an ULTRIX/SQL installation or its servers.

E.11.2 Syntax

iishutdown [**-B**] [-s] [n]

E.11.3 Description

The **iishutdown** command shuts down your entire ULTRIX/SQL installation, or parts of it, for reconfiguration or system shutdown. It can shut down servers, as well as the archiver and recovery processes, and deallocate shared memory. You will be prompted by **iishutdown** for the elements of your installation that you want to shut down. This command can be used only by the ULTRIX/SQL System Administrator.

The options for **iishutdown** have the following meanings:

Option	Function
-B	Shutdown without broadcasting a shutdown message.
-s	Shutdown without waiting for response to the shutdown message.
-n	Shutdown immediately rather than waiting 30 seconds.

E.12 iistartup

E.12.1 Purpose

Starts up an ULTRIX/SQL installation or its servers.

E.12.2 Syntax

iistartup [**-init**] [**-B**] [-s] [$II_SYSTEM]

E.12.3 Description

The **iistartup** commmand initializes system shared memory resources as necessary and starts up ULTRIX/SQL name server, communications servers, recovery and archiver daemons, and database management system servers. Use **iistartup** to start up all the elements of your ULTRIX/SQL installation in the correct sequence. Subsequent to installation startup use **iistartup** to configure and start up additional database management system servers. This command can only be used by the ULTRIX/SQL System Administrator.

If you use **iistartup** without command line options, it will not prompt you for configuration changes. Use this syntax when no reconfiguration is desired. To start your installation automatically each time the system is rebooted, include the command **iistartup** $II_SYSTEM in your ULTRIX system startup file. This file is usually /etc/rc.local. It will use the installation configuration information saved from the last time **iistartup -init** was run. It is an error to run **iistartup** without having first run **iistartup -init**.

To reconfigure database management system server parameters use the **-init** option. Selected database management system server parameters are prompted for and the latest configuration is saved in $II_CONFIG/rundbms.opt. To configure the database management system server with parameters that are not prompted for by **iistartup -init**, edit this file. Locking and logging facility configuration information is saved in $II_CONFIG/rcp.par. If your installation has ULTRIX/SQL NET, **iigcc** saved information in $II_CONFIG/gcc.opt.

The options and parameter names for **iistartup** have the following meanings:

Parameter	Function
-init	Indicates to **iistartup** that you want to specify or change configuration parameters for the database management system server and recovery system before starting up these systems. This flag is used by **iibuild** during the initialization procedure to ensure that the ULTRIX/SQL System Administrator specifies configuration parameters at installation time. It may also be used interactively if you want to change configuration parameters, reinitialize the log file, or reinstall shared memory.
-B	Prevents a startup message from being broadcast.
-s	Startup without waiting for a response to the startup message.

Parameter	Function
$II_SYSTEM	Specifies the value of II_SYSTEM for your installation. This option must be used if the ULTRIX/SQL environment variable $II_SYSTEM is not set in the local environment.

E.12.4 Examples

Start up your installation interactively using the configuration option, where $II_SYSTEM is set to /usr/lp/sql:

```
iistartup  -init  /usr/lp/sql
```

Start up your installation automatically by including **iistartup** in the /etc/rc.local or other boot script. For example:

```
su ingres -c "/usr/lp/sql/utility/iistartup /usr/lp/sql"
```

E.13 lockstat

E.13.1 Purpose

Displays locking status.

E.13.2 Syntax

lockstat

E.13.3 Description

The **lockstat** command displays locking status information for your ULTRIX/SQL installation.

E.14 logstat

E.14.1 Purpose

Displays logging status.

E.14.2 Syntax

logstat

E.14.3 Description

The **logstat** command displays information about the logging system.

E.15 rcpconfig

E.15.1 Purpose

Configures or shuts down the logging and locking systems.

E.15.2 Syntax

rcpconfig -config | -init | -shutdown | -imm_shutdown

E.15.3 Description

The **rcpconfig** command can configure or shutdown the logging and locking systems. It is called by **iistartup** and **iishutdown** to configure or shut down the archiver and recovery processes for your ULTRIX/SQL installation. This command can be used interactively only by the ULTRIX/SQL System Administrator.

The options for **rcpconfig** have the following meanings:

Option	**Function**
-config	Configures the logging/locking systems.
-init	Configures the logging/locking systems and erases the log file.
-shutdown	Shuts down the logging/locking systems. The ULTRIX/SQL system will refuse connections and transaction processing. It will let currently executing transactions finish and then shut down the recovery and archiver processes.
-imm_shutdown	Shuts down the logging/locking systems immediately. It executes an immediate stop on pending transactions and shuts the recovery and archiver processes down. Use this flag *only* if it is *critical* to shut down the ULTRIX/SQL recovery system without waiting for pending transactions to finish.

E.16 syscheck

E.16.1 Purpose

Checks your ULTRIX system configuration for memory, semaphores, and swap space.

E.16.2 Syntax

syscheck

E.16.3 Description

The **syscheck** command is called by **iistartup** to check your ULTRIX system configuration for the minimum memory, semaphores and swap space recommended to support your ULTRIX/SQL installation. You can use **syscheck** interactively to display minimum system recommendations and to check your systems resources before reconfiguring. This command can be used only by the ULTRIX/SQL System Administrator.

Note

> The **syscheck** command requires that the **ingres** user has **kmem** group privileges to operate properly.

Running ULTRIX/SQL Under Network File System F

F.1 Introduction

The following appendix assumes that you are familiar with the basics of mounting file systems in an environment that provides Network File System (NFS) services.

As the ULTRIX/SQL System Administrator, you should understand the following issues and read the scenarios below before installing ULTRIX/SQL in an NFS environment.

Note

> ULTRIX/SQL may be installed into and used from an environment managed by Diskless Management Services (DMS), which indirectly mounts ULTRIX/SQL files through NFS. This appendix is intended for the ULTRIX/SQL System Administrator who who wants to control NFS features directly rather than use them through DMS.

ULTRIX/SQL provides its own concurrency control services to permit multi-user access to database tables. It is important to know that concurrency control is bound to the shared memory associated with a single processor. The ULTRIX/SQL database management system lock manager will provide concurrency control for multiple users running on a single processor, but there is no provision within the database management system server process (**iidbms**) for synchronized locking of file resources between multiple network nodes.

If ULTRIX/SQL data resides on NFS disk partitions that are exported to other network nodes capable of running the ULTRIX/SQL database management system server process, transaction concurrency may be compromised. In this environment the potential exists for two or more ULTRIX/SQL database management system server processes, using different shared memory, to access the same database independently. Then, corruption of the user tables and system catalogs is possible. To prevent this, please use one of the following scenarios as a guideline for installing ULTRIX/SQL on multiple nodes that share Network File Systems.

F.2 Configuration Scenarios

The following two scenarios have been described to assist the ULTRIX/SQL System Administrator at configuration startup time. Both are typical of an ULTRIX/SQL installation in an environment that provides NFS services.

The ULTRIX/SQL $II_SYSTEM environment variable is used throughout these scenarios to describe an ULTRIX file system path specification. When specifying the disk partitions to be exported, mounted, and so on, you will need to replace $II_SYSTEM with the physical path specification.

F.2.1 Scenario 1 (Diskless Client)

This scenario describes a shared ULTRIX/SQL installation, where the ULTRIX/SQL processing is shared by the NFS server and clients using ULTRIX/SQL. The ULTRIX/SQL applications may be executed on both the server node and the client nodes. The database management system server and other installation related processes are executed only on the server node.

The ULTRIX/SQL system software distribution is installed on the NFS server. The binaries, libraries, and auxiliary files are shared. The $II_DATABASE, $II_CHECKPOINT, and $II_JOURNAL locations are shared. All ULTRIX/SQL server processes (**iidbms, iigcn, iigcc**), and the archiver (**dmfacp**) and recovery (**dmfrcp**) processes are executed on the NFS server. Concurrency management is maintained only by the NFS server node. The client's ULTRIX/SQL application processes are connected to a database management system server process, on the NFS server, by using ULTRIX/SQL NET.

Note

> It is important for the resolution of $II_SYSTEM to be the same for both the NFS server and the clients. You may do this with symbolic links.

Server characteristics:

- $II_SYSTEM/ingres is exported to the application clients.
- The **iibuild** procedure is executed at ULTRIX/SQL initialization time. See details in Chapter 4 of this guide.
- The **iidbms, iigcn, iigcc, dmfacp,** and **dmfrcp** processes are running.
- The **iistartup** command is in /etc/rc.local and it is executed at system startup time. See details in the section, "iirundbms and Database Management System Server Parameters" in Chapter 6.
- ULTRIX/SQL applications may be executed locally on the NFS server node.

Client characteristics:

- Only ULTRIX/SQL utilities, applications, **iigcn**, **iigcc**, and server processes execute on this node. No database management system server, archiver or recovery process, or concurrency control runs on this node.
- $II_SYSTEM/ingres is imported from the NFS server to a "mount" location.
- A symbolic link is created to $II_SYSTEM/ingres from $II_SYSTEM/sql.
- The **iigcn** and **iigcc** processes are running.

- The **iigcn** and **iigcc** processes need to be started from /etc/rc.local. This can be achieved by appending the following lines to this file.

  ```
  $II_SYSTEM/sql/utility/iirungcn
  $II_SYSTEM/sql/utility/iirungcc
  ```

  where $II_SYSTEM is the full physical path specification the level above your ULTRIX/SQL installation.

  In the following example of a file system, configuration for a clients, /usr/lp/sql is $II_SYSTEM for both NFS server and clients:

  ```
  df
  Filesystem kbytes used avail capacity    Mounted on
  server:/export/root/client              /
  server:/export/usr/client                /usr
  server:/usr/lp/sql/ingres                /usr/lp/sql/ingres
  ```

F.2.2 Scenario 2 (Client with Disk)

This scenario describes separate ULTRIX/SQL installations for the NFS server and clients. The NFS server and clients maintain logically separate installations with distinct $II_DATABASE, $II_CHECKPOINT, and $II_JOURNAL locations. The ULTRIX/SQL system software distribution is installed on the NFS server and may be reinstated entirely on the client, where symbolic links to the server installation are available. This minimizes disk space usage at the expense of communications traffic overhead.

The clients may share static ULTRIX/SQL files with the NFS server, such as the ULTRIX/SQL binaries, libraries, utilities, and technical notes. ULTRIX/SQL server processes (**iidbms, iigcn, iigcc**), archiver (**dmfacp**) and recovery (**dmfrcp**) processes, and concurrency control run on *both* the NFS server and clients. ULTRIX/SQL applications can be executed either locally or in connection with a remote database management system server process by using ULTRIX/SQL.

Note

> The following environment variables must be set to directories which are physically resident on the local disk, and *not* set to directories which are resident on an NFS mounted disk. For example:
>
> ```
> $II_DATABASE = /db
> $II_CHECKPOINT = /bck
> $II_JOURNAL = /bck
> $II_LOG_FILE = /bck
> ```

Likewise, alternative ULTRIX/SQL locations must reside on local disks.

To improve performance in the development environment, all clients should have their own copies of the sql/lib directory.

Server characteristics:

- $II_SYSTEM/ingres is exported to NFS clients. For example, to see the list currently exported, type:

  ```
  cat  /etc/exports
  ```

- The **iibuild** procedure is executed at ULTRIX/SQL initialization time. See details in Chapter 4.
- The **iidbms, iigcn, iigcc, dmfacp,** and **dmfrcp** processes are running.
- The **iistartup** command in /etc/rc.local is executed at system startup time. See details in "iirundbms and Database Management System Server Parameters" in Chapter 6.
- ULTRIX/SQL utilities and applications may be run locally on the NFS server. For example:

  ```
  sql  dbname
  ```

Client characteristics:

- The clients will share the bin, lib, utility, notes, and doc directories with the NFS server, through an NFS mount *and* symbolic link. The data, jnl, ckp, and files directories will be set locally using the **iibuild** utility. The files directory will need to be created by copying the central files directory.
- $II_SYSTEM/ingres is imported from the NFS server to some "mount" location. This "mount" location will not be used as $II_SYSTEM; a symbolic link will need to be set up for this purpose.
- The **iibuild** command is executed at ULTRIX/SQL installation time. See the file system configuration example below and details in Chapter 4.
- Concurrency control and **iidbms**, **dmfacp**, and **dmfrcp** processes are running on the clients, along with **iigcn** and **iigcc**.
- The **iistartup** command in /etc/rc.local is executed at system startup time.

  ULTRIX/SQL applications and utilities may be connected to a remote database management system server process (iidbms) by using ULTRIX/SQL:

  ```
  sql  nodename::dbname
  ```

  or to a local database management system server process (iidbms):

  ```
  sql  dbname
  ```

  An example of how you may configure a file system for a clients follows:

  ```
  df
  Filesystem kbytes used avail capacity     Mounted on
  server:/dev/sd0a                          /db
  server:/dev/sd0b                          /bck
  server:/export/root/client                /
  server:/export/usr/client                 /usr
  server:/usr                               /m/server/usr
  server:/usr/lp/sql/ingres                 /m/server/sql
  ```

- Identify a location for $II_SYSTEM on the clients. Define a symbolic link for the following directories from the mounted location to the $II_SYSTEM location. For this example, $II_SYSTEM is set to /usr/lp/sql.

```
ln -s /m/server/ingres/bin         install/rl/usr/lp/sql/ingres/bin
ln -s /m/server/ingres/notes       install/rl/usr/lp/sql/ingres/notes
ln -s /m/server/ingres/lib         install/rl/usr/lp/sql/ingres/lib
ln -s /m/server/ingres/utility     install/rl/usr/lp/sql/ingres/utility
ln -s /m/server/ingres/dbtmplt     install/rl/usr/lp/sql/ingres/dbtmplt
ln -s /m/server/ingres/release.doc install/rl/usr/lp/sql/ingres/doc
```

- Copy the $II_SYSTEM/files directory from the server node to $II_SYSTEM location of the clients node.

```
cp -r /m/server/ingres/files  /usr/lp/sql/ingres
```

- Create a symbolic link on the client node to $II_SYSTEM/ingres from $II_SYSTEM/sql.

```
ln -s /usr/lp/sql/ingres /usr/lp/sql/sql
```

- Before **iibuild** is executed, the .version and version.rel file must be copied from the server node to the $II_SYSTEM/ingres directory of the clients.

```
cp  /m/server/ingres/.version /usr/lp/sql/ingres
cp  /m/server/ingres/.version.rel  /usr/lp/sql/ingres
```

- Execute the **iibuild** procedure in Chapter 4 to create directories and set up the installation.

Note

A .version file must exist before **iibuild** is executed.

When upgrading a Scenario 2 installation, it is important to copy new file versions to the non-linked directories. Backup copies of the installation-dependent files, as well as those in the sql/ files directory, should be made.

Index

How to Order Additional Documentation

Technical Support

If you need help deciding which documentation best meets your needs, call 800-343-4040 before placing your electronic, telephone, or direct mail order.

Electronic Orders

To place an order at the Electronic Store, dial 800-234-1998 using a 1200- or 2400-baud modem from anywhere in the USA, Canada, or Puerto Rico. If you need assistance using the Electronic Store, call 800-DIGITAL (800-344-4825).

Telephone and Direct Mail Orders

Your Location	Call	Contact
Continental USA, Alaska, or Hawaii	800-DIGITAL	Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061
Puerto Rico	809-754-7575	Local Digital Subsidiary
Canada	800-267-6215	Digital Equipment of Canada Attn: DECdirect Operations KAO2/2 P.O. Box 13000 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6
International	————	Local Digital subsidiary or approved distributor
Internal*	————	SSB Order Processing - WMO/E15 *or* Software Supply Business Digital Equipment Corporation Westminster, Massachusetts 01473

* For internal orders, you must submit an Internal Software Order Form (EN-01740-07).

Reader's Comments

ULTRIX
ULTRIX/SQL Operations Guide
AA-PBZ9A-TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

Please rate this manual:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	□	□	□	□
Completeness (enough information)	□	□	□	□
Clarity (easy to understand)	□	□	□	□
Organization (structure of subject matter)	□	□	□	□
Figures (useful)	□	□	□	□
Examples (useful)	□	□	□	□
Index (ability to find topic)	□	□	□	□
Page layout (easy to find information)	□	□	□	□

What would you like to see more/less of? ____________________

____

____

What do you like best about this manual? ____________________

____

____

What do you like least about this manual? ____________________

____

____

Please list errors you have found in this manual:

Page Description

_____ ____________________

_____ ____________________

_____ ____________________

_____ ____________________

_____ ____________________

Additional comments or suggestions to improve this manual:

____

____

____

____

What version of the software described by this manual are you using? __________

Name/Title ____________________ Dept. __________

Company ____________________ Date __________

Mailing Address ____________________

____________________ Email ____________________ Phone __________

---- Do Not Tear – Fold Here and Tape --

digital™

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 33 MAYNARD MA

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
OPEN SOFTWARE PUBLICATIONS MANAGER
ZKO3-2/Z04
110 SPIT BROOK ROAD
NASHUA NH 03062-9987

---- Do Not Tear – Fold Here --

Cut
Along
Dotted
Line

Reader's Comments

ULTRIX
ULTRIX/SQL Operations Guide
AA-PBZ9A-TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

Please rate this manual:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	□	□	□	□
Completeness (enough information)	□	□	□	□
Clarity (easy to understand)	□	□	□	□
Organization (structure of subject matter)	□	□	□	□
Figures (useful)	□	□	□	□
Examples (useful)	□	□	□	□
Index (ability to find topic)	□	□	□	□
Page layout (easy to find information)	□	□	□	□

What would you like to see more/less of?

What do you like best about this manual?

What do you like least about this manual?

Please list errors you have found in this manual:

Page Description

Additional comments or suggestions to improve this manual:

What version of the software described by this manual are you using?

Name/Title Dept.

Company Date

Mailing Address

Email Phone

Do Not Tear – Fold Here and Tape

digital™

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST–CLASS MAIL PERMIT NO. 33 MAYNARD MA

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
OPEN SOFTWARE PUBLICATIONS MANAGER
ZKO3–2/Z04
110 SPIT BROOK ROAD
NASHUA NH 03062–9987

Do Not Tear – Fold Here

Cut
Along
Dotted
Line