

Advantage™ Ingres® Embedded Edition

Administrator's Guide

2.6



Computer Associates™

This documentation and related computer software program (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by Computer Associates International, Inc. ("CA") at any time.

This documentation may not be copied, transferred, reproduced, disclosed or duplicated, in whole or in part, without the prior written consent of CA. This documentation is proprietary information of CA and protected by the copyright laws of the United States and international treaties.

Notwithstanding the foregoing, licensed users may print a reasonable number of copies of this documentation for their own internal use, provided that all CA copyright notices and legends are affixed to each reproduced copy. Only authorized employees, consultants, or agents of the user who are bound by the confidentiality provisions of the license for the software are permitted to have access to such copies.

This right to print copies is limited to the period during which the license for the product remains in full force and effect. Should the license terminate for any reason, it shall be the user's responsibility to return to CA the reproduced copies or to certify to CA that same have been destroyed.

To the extent permitted by applicable law, CA provides this documentation "as is" without warranty of any kind, including without limitation, any implied warranties of merchantability, fitness for a particular purpose or noninfringement. In no event will CA be liable to the end user or any third party for any loss or damage, direct or indirect, from the use of this documentation, including without limitation, lost profits, business interruption, goodwill, or lost data, even if CA is expressly advised of such loss or damage.

The use of any product referenced in this documentation and this documentation is governed by the end user's applicable license agreement.

The manufacturer of this documentation is Computer Associates International, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227-7013(c)(1)(ii) or applicable successor provisions.

© 2002 Computer Associates International, Inc. (CA)

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.



Contents

Chapter 1: Overview of Ingres

Ingres Installation	1-1
Ingres Architecture	1-2
Ingres Querying and Reporting Tools	1-3
DBMS Server	1-3
How Ingres Executes a Query	1-3
DBMS Server Facilities	1-4
Ingres Visual Tools	1-5
Logging and Locking Systems	1-6
General Communication Facility (GCF)	1-14
JDBC Connectivity	1-15

Chapter 2: Managing Your System and Monitoring Performance

Ingres Visual Manager	2-1
Using Ingres Visual Manager	2-1
Ingres Visual Manager Window	2-2
Configuring Parameters	2-3
Defining Message Categories and Notification Levels	2-3
Setting Preferences	2-3
Monitoring Components	2-4
Visual DBA	2-5
Using the Performance Monitor	2-6
Performance Monitor Window	2-6
Working with Performance Monitor Components	2-7
Monitoring Components	2-7

Chapter 3: Maintaining Databases

Viewing Database Objects	3-2
Deleting Database Objects	3-2
Routine Maintenance Tips	3-3
Operating System Maintenance Tips	3-4
Verifying Databases	3-5
Avoiding User Errors	3-6
Translating File Names into Table Names	3-6
Retaining Templates of Important Tables	3-7

Chapter 4: Maintaining Storage Structures

Storage Structures and Performance	4-1
Limitations of Heap Structure	4-2
Modifying Storage Structures	4-4
Modify Cautions	4-5
Modify Options	4-5
Shrinking a B-tree Index	4-16
Extending a Table or Index	4-17
Modifying Secondary Indexes	4-17
When to Remodify B-tree Tables	4-19
Common Errors During the Modify Procedure	4-20
Overflow	4-20
Measuring the Amount of Overflow	4-20
Repetitive Key Overflow	4-21
Poorly Distributed Overflow	4-22
Overflow and Storage Structures	4-23

Chapter 5: Backup and Recovery

The Logging System	5-1
The Logging Facility	5-2
Log Space Reservation	5-2
The Recovery Process	5-3
The Archiver Process	5-3
The Cluster Server Process	5-3
Verifying Data Accessibility Before Backup	5-4
Backing Up a Database with Checkpoints	5-5
Checkpointing a Database	5-5

Checkpointing Tables	5-6
Online and Offline Checkpoints	5-7
Locking During a Checkpoint	5-7
Cleaning Up Outdated Checkpoints	5-8
Deleting the Oldest Checkpoint	5-9
Checkpoints and Destroyed Databases	5-9
Parallel Checkpointing in UNIX	5-9
Putting Checkpoints on Tape in Windows	5-10
Putting Checkpoints on Tape in UNIX	5-10
Putting Checkpoints on Tape in VMS	5-14
Using the Journaling System	5-15
Starting Journaling	5-15
Disabling Journaling	5-16
Stopping Journaling	5-17
Altering Database Characteristics	5-19
Resizing the Journal File	5-19
Producing Audit Trails with Journals	5-22
Copying a Database	5-26
Creating Copy Scripts	5-26
Copying a Database	5-26
Unloading a Database	5-27
Recovering Databases	5-27
Recovering Databases from Checkpoints and Journals	5-28
Recovering from the Loss of the Transaction Log File	5-33
Altering the Checkpoint Template File	5-33
Checkpoint Template Codes	5-34
Substitution Parameters	5-35
Valid Code Combinations	5-36
Format of the Checkpoint Template File in Windows	5-37
Format of the Checkpoint Template File in UNIX	5-38
Format of the Checkpoint Template File in VMS	5-39
An Alternate Checkpoint Template File in UNIX	5-39
Backup and Recovery of the Master Database (iiddb)	5-40
Tracing with Set Log_Trace	5-40

Chapter 6: Improving Database and Query Performance

Locking and Concurrency Issues	6-1
Identifying Lock Waits	6-2
Multi-Query Transactions (MQTs)	6-2
Managing Overflow	6-3
Set Statements	6-4

Database Maintenance	6-4
Design Issues	6-6
Diagnostic Hierarchy	6-6
Storage Structures and Index Design	6-6
Keys	6-7
Query Design	6-9
Before Calling Technical Support	6-9
Isolate and Analyze the Suspect Query	6-10
Create a Test Case	6-11

Appendix A: Ingres Commands

Audience	A-1
Special Considerations	A-1
Command Syntax	A-2
Standard Command Line Flags and Parameters	A-2
Uppercase Flags	A-5
Using Schemas for Owner Qualification	A-5
Delimited Identifiers on the Command Line	A-6
abf	A-9
Syntax	A-9
Description	A-9
accessdb	A-10
Syntax	A-10
Description	A-10
alterdb	A-10
Syntax	A-10
Description	A-11
arcclean	A-12
Syntax	A-12
Description	A-13
Examples	A-14
auditdb	A-15
Syntax	A-15
Description	A-15
Examples	A-18
ckpdb	A-20
Syntax	A-20
Description	A-20
Examples	A-22
convrep	A-23
Syntax	A-23

copyapp	A-23
Syntax	A-23
Description	A-23
Examples	A-26
copydb	A-26
Syntax	A-26
Description	A-26
Examples	A-30
createdb	A-32
Syntax	A-32
Description	A-32
Examples	A-35
dereplic	A-35
Syntax	A-35
destroydb	A-35
Syntax	A-36
Description	A-36
Examples	A-36
esqla (ESQL Preprocessor)	A-37
Syntax	A-37
Description	A-37
extenddb	A-42
Syntax	A-42
Description	A-42
Examples	A-43
fastload	A-43
Syntax	A-43
Description	A-43
Examples	A-44
genxml	A-45
Syntax	A-45
Description	A-45
Examples	A-47
infodb	A-47
Syntax	A-48
Description	A-48
Example	A-48
ingmenu	A-57
Syntax	A-57
Description	A-58

ingprenv	A-58
Syntax	A-58
Description	A-59
ingsetenv	A-60
Syntax	A-60
Description	A-60
ingunset	A-60
Syntax	A-60
Description	A-61
ipm	A-61
Syntax	A-61
Description	A-61
isql	A-62
Syntax	A-62
Description	A-62
Example	A-63
modifyfe	A-63
Syntax	A-63
Description	A-63
netutil	A-64
Syntax	A-64
Description	A-65
Examples	A-65
optimizedb	A-66
Syntax	A-66
Description	A-66
Examples	A-70
qbf	A-71
Syntax	A-72
Description	A-72
query	A-73
Syntax	A-73
Description	A-74
reconcil	A-74
Syntax	A-74
Description	A-76
Example	A-76

relocatedb	A-78
Syntax	A-78
Description	A-78
Examples	A-79
repcat	A-81
Syntax	A-81
Description	A-82
Examples	A-82
repcfg	A-82
Syntax	A-82
Examples	A-83
repdbcfg	A-83
Syntax	A-83
Description	A-85
Examples	A-85
repinst	A-85
Syntax	A-85
Description	A-86
Examples	A-86
repmgr	A-86
Syntax	A-86
Description	A-87
Example	A-87
repmod	A-88
Syntax	A-88
Description	A-88
report	A-88
Syntax	A-88
Description	A-88
repstat	A-92
Syntax	A-92
Description	A-92
rollforwarddb	A-92
Syntax	A-92
Description	A-92
Examples	A-97
rpserver	A-98
Syntax	A-98
Description	A-98
rsstatd	A-98
Syntax	A-98

sql	A-98
Syntax	A-98
Description	A-99
Examples	A-102
starview	A-103
Syntax	A-103
Description	A-103
Example	A-104
statdump	A-104
Syntax	A-104
Description	A-104
Examples	A-106
sysmod	A-107
Syntax	A-107
Description	A-107
tables	A-108
Syntax	A-108
Description	A-108
Example	A-109
unloaddb	A-109
Syntax	A-109
Description	A-109
Examples	A-112
upgradedb	A-113
Syntax	A-113
Description	A-113
upgradeefe	A-115
Syntax	A-115
Description	A-115
Examples	A-116
usermod	A-116
Syntax	A-116
Description	A-116
verifydb	A-117
Syntax	A-117
Description	A-117
Examples	A-122
xmlimport	A-123
Syntax	A-123
Description	A-123
Example	A-124

Appendix B: Ingres Utilities

cacheutil	B-1
Syntax	B-1
Description	B-1
catalogdb	B-2
Syntax	B-2
Description	B-2
Examples	B-3
cbf	B-3
Syntax	B-4
Description	B-4
cscleanup	B-4
Syntax	B-5
Description	B-5
csreport	B-5
Syntax	B-5
Description	B-5
Examples	B-6
deregdocs	B-6
Syntax	B-6
Description	B-6
ICETranslate	B-7
Syntax	B-7
Description	B-7
Example	B-8
iigenres	B-8
Syntax	B-8
Description	B-8
Example	B-8
iigetres	B-9
Syntax	B-9
Description	B-9
iimklog	B-9
Syntax	B-9
Description	B-9
iiremres	B-9
Syntax	B-10
Example	B-10
iisetres	B-10
Syntax	B-10
Example	B-10

iivalres	B-11
Syntax	B-11
Example	B-11
iimonitor	B-11
Syntax	B-11
Description	B-12
iinamu	B-15
Syntax	B-15
Description	B-15
Examples	B-17
iishowres	B-19
Syntax	B-19
Description	B-19
iiizic	B-19
Syntax	B-19
Description	B-19
iizck	B-20
Syntax	B-20
Description	B-20
ingbuild (UNIX) or vmsinstal (OpenVMS)	B-21
Syntax	B-21
Description	B-21
ingnet	B-21
Syntax	B-21
Description	B-22
ingstart	B-22
Syntax	B-22
Description	B-22
Examples	B-23
ingstop	B-24
Syntax	B-24
Description	B-24
ivm	B-25
Syntax	B-25
Description	B-25
lockstat	B-26
Syntax	B-26
Description	B-26
Example	B-27

logstat	B-33
Syntax	B-34
Description	B-34
Example	B-45
mkrawarea	B-48
Syntax	B-48
Description	B-48
mkrawlog	B-48
Syntax	B-48
Description	B-48
rcpconfig	B-49
Syntax	B-49
Description	B-49
rcpstat	B-50
Syntax	B-50
Description	B-51
regdocs	B-51
Syntax	B-52
Description	B-52
rmcmdgen	B-53
Description	B-53
rmcmdrmv	B-54
Description	B-54
rmcmdstp	B-54
Description	B-54
syscheck	B-54
Syntax	B-54
Description	B-54
vcbf	B-55
Syntax	B-55
Description	B-55
vdba	B-56
Syntax	B-56
Description	B-56
Examples	B-59

Overview of Ingres

The Ingres system administrator has all privileges and holds the primary responsibility for installing and maintaining Ingres. Logging in as the Ingres system administrator provides the permissions needed for an embedded installation and subsequent maintenance.

The Ingres system administrator has the following responsibilities:

- Authorize users to access Ingres
- Install Ingres files
- Define Ingres variables such as II_DATABASE and II_INSTALLATION
- Start, stop, configure, and monitor server(s)
- Disconnect or suspend a session connected to a server
- Shut down the Ingres installation (or parts of it)

Ingres Installation

The Ingres system administrator installs and maintains the following principal components or associated family of compatible components:

- DBMS Servers (including distributed servers)
- Internet Communication
- General Communications (including Name server, Networking, and JDBC connectivity)
- Ingres Visual Tools (including Visual DBA, Configuration Manager, Ingres Visual Manager, Ingres Journal Analyzer, Ingres Network Utility, and the Remote Command server)
- Logging and locking systems

An Ingres installation also has:

- Library files and utilities, provided on your release media
- Configuration files and error log files, created during installation and at runtime by Ingres
- Databases and their associated files: the checkpoint, journal, dump, and work files, also created by Ingres during installation and at runtime

The locations of these files are selected at installation time, together with the configuration of the DBMS server(s) and the logging and locking systems. For the initial installation, default settings are provided by the system whenever possible.

After the installation is running, you should monitor all of your Ingres servers (DBMS, Name, Star, Net communications, Bridge, and Ice) to ensure that they are running and that they are configured for top performance. If necessary, you can reconfigure selected Ingres parameters as described later in this guide.

Ingres Architecture

The following list contains elements of the Ingres architecture with which you should be familiar:

- DBMS servers (including distributed servers)
- Internet communication
- General communications (including Name server, Networking, and JDBC connectivity)
- Ingres Visual Tools (including Visual DBA, Configuration Manager, Ingres Visual Manager, Ingres Journal Analyzer, Ingres Network Utility, and the Remote Command server)
- Logging and locking systems

If your installation uses Net, it will include one or more Net communications servers (also referred to simply as *communications servers*). The communications server is discussed in the General Communication Facility section in this chapter. For more information, refer to the “Establishing Communications” chapter of this guide.

Ingres Querying and Reporting Tools

Users can employ Ingres querying and reporting tools (such as Report Writer, Query-By-Forms (QBF), and Vision) to access databases.

Whenever a user accesses a database through one of these Ingres tools, a process is created and communication is initiated with the database through a DBMS server. The information about how the user may connect to the server is provided by the Name server, which is part of the General Communication Facility.

DBMS Server

The Ingres client-server architecture allows multiple users access to databases through connections to one or more DBMS server processes. The DBMS server (iidsbms) is a multi-threaded daemon process that performs asynchronous disk input/output. The number of users connected to a server is in practice limited by hardware constraints or operating system limits.

Ingres DBMS servers can be configured to fit specific needs. For example, you can:

- Designate one server to access a particular database, denying other servers access to that database.
- Configure a server as a “fast commit” server to achieve higher levels of performance.
- Specify that two or more servers share a common memory buffer cache.
- Specify private buffer cache with distributed multi-cache protocol.

Some of the elements of the DBMS server are described as facilities. The following section contains a list of facilities, their acronyms, and brief descriptions.

How Ingres Executes a Query

Typically, your interaction with Ingres consists of at least two processes: an Ingres tool or application of some kind and the DBMS server process. The Ingres tool program handles screen display and prompts, takes user input and then issues a query that is sent to the DBMS server, where it is formatted, optimized and then executed on behalf of the user. The DBMS server then returns the data to the Ingres tool program.

The DBMS server can execute queries for a large number of users, each running an Ingres tool. To accomplish this, the DBMS server is a multi-threaded process. Even though it is a single process, it can execute queries on behalf of multiple users. These queries execute as multiple “sessions” inside the DBMS. Visual DBA permits you to view those sessions that are running in the DBMS server at any moment. See the Visual DBA procedural help for detailed steps about viewing DBMS sessions.

Query Environment

When a thread or session executes a query inside the DBMS, it does so in a *query environment*. The query environment consists of:

- A quantity of resources available from the operating system for use by the session
- The rules under which the query is executed

These rules have to do with which query language is used, what locking strategy is employed, what diagnostic information is also returned, what default behavior Ingres adopts for various query language statements, and so on.

DBMS Server Facilities

The following are DBMS server facilities. Some of the configuration parameters configure DBMS server facilities. You may find this section helpful to use with the “Configuring Ingres” chapter and the troubleshooting information in the “Basic Troubleshooting” chapter.

The server consists of these components:

- **Abstract Data Type Facility (ADF)**— ADF does all the work that involves 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 Ingres tools also use ADF.

This abstract data type facility is also compiled into the Ingres tools, such as QBF and the Terminal Monitor. It is used to manipulate Ingres data types in the front-end process before sending them to the DBMS.

- **Data Manipulation Facility (DMF)**— The Data Manipulation Facility (DMF) manages the DBMS interface to disk storage. In addition to managing all storage structures (hash, heap, ISAM, B-tree, BLOBs, and so on), DMF uses the logging and locking systems to control transaction processing and to handle concurrency issues. Included within DMF is a “buffer manager” that controls access to a cache (possibly shared) of database pages.

- **Optimizer Facility (OPF)** – The optimizer, OPF, selects the optimal plan for implementing queries. It also converts the query tree that comes out of the parser into the query execution plan (QEP). OPF uses a memory pool for its operations.
- **Parser Facility (PSF)** – The parser, PSF, converts queries from text form to internal format. 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.
- **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.
- **Query Storage Facility (QSF)** – QSF provides shared memory facilities with a temporary or permanent place to store query trees and query plans.
- **Relation Description Facility (RDF)** – RDF is a central caching point for information about tables. It is used by Star, PSF, and OPF.
- **System Control Facility (SCF)** – SCF is the central controlling facility that manages sessions on behalf of user (client) requests. It coordinates the actions among the various facilities involved in processing a query, including thread monitoring and switching. SCF is also responsible for managing server-wide access to shared resources, such as operating system semaphores and memory.

Ingres Visual Tools

The following are GUI tools for use in visually starting, stopping, managing, and monitoring your Ingres installation.

Ingres Visual Manager	Ingres Visual Manager provides a global view into your Ingres installation. It serves as a system console from which you can manage Ingres components and access other utilities. This utility captures events that are occurring in the system and allows them to be filtered for emphasis, according to the system administrator's preferences. For information on command line tasks, see the appendices, "Ingres Commands" and "Ingres Utilities" in this guide.
Visual DBA	Visual DBA provides an alternative set of system administration tools. It allows you to perform system administrator functions—including configuring, performance monitoring, backup and recovery, and remote database optimization—using a GUI interface. For information on command line tasks, see the appendices, "Ingres Commands" and "Ingres Utilities" in this guide.
Remote Command Server	The remote command server (RMCMD) must be started in installations where an Ingres DBMS server is running in order for certain DBA tasks to be accessible remotely with Visual DBA. The remote command server is started on the server side, not on the Visual DBA client.

The DBA tasks are primarily those that would not have an equivalent through an SQL statement, for example, creating or dropping a database, displaying selected portions of the journal for a database, or starting a replication server remotely. By default, only the ingres system administrator is authorized to perform such tasks remotely through Visual DBA.

To allow a user other than the ingres system administrator to execute the remote commands, issue the following SQL statements while connected to the `iidbdb` as the ingres system administrator:

```
grant select, insert, update, delete on remotecmdinview to user
grant select, insert, update, delete on remotecmdoutview to user
grant select, insert, update, delete on remotecmdview to user
grant execute on procedure launchremotecmd to user
grant execute on procedure sendrmcmdinput to user
grant register, raise on dbevent rmcmdcmdend to user
grant register, raise on dbevent rmcmdnewcmd to user
grant register, raise on dbevent rmcmdnewinputline to user
grant register, raise on dbevent rmcmdnewoutputline to user
grant register, raise on dbevent rmcmdstp to user
```

Note: No grants should be made directly to the underlying tables.

Configuration Manager	The Configuration Manager utility provides a GUI interface for configuring your Ingres installation. For more information, see the appendices, "Ingres Commands" and "Ingres Utilities" in this guide.
Ingres Network Utility	The Ingres Network Utility is a stand-alone tool that permits you to view and define Ingres/Net node definitions, which then allows you to connect to remote Ingres installation through Ingres/Net. In addition, Ingres Network Utility allows you to launch the stand-alone Database Object Manager, Monitor, and SQL/Test windows for such installations.
Ingres Import Assistant	The Ingres Import Assistant assists you with importing an external text file into an Ingres or gateway table. It can be run as a stand-alone utility or from within Visual DBA.
Ingres Journal Analyzer	The Ingres Journal Analyzer utility allows you to view and analyze journaled transactions and individual underlying statements. It also enables you to create, recover, and redo SQL scripts, which are used to recover or redo individual row operations without redoing or rolling back the whole database or table.

Logging and Locking Systems

The logging and locking systems coordinate the locking, recovery, and journaling of databases. The system is composed of the following components:

- The Lock Manager
- The Logging Facility
- Recovery process
- Archiver process

- Primary and dual transaction log files
- Other log files

Lock Manager



The locking component controls concurrent access to a database.

In UNIX, this component makes use of the shared memory segments and semaphore resources that you install when you configure the UNIX kernel. ■

Shared memory is allocated to components initially during the installation procedure. The amount of shared memory that your installation requires depends on the logging and DBMS server parameters that you select during the procedure.

Logging Facility

The logging facility implements a circular “write-ahead” transaction log file for the management of transactions within the installation. It ensures that log records are written in a way that makes them accessible to the recovery and archiver processes. The recovery and archiver processes manipulate the data in the transaction log file when certain events occur. For example, after a transaction is committed, the logging facility moves the log buffer, which resides in shared memory, to the transaction log file.

The logging facility is shared by all servers in the installation, as well as the recovery and archiver processes. Log records written by different servers, or written by several threads in the same server, may be combined with log records written by other servers or threads.

Log records are first copied to in-memory log buffers, which are written to the log file as they fill. Memory log buffers and disk log file blocks are the same length. The number of log buffers is configurable, set according to the performance requirements of the system. The logging system manages multiple asynchronous writes of log buffers to the log file. Once written, a log file block is never rewritten until the file wraps around.

The number of log file blocks corresponds to the size of the log file, and is specified when the log file is created.

In a properly tuned system, most log file buffers are completely full of log records when they are written to the log file. However, as all log records associated with a transaction must be forced to the log file at certain times, principally at end transaction time, a small percentage of log file blocks may contain unused space. Server group commit logic is designed to minimize the frequency of log force operations and increase log file space utilization.

Recovery Process

Each installation has a dmfrcp (data manipulation facility recovery process) that is assigned recovery responsibilities. In normal circumstances, transaction commit and rollback processing are handled within each DBMS server. In the event of a server or system failure, however, the recovery process performs the required recovery, including backing out uncommitted transactions and ensuring that committed transactions are properly reflected on disk.

The recovery process operates in both online and offline modes:

- Online recovery is performed when a server stops abnormally. In this case, users connected to other servers are generally unaffected by the recovery.
- Offline recovery is performed when the installation is brought back up after it has stopped abnormally. In this case, the installation remains unavailable until the recovery process completes all required recovery.

The recovery process maintains a history of important actions in its own message log file, named iircp.log. See the Recovery Log section for details.

Archiver Process

Each installation has a single archiver process called the dmfacp (data manipulation facility archiver process). The archiver process is responsible for copying the history of operations performed on journaled databases from the transaction log file to the journal files. Journal files contain the subset of transaction log file information associated with a specific database.

Journal files are created (and optionally destroyed) during a *checkpoint* operation. In the event of a disaster, the database can be rebuilt by restoring the latest checkpoint and then applying journal file information. Disaster recovery operations are coordinated by the rollforwarddb command. It can be accessed through the Visual DBA Database menu, DOM window.

The archiver process maintains a history of important actions in its own message log file, iiacp.log. See the Archiver Log section for details.

Transaction Recovery

Transaction recovery involves the transaction log file (see the Transaction Log File section for details) that is used as a write-ahead log, plus journal files maintained on a per-database basis. Log files contain short-term recovery information regarding active databases, while the journal files contain long-term information used for auditing and disaster recovery. While the log file is circular and wraps around, journal files are of configurable length and are retained indefinitely.

Ingres employs a page-oriented recovery scheme, where changes to pages are reflected in the transaction log file. Recovery information is divided into two types: undo (or rollback) operations and redo (or cache restore) operations.

Undo or transaction backout recovery is performed by the DBMS server. For example, when a transaction is aborted, transaction log file information is used to roll back all related updates. The DBMS server writes the Compensation Log Records (CLRs) to record a history of the actions taken during undo operations.

While undo recovery is transaction-oriented, redo recovery is database-oriented. Redo recovery is performed after a server or an installation fails. Its main purpose is to recover the contents of the DMF cached data pages that are lost when a fast-commit server fails. Redo recovery is performed (in a non-clustered installation) by the recovery process. Redo recovery precedes undo recovery.

Ingres performs both online and offline recovery, as described in the Recovery Process section.


Ingres Log Files

Ingres maintains log files to which it writes information about the installation activities.

Transaction Log File

Each installation has a transaction log file, and an optional dual log file. The log file holds information about all open transactions and is used to recover active databases after a system failure. You have the option to change its size and number of partitions at startup.

UNIX

The UNIX log file can be created as a raw partition, or as a number of raw partitions if you choose. 

Error Log

The main error log is a readable text file that you can use for troubleshooting. The error log is maintained in the following file:

Windows

%II_SYSTEM%\INGRES\FILES\ERRLOG.LOG 

UNIX

\$II_SYSTEM/ingres/files/errlog.log 

VMS`II_SYSTEM:[INGRES.FILES]ERRLOG.LOG` 

Messages about the installation are appended to this log with the date and time at which the error occurred. This is generally the first place to look when troubleshooting a problem.

The error log contains the following information:

- Error messages
- Warning messages
- Server start up and shutdown messages

The Ingres system administrator maintains the error log file. The file continues to grow until manually truncated. The installation should be shut down before truncating or removing the errlog.log file.

Archiver Log

The archiver log contains information about the current archiver process in the following file:

Windows`%II_SYSTEM%\INGRES\FILES\IIACP.LOG` **UNIX**`$II_SYSTEM/ingres/files/iiacp.log` **VMS**`II_SYSTEM:[INGRES.FILES]IIACP.LOG` 

This file is appended to when the archiver process starts. The log contains the following information:

- Archiver start up
- Archiver error messages
- Archiver warning messages

Recovery Log

The recovery (dmfrcp) log contains information about the current recovery process in the following file:

Windows

%II_SYSTEM%\INGRES\FILES\IIRCP.LOG 

UNIX

\$II_SYSTEM/ingres/files/iircp.log 

VMS

II_SYSTEM:[INGRES.FILES]IIRCP.LOG 

This file is appended to when the recovery process starts. The log contains the following information:

- Current logging and locking parameter values
- Recovery process error messages
- Recovery process warning messages
- Recovery operations information

The recovery log should be monitored if you are unable to connect to Ingres and suspect that the DBMS server is in recovery mode. You can determine the recovery state by looking at the log file header information. To view log file header information in Visual DBA, select the Log Information branch in a Performance Monitor window and click the Header tab.

VMS Cluster Service Process Log

VMS

The cluster service process (on VMS clustered installations only) log maintains information about the cluster service process (CSP). A new version of this file is written each time the CSP is started.

The CSP log file contains information on the following:

- Archiving and recovery messages
- Node information

You should monitor the CSP log if problems arise with node logging or locking. 

Primary Configuration Log Files

Ingres maintains transcripts of various configuration operations. Configuration log files that you may find useful are given in the table below. These files are in the directory indicated by `IL_CONFIG`:

Log File Name	Description
config.log	Contains a log of the changes made with the configuration utility (cbf).
rcpconfig.log	Contains log and error information of the last time rcpconfig was run. This might have been for reconfiguration, shutdown, or initial installation.

Optional Configuration Log Files

Configuration log files that are optionally maintained are given in the table below. These files, if present, are in the `IL_CONFIG` directory (in UNIX, the setting should always be `$IL_SYSTEM/ingres/files`):

Log File Name	Description
iilink.log	The file contains a log of the last time the iilink command was run. For details on the iilink command, see the appendices, "Ingres Commands," and "Ingres Utilities," in this guide.
iivdb.log	Contains a transcript of the last time verifydb was used to diagnose or attempt recovery of a damaged or inconsistent database. This file is created the first time verifydb is run.
lartool.log	Contains a transcript, with any errors, of the last time lartool was used to manually abort or commit a running transaction. This file is created the first time lartool is run.

Other Optional Log Files

The following types of optional log files may be present on your installation:

- Individual process logs. For the following processes you can set up a separate log file to isolate the error messages relating to that process:
 - DBMS
 - GCC

You would normally do this for a specific troubleshooting purpose. These messages go to the `errlog.log` by default. If you define one of these separate error logs, all messages will go to *both* that file and `errlog.log`.

- Trace log for tracking messages at a greater level of detail
- Log for an optional Ingres facility

These optional log and trace log files can be established by setting the associated Ingres variables:

Log File Name	Description
DBMS error log	The DBMS error log, optionally defined as a separate file. This log file is established by setting the Ingres variable <code>II_DBMS_LOG</code> to a user file name. All DBMS errors and messages go to <code>errlog.log</code> by default.
GCC error log	The GCC error log, for installations using Net. This is in a user-defined file. This log file is established by setting <code>II_GCC_LOG</code> to the full name of a file. The error logging level is specified by <code>II_GCC_LOG_LVL</code> .
GCC trace log	The GCC trace log is set up for specific troubleshooting efforts. You set <code>II_GC_LOG</code> to a user file name. The associated Ingres variable <code>II_GC_TRACE</code> defines the level of tracing.
Star error log	The Star error log, optionally defined as a separate file. This log file is established by setting the Ingres variable <code>II_STAR_LOG</code> to a user file name. All STAR errors and messages go to <code>errlog.log</code> by default.

General Communication Facility (GCF)

GCF manages communication among all the components of Ingres. GCF has four elements whose acronyms you will encounter:

- **Name server**—The Name server (`iigcn`) keeps track of all DBMS, Star, JDBC, Bridge, ICE, and communications (Net) servers associated within an installation. There is one Name server process per installation. The Name server provides information to user processes that allows the user to connect to a local DBMS server. When a user process wants to connect to a remote DBMS server, the Name server provides information that allows the user to connect to a communications server. The communications server establishes communication with the remote DBMS server.

The Name server checks regularly (the default is every five minutes) to ensure that all DBMS servers on its list are functioning. If a server has shut down, the Name server automatically deletes its registration.

As part of the General Communication Facility services, Ingres provides a Name Server Maintenance Utility, *iinamu*. This utility allows the Ingres system administrator to manually add servers to or delete servers from the list maintained by the Name server or to stop and restart the Name server itself. For more information about this utility, see the online *Command Reference Guide*.

- **Communications server**— The communications server (*iigcc*) is a daemon process that provides the network communication element of the Net product. It monitors outgoing communication from local applications to remote DBMS servers and incoming communication from remote applications to local DBMS servers. An installation can have multiple communications server processes. For additional information about the communications server, see the “Establishing Connections” chapter and the “Using Netutil” appendix of this guide.
- **Bridge server**— Bridge enables a client application running on one type of local area network to access an Ingres server running on a different type of network. Bridge will “bridge” a client using one network protocol to a server using another. For more information on the Bridge server, see the “Establishing Communications” chapter and “Bridge Server Configuration” appendix of this guide.
- **General Communications Architecture (GCA)**— This is the lowest level GCF Application Program Interface. The GCA maintains communication connections between processes on the same local Ingres installation. The GCA is a subroutine library that is a part of all Ingres tools, DBMS servers, Star servers, and the libraries associated with embedded SQL and EQUEL.

JDBC Connectivity

The Ingres JDBC driver is a pure Java implementation of the JDBC 2.1 interface. The driver supports application, applet, and servlet access to Ingres data sources via a middle-ware JDBC server. The driver supports the full JDBC 2.1 API and the JDBC 2.0 Standard Extension API interfaces *DataSource*, *ConnectionPoolDataSource*, and *XADataSource*.

The following features are available with the JDBC server/driver:

- Driver properties *autocommit_mode* and *select_loop*. All properties may be specified using URL attributes.
- Autocommit transaction extensions
- Select loop support
- Extensions for Ingres ‘empty’ dates and date intervals
- Support for Native Character Set (*nchar*, *nvarchar*) columns

- Database procedure support using CallableStatement interface
- JDBC 2.1 API support
- JDBC 2.0 Standard Extension API interfaces DataSource, ConnectionPoolDataSource, and XADataSource

JDBC Components

The JDBC product is composed of two units: an installation server and a Java client driver. The JDBC server runs as a part of a standard Ingres installation. It may be configured using `cbf` and started using `ingstart`. The server translates JDBC requests from the driver into Ingres internal format and forwards the request to the appropriate DBMS server. Through the JDBC server, a JDBC client has full access to Ingres, Star, and EDBC databases. The JDBC server can also access servers on other platforms via Ingres/Net.

The JDBC driver is delivered as a Java archive, named `edbc.jar`, located in the `lib` directory of the Ingres installation. In addition, the class files for a Java utility program, `EdbcInfo`, which displays the driver version information, are also located in the same directory. Access to the Ingres driver may require, depending on the Java environment used, adding the Java archive to the `CLASSPATH` environment setting or as a resource in the appropriate utility. For browser/applet access, the Java archive may need to be copied to the web server directories.

JDBC 2.1 API Features

The JDBC driver is compliant with the JDBC 2.1 API specification. JDBC 2.1 API interfaces are fully supported with the following exceptions:

- Calendars

Ingres stores date/time values in GMT (same as Java). With an Ingres DBMS, the JDBC driver handles all date/time values in GMT, so calendars provided in `setXXX()` and `getXXX()` methods may be ignored. EDBC servers do not reference date/time values to a particular timezone. The JDBC driver uses the local timezone when accessing a non-Ingres DBMS, and will use calendars if provided. Calendars may also be used when converting values (`getDate()` of a CHAR column, or `getTimestamp()` of a DATE column with no time component).

- Batch updates

Batched execution for Statements, PreparedStatements, and CallableStatements is supported by individual execution of each batched request. The driver implementation for batch updates is only as efficient as an application making individual update requests.

- Data types

The new SQL data types BLOB, CLOB, ARRAY, REF, DISTINCT, STRUCT, and JAVA_OBJECT are not supported. Neither is the storage, nor mapping, of Java objects (SQLInput, SQLOutput, and SQLData).

- Result sets

Result sets generated by executeQuery() requests will always be TYPE_FORWARD_ONLY (non-scrollable) and CONCUR_READ_ONLY (non-updateable). Methods associated with scrolling and updating will throw SQLExceptions. The row query methods (isFirst(), isBeforeFirst(), isAfterLast(), and getRow()) are supported with the exception of isLast(), which always returns false.

JDBC 2.0 Standard Extensions

The following classes are provided by the JDBC driver, and implement interfaces defined by the JDBC 2.0 Standard Extension API:

JDBC Driver Class	Implemented JDBC 2.0 Interface
ca.edbc.jdbcx.EdbcDataSource	javax.sql.DataSource
ca.edbc.jdbcx.EdbcCPDataSource	javax.sql.ConnectionPoolDataSource
ca.edbc.jdbcx.EdbcXADataSource	javax.sql.XADataSource

The DataSource classes support the following properties and associated getter/setter methods:

DS Property	Driver Property	Description
DataSourceName		Data source name
Description		Data source description
ServerName		Server host name (required)
PortNumber		Numeric port ID (required). A port ID must be provided either numerically or symbolically.
DatabaseName		Database name (required)
User	user	User name (required)
Password	password	User's password (required)
RoleName	role	Role identifier
GroupName	group	Group identifier
DbmsUser	dbms_user	User name for DBMS session

DS Property	Driver Property	Description
DbmsPassword	dbms_password	User's DBMS password
ConnectionPool	connect_pool	Server connection pooling
AutocommitMode	autocommit_mode	Autocommit cursor handling
SelectLoops	select_loop	Select loop vs. cursor queries

The DataSourceName and Description properties are for the use of the deployer of the DataSource. The required properties correspond to the information contained in a connection URL (see Accessing the Driver section). The remaining DataSource properties correspond to driver properties defined in the Driver Properties section.

Running the JDBC Server

The JDBC server is started and stopped via the standard installation startup and shutdown commands. The server may also be stopped using the following command:

```
iigcstop addr
```

The server address may be obtained using the iinamu utility and issuing the following:

```
show jdbc
```

Loading the Driver

The following Java source line should be added to an application or applet prior to attempting to establish a connection using the JDBC driver:

```
Class.forName( "ca.edbc.jdbc.EdbcDriver" ).newInstance();
```

Depending upon the Java environment, calling the forName() method may be sufficient to load and initialize the driver classes. Some environments, most notably Microsoft Internet Explorer, require the instantiation of an EdbcDriver object to fully initialize the driver.

Accessing the Driver

The JDBC driver can be accessed using a DriverManager getConnection() method with a URL in the following format:

```
jdbc:edbc://host:port/db;attr=value
```

The parameters are described as follows:

host—the network name or address of the host on which the target JDBC server is running.

port—the network port used by the JDBC server. This may be a numeric port number or an Ingres symbol port address such as II7.

db—the target database specification. Any valid Ingres database designation may be used, including vnode and server class (that is, vnode::database/class).

attr=value—optional attribute name and value pair. Multiple attribute pairs are separated by a semi-colon.

Attributes represent driver properties that are implementation-specific and may be used to configure the new connection. Driver properties are discussed in the following section.

Driver Properties

Driver properties allow applications to establish connection parameters that are driver-dependent. JDBC driver properties may be specified as connection URL attributes or as a properties set in a DriverManager getConnection() method.

A user name and associated password are required when making a connection. They may be provided as attributes, properties, or as parameters to a DriverManager getConnection() method.

The following properties are supported by the JDBC driver:

Property	Attribute	Description
user	UID	The user ID on the DBMS machine (required).
password	PWD	The user's operating system password (required).
role	ROLE	The desired role identifier. If a role password is required, include it with the role name as follows: <i>name/password</i> .
group	GRP	The user's group identifier.
dbms_user	DBUSR	The user name to be associated with the DBMS session (Ingres -u flag, may require admin privileges).
dbms_password	DBPWD	The user's DBMS password (Ingres -P flag).

Property	Attribute	Description
connect_pool	POOL	<p>Server connection pool control. Available options are:</p> <ul style="list-style-type: none"> ■ off—requests a non-pooled connection when server pooling is enabled. ■ on—requests a pooled connection when server pooling is optional.
autocommit_mode	AUTO	<p>Autocommit cursor handling mode. Available options are:</p> <ul style="list-style-type: none"> ■ dbms (default)—autocommit processing is done by the DBMS server. ■ single—JDBC server enforces single cursor operation during autocommit. ■ multi—JDBC server simulates autocommit operations when more than one cursor is open. <p>See the Autocommit Transactions section for further details.</p>
select_loop	LOOP	<p>Select loop vs. cursor queries. Available options are:</p> <ul style="list-style-type: none"> ■ on—uses select loops to retrieve query results. ■ off—uses cursors (default). <p>See the Cursors and Select Loops section for further details.</p>

Attributes may also be specified using the property name as the attribute name. Thus 'UID=user1' and 'user=user1' are semantically the same. When provided both as properties and attributes, the property value takes precedence.

Autocommitting Transactions

Application writers should be aware that the Ingres DBMS imposes severe limits on the operations that may be performed when autocommit is enabled (the JDBC default transaction mode) and a cursor is opened. In general, only one cursor at a time may be open during autocommit, and only cursor-related operations (cursor delete, cursor update) can be performed. Violating this restriction will result in an exception being thrown with the message text 'No MST is currently in progress, cannot declare another cursor'.

Cursors are opened by the Statement and PreparedStatement executeQuery() methods and remain open until the associated ResultSet is closed. The driver may close a cursor automatically when the end of the result set is reached, but applications should not rely on this behavior. JDBC applications can avoid many problems by calling the close() method of each JDBC object when the object is no longer needed.

The JDBC driver provides alternative autocommit processing modes that may help overcome the restriction of autocommit transactions or handle problems applications have with closing result sets. The autocommit processing modes may be selected by setting the connection property 'autocommit_mode' (see the Driver Properties section) to one of the following values:

Value	Mode	Description
dbms	DBMS (default)	Autocommit processing is done by the DBMS and is subject to the restrictions mentioned above.
single	Single-cursor	The JDBC Server allows only a single cursor to be open during autocommit. If a query or non-cursor operation is requested while a cursor is open, the JDBC server will close the open cursor. Any future attempts to access the cursor will fail with an unknown cursor exception. This mode is useful for applications that fail to close result sets, but does not perform other queries or non-cursor related operations while the result set is being used.
multi	Multi-cursor	Autocommit processing is done by the DBMS when no cursors are open. The JDBC server disables autocommit and begins a standard transaction when a cursor is opened. Since autocommit processing is disabled, multiple cursors may be open at the same time and non-cursor operations are permitted. When a cursor is closed, and no other cursor is open, the JDBC server commits the standard transaction and re-enables autocommit in the DBMS. This mode overcomes the restrictions imposed by the DBMS during autocommit, but requires the application to be very careful in closing result sets. Since the JDBC server will not commit the transaction until all cursors are closed, a cursor left open inadvertently will eventually run into log-file full problems and transaction aborts.

Cursors and Select Loops

By default, the JDBC driver uses a cursor to issue SQL select queries. Cursors permit other SQL operations, such as deletes or updates, to be performed while the cursor is open (operations may be restricted during autocommit, see the Autocommit Transactions section). Cursors also permit multiple queries to be active at the same time. These capabilities are possible because only a limited number of result rows, frequently only a single row, are returned by the DBMS for each cursor fetch request. The low ratio of driver requests to returned rows results in low performance compared to other access methods.

The JDBC driver does use cursor pre-fetch capabilities when possible. For READONLY queries, as many rows as fit in one communications block are retrieved on each fetch request. Depending on row size, this can greatly increase efficiency of the data access. Updateable cursors, however, only retrieve a single row with each fetch request.

The JDBC driver also permits the JDBC application to use a data access method called a select loop. In a select loop request, the DBMS returns all the result rows in a single data stream to the driver. Since select loops use the connection while the result set is open, no other operation or query may be performed until the result set is closed. The statement cancel() method can be used to interrupt a select loop data stream when a result set needs to be closed before the last row is processed. Since the DBMS does not wait for fetch requests from the driver, this access method is the most efficient available.

Select loops may be enabled in the JDBC driver by setting the driver connection property select_loop to a value of 'on' (see the Driver Properties section). With select loops enabled, the driver avoids using cursors for select queries unless explicitly indicated by the application. An application can request a cursor be used for a query by assigning a cursor name to the statement (setCursorName() method) or by using the JDBC syntax select for update ... to request an updateable cursor.

Date/Time Columns and Values

Ingres date literal formats are not supported by the JDBC driver. JDBC specifies the format for date, time, and timestamp literals, using the following escape clause syntax:

Literal	Syntax
date	{d 'yyyy-mm-dd'}
time	{t 'hh:mm:ss'}
timestamp	{ts 'yyyy-mm-dd hh:mm:ss.f...'} f... is optional

Note: Fractional seconds are ignored by the driver.

These escape clauses must be used to include date, time, and timestamp literals in SQL text. Applications may use other date/time formats by using the classes `java.sql.Date`, `java.sql.Time`, `java.sql.Timestamp`, and `java.util.Date` with an appropriately configured date formatter (`java.text.DateFormat`).

The JDBC driver supports Ingres empty dates ('') by returning the JDBC date/time epoch values ('1970-01-01','00:00:00'). A `DataTruncation` warning is created by the driver when an empty date is returned by the methods `getString()`, `getDate()`, `getTime()`, and `getTimestamp()`. An application may check for the warning by calling the `getWarnings()` method after calling one of the previously mentioned methods. An Ingres empty date is different than a `NULL` value, and cannot be detected using the `wasNull()` method.

Ingres interval values are not supported by the methods `getDate()`, `getTime()`, and `getTimestamp()`. An exception will be thrown if an Ingres date column containing an interval value is accessed using these methods. Ingres interval values may be retrieved using the `getString()` method. Since the output of `getString()` for an interval value is not in a standard JDBC date/time format, the JDBC driver creates a warning which may be checked by calling the `getWarnings()` method following the call to `getString()`.

National Character Set Columns

The JDBC driver supports the new Ingres DBMS data types of `nchar`, `nvarchar`, and `long nvarchar`. Retrieval of National Character Set values is done transparently through the existing `getXXX()` `ResultSet` methods.

Tracing

The JDBC driver supports both `DriverManager` and `DataSource` tracing. Trace information consists of JDBC method entrance and exit points with corresponding parameter and return values.

Internal JDBC driver tracing may be enabled by defining system properties on the java command line (-D flag). The following properties are supported:

Property	Value	Description
<code>edbc.trace.log</code>	<i>log</i>	Log file path and name
<code>edbc.trace.id</code>	<i>level</i>	Numeric tracing level

Internal driver tracing permits separate tracing level settings for the following trace IDs (*id*):

Trace ID	Description
IO	Communication I/O modules
EDBC	Standard JDBC interface
EDBCX	Extension JDBC interface

The tracing level determines the type of information that is logged. The following levels are currently defined:

Level	Description
1	Errors
2	High level method invocation
3	High level method details
4	Low level method invocation
5	Low level method details

Managing Your System and Monitoring Performance

Ingres provides two visual tools for managing your system and monitoring performance: Ingres Visual Manager and Visual DBA. Each is described in the following sections.

Ingres Visual Manager

Ingres Visual Manager provides a global view into your Ingres installation. It serves as a system console from which you can manage Ingres components and access other utilities. This utility captures events that are occurring in the system and allows them to be filtered for emphasis, according to the system administrator's preferences.

Ingres Visual Manager is an Ingres-specific utility that allows you to monitor (and start/stop) the different servers in the installation (Bridge, DBMS, JDBC, Net, Star, Internet Communication, Name, Recovery) as well as Remote Command, the Logging and Locking systems, and the Primary Transaction Log and Archiver Process. It shows events occurring in the system both at the installation level and the level of each server. Using Ingres Visual Manager, you can also edit all the Ingres system and user environment variable settings.

Using Ingres Visual Manager

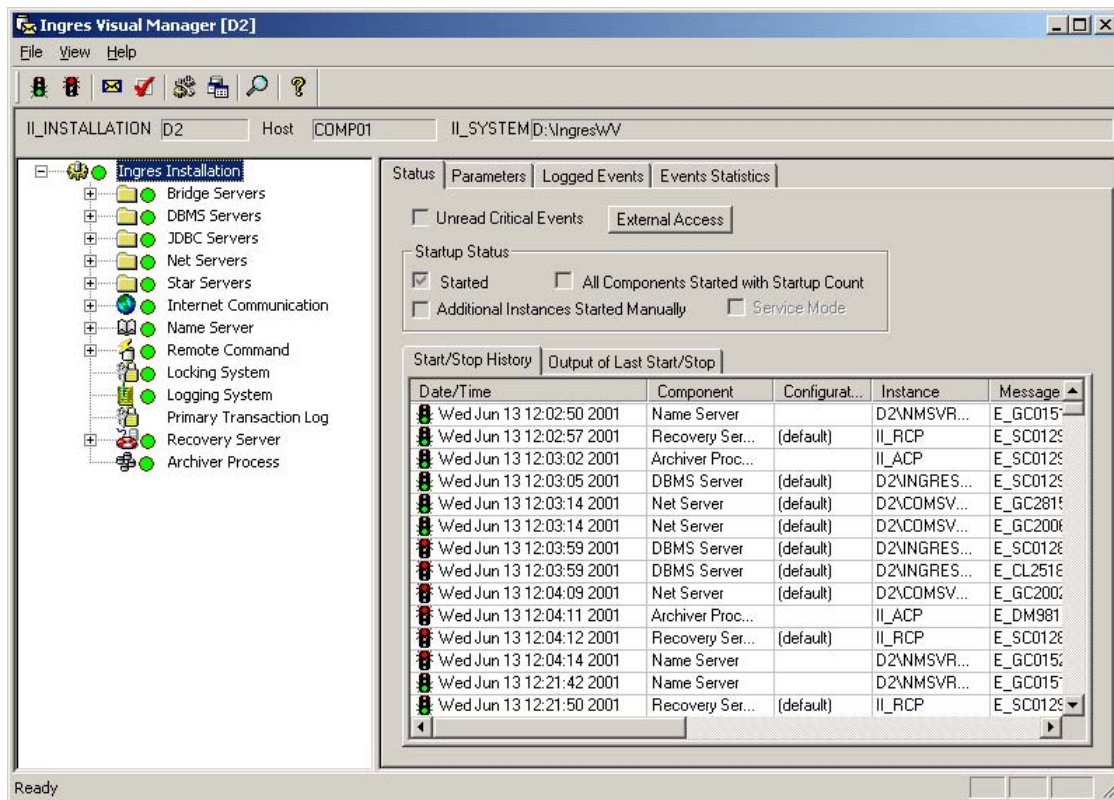
Using Ingres Visual Manager, you can do the following:

- View the started and stopped components and servers in the installation as well as the history of start/stop events
- Start or stop the entire installation or individual components
- View and filter events that occurred both at the installation level and the component or server level
- View statistics on these events
- View and edit Ingres system and user parameters

For more information about each of these functions, see the online help for Ingres Visual Manager.

Ingres Visual Manager Window

The Ingres Visual Manager window contains a list of Ingres components. By selecting any branch of the corresponding tree, you can access information related to the branch, which may correspond either to the entire Ingres installation (the root branch), an individual component configuration, or an individual instance of a given configuration. For example, on the Status page, shown in the following illustration, the Ingres Installation branch is selected, allowing you to view the summary startup status of the components of the installation, the start/stop history of any component in the configuration, and related information such as the output of the last start or stop operation.



On the Parameters page, which appears in the Ingres Visual Manager window when Ingres Installation is selected in the left pane, you can set system, user, or extra parameters.

On the Logged Events page, you can view and acknowledge events.

The Events Statistics page presents statistics on events that have occurred for the selected component, grouped per event ID, and presented as a numerical count and in graph form.

For more information, see the online help for Ingres Visual Manager.

Configuring Parameters

Ingres Visual Manager provides a straightforward way to set parameters.

System and User Parameters

The Parameters page contains lists of parameters for system, user, and extra parameters, along with their values and descriptions. You can add or unset a parameter and edit its value. You can choose whether you want to view unset Ingres parameters that are not currently set.

For more information, see the online help topic, Parameters Page, Ingres Installation branch (Ingres Visual Manager window).

Server Parameters

To configure parameters for each component, select the desired component in the left pane of the Ingres Visual Manager window and click the Configuration Manager toolbar button, or choose File, Configure. For more information, see the online help topic, Configure command (File menu, Ingres Visual Manager).

Defining Message Categories and Notification Levels

Using Ingres Visual Manager, you can define message categories and notification levels. To access the Define Message Categories and Notification Levels dialog, click the Categories and Notification Levels toolbar button, or choose File, Categories and Notification Levels. This dialog allows you to specify which messages in the errlog.log you would like to discard, which you would like to be displayed normally in Ingres Visual Manager, and which you would like to receive a special alert for.

For more information, see online help for Ingres Visual Manager.

Setting Preferences

The Ingres Visual Manager allows you to set various preferences for monitoring events. To access the Preferences dialog, click the Preferences toolbar button, or choose File, Preferences.

Monitoring Components

You can monitor the components that are on your system using Ingres Visual Manager.

Servers

For each of the following components as well as their instances, you can use Ingres Visual Manager to monitor status, logged events, and event statistics:

- Bridge Servers
- DBMS Servers
- JDBC Servers
- Net Servers
- Star Servers
- Internet Communication Server
- Name Server
- Remote Command
- Recovery Server—a Recovery Log File page appears in the Ingres Visual Manager window.
- Archiver Process—an Archiver Log File page appears in the Ingres Visual Manager window.

For more information, see online help for Ingres Visual Manager.

Log Information

The Logged Events page displays a list of the logged events for the Logging System branch, and any of its sub-branches. However, the messages are filtered according to certain preferences you define in the Preferences dialog. For more information, see online help for Ingres Visual Manager.

Lock Information

The Logged Events page displays a list of the logged events for the Locking System. However, the messages are filtered according to certain preferences you define in the Preferences dialog. For more information, see online help for Ingres Visual Manager.

Visual DBA

The Visual DBA Performance Monitor can be used as a monitoring tool, a performance analysis tool, and a system management tool. It displays useful information about the servers, sessions, and locking and logging activities on your Ingres installation.

With the Performance Monitor, you can:

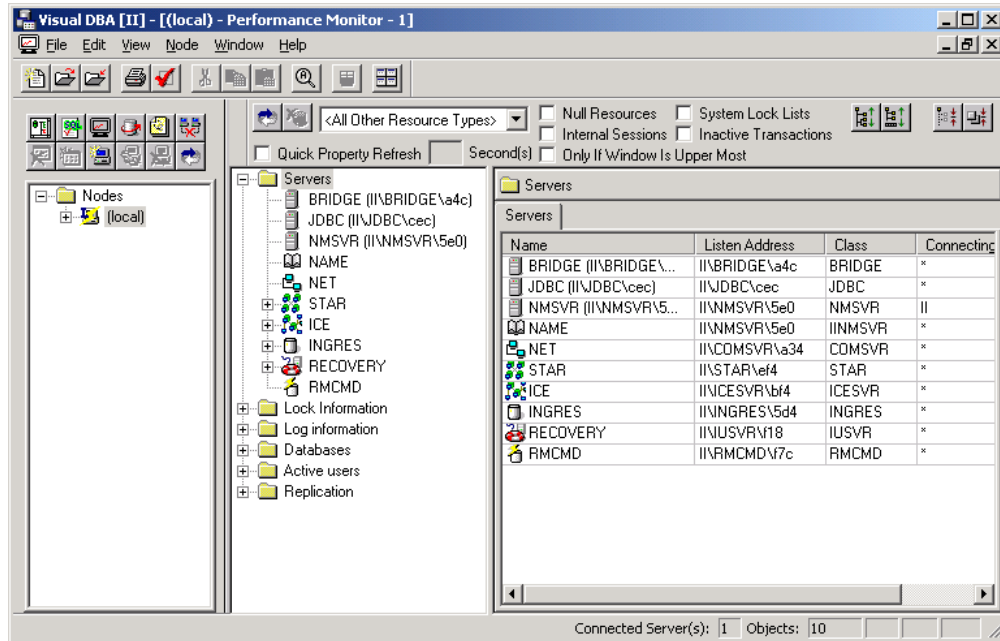
- Monitor information on servers, sessions, and the amount of logging and locking resources being consumed
- Analyze system performance
- Refresh data in the window from a server
- Shut down a database server or close a session
- Monitor replication

Detailed steps for performing these procedures can be found in the Procedures section of the online help for Visual DBA. See the following topics:

- Viewing Performance Information
- Closing a Session

Using the Performance Monitor

Using the Performance Monitor, you can view and monitor information for various Ingres components: Servers, Lock Information, Log Information, Databases, Active Users, and Replication.



Performance Monitor Window

The Performance Monitor window contains a list of categories under which various types of Ingres performance information appears. The Performance Monitor window is used to examine the Ingres system information that is related to performance.

Under each root object category there are sub-branches representing each performance entity. Beneath these branches are sub-branches that pertain to the particular type of performance entity.

Working with Performance Monitor Components

Within Visual DBA, you can view the servers that are started on an Ingres installation and access monitor information about these servers.

Force Refresh or Shut Down Instances

You can force a refresh of the servers or remove/shut down instances of component configurations. To force refresh, right-click the branch representing the component, and choose Force Refresh. To remove or shut down a configuration instance, choose Remove/Shut Down. When a specific server instance has been selected, you are given the opportunity to close or open it.

Background Refresh

You can set up background refreshing, which automatically performs the refresh periodically, based on intervals that you define, through the File Preferences menu command.

Note: The background refresh feature is activated only if View, Activate Background Refresh is chosen.

Monitoring Components

Within Visual DBA, you can monitor the servers that are started on an Ingres installation using the Performance Monitor.

Servers

You can view a list of the servers that are started on an Ingres installation and a quantity of information about each of these servers. For example, you can view the sessions that are currently active for each server. You can also remove a session or stop a server (if you are a privileged user).

Viewing Other Server Information

In addition to examining servers and their sessions, Visual DBA provides additional information for each server. You can access the following information for a particular server:

- Lock lists
- Locks
- Transactions
- Locked databases, tables, and pages
- Other locked resources

For detailed information on monitoring these types of data, press F1 for context-sensitive help on any active dialog or window.

Shutting Down a Server

The recommended way to shut down servers is to use Ingres Visual Manager. For information about performing this task, see the Stop command (File menu, Ingres Visual Manager) topic in Ingres Visual Manager online help.

You can also shut down a server using the Performance Monitor. The “soft” shutdown operation waits for all sessions to end before stopping the server. You may want to first close the sessions (or ask the users of those sessions to close them) before shutting down a server.

Lock Information

Monitoring lock information can be used to determine which lock parameters need to be adjusted. Viewing locking system summaries, lock lists and resources provides you with the information you need to spot conditions when, for example, additional locking system resources may need to be added.

Viewing your *lock lists* is useful for locating transactions that cannot proceed because they are blocked by another transaction.

Log Information

You can view logging system summaries, transaction lists, process, and database lists. Log information can be used to monitor transaction rates, log file activity, processes, and databases in the logging system. This information is useful in determining which logging parameters need to be adjusted.

Databases

Visual DBA allows you to view your performance information from a “database” point of view. This means you can access performance information via database, as opposed to through root branches of the Performance Monitor.

Active User Information

Monitor users for whom there are open sessions currently. You can find out which sessions are open for a particular user, and drill down further to reveal all the related information about those sessions (lock lists, transactions locked, databases locked, and so on).

Replication

You can start, stop, and monitor Replicator servers that are required for the replication scheme that has been defined in a Visual DBA DOM window. You can set up startup parameters for these servers, send events to these servers, view and manage collisions, and display other miscellaneous replication monitor information.

Maintaining Databases

This chapter discusses various database maintenance tools and techniques that are available to the DBA. It includes the following topics:

- Viewing database objects
- Deleting database objects
- Routine maintenance tips
- Operating system maintenance tips
- Verifying databases
- Avoiding user errors
- Translating file names into table names
- Retaining templates of important tables

Databases benefit from optimal maintenance activities. By making use of the information in this chapter, your databases will stay in better condition, and any problems that might develop will be more quickly brought to your attention.

Viewing Database Objects

The DBA must make sure important database objects, such as tables and views, are available; devise a way to separate temporary objects from important objects; and keep private objects to a minimum.

Database objects, such as tables, views, secondary indexes, and synonyms, can be viewed from within the Ingres/Visual DBA Database Object Manager window, as mentioned in several of the previous chapters of this guide. You can view a list of objects using a tree structure, and you can also view specific details for any object in the tree by selecting it and using the panes to the right of the tree structure (see the online help topic Viewing Object Properties for more information).

Note: For information on the SQL statement equivalent to accomplish this task, see the entry for the help statement in the *SQL Reference Guide*. This statement has various options, such as index, table, and view, to allow you to obtain information about various types of database objects.

Impersonating Another User

By default, when you open a Database Object Manager window, the objects belonging to you are visible in the tree structure, but objects belonging to other users are not. To view and work with objects belonging to another user, you need to impersonate that user (which requires the security privilege).

To impersonate another user, select that user from the Users branch in the Virtual Nodes window within Ingres/Visual DBA, then open a Database Object Manager window. The objects belonging to that user and those belonging to the DBA will appear in the window, where you can view and manage them.

The Ingres/Visual DBA Database Object Manager cannot show a list of all tables owned by all users; however, the view `iifile_info`, described below, permits you to select all tables and their owners. For example, the following query lists all user tables not owned by the DBA:

```
select tablename, table_owner, table_type
  from iitables
 where table_owner != '$INGRES' and
        table_owner != 'DBA';
```

Note: For more information on using the `iifile_info` view, see the Translating File Names into Table Names section later in this chapter.

Deleting Database Objects

Database objects, such as tables, views, secondary indexes, and synonyms, can be deleted from within the Ingres/Visual DBA Database Object Manager window. When you drop a table, objects that are directly dependent on that table, such as indexes and views, are automatically dropped as a result.

The online help topic *Dropping Objects* gives a generic description for dropping any type of database object from within the Database Object Manager window, while each type of object has a separate help topic, such as *Dropping a Table* or *Dropping a View*, that is specific to that object type.

While impersonating another user within Ingres/Visual DBA (as described in the previous section), the Database Object Manager window shows the objects belonging to that user. Within this window, you can manage the objects belonging to that user as if they belonged to you, including dropping objects that you deem no longer needed.

Note: For information on the SQL statement equivalent to accomplish this task, see the entry for the drop statement in the *SQL Reference Guide*.

There is another method by which tables can be deleted, if for some reason you cannot drop them from within Ingres/Visual DBA. See the *Verifying Databases* section later in this chapter for more information.

Routine Maintenance Tips

Note: Before making any changes to table structure or database design, please check with this product's local support center for clarification.

It is recommended that you run the following maintenance tools periodically, to keep your tables in good condition:

- Modify database tables periodically if they are subject to frequent updates or inserts. Frequent updates and inserts to all table structures except btree cause overflow data pages to be created, which are inefficiently searched.

If you do not have enough disk space to modify a large btree table, then you can modify the table in order to shrink the btree index. This will tidy the structure of the btree index pages, but does not require the amount of free disk space required by other modify options.

The “Maintaining Storage Structures” chapter of this guide provides specific details on how to modify tables. The Overflow section discusses minimizing overflow pages, and the Shrink Btree Index section tells you how to shrink a btree index that is too large to otherwise modify.

Note: Choosing the correct storage structure for your needs will make maintenance of the database easier. If the storage structure you are using is not the best one, you can modify it using the information in the “Maintaining Storage Structures” chapter of this guide.

- Run system modification on the database if the database is active (that is, users frequently create or modify tables, views, or other database objects). Both system catalog data page overflow and locking contention will be reduced by regular use of system modification.
- Use optimization to help maintain databases. When you optimize a database, data distribution statistics are collected that help queries run more quickly and use fewer system resources. You should optimize your database when the data distribution patterns of a database change.

Optimization should not be run on all columns of all tables in your database. Instead, you should run it only on those columns that are commonly referenced in the where clauses of queries. Collecting more statistics than you need will consume extra disk space and require the Ingres query optimizer to consume more system resources to arrive at an appropriate query execution plan.

Tip: You can set up these routine maintenance tasks to be done inside maintenance batch jobs to avoid the need to run them interactively.

Operating System Maintenance Tips


It is important for you, as DBA, to monitor the operating system. If you are not also the Ingres system administrator, then you should work closely with your system administrator so that you are aware of any operating system problems.

Ingres relies on the operating system in order to access data in tables. If the operating system develops problems, such as system resource shortages, lack of free disk space, or hardware errors, this can affect the responsiveness of the Ingres system and its ability to process requests on behalf of its clients.


Disk errors, memory errors, or operating system resource shortages are the problems most likely to affect the quality of operation. Most hardware errors are dependably logged by the operating system. If you are not the system administrator, make sure that the system administrator is aware of your concern about the efficiency of the operating system.

The operating system offers tools to check and verify the health of the hardware. These include disk drive verification programs and diagnostic programs for memory boards.


Windows

Windows lets the system administrator check for and optionally fix problems in a file system. Free disk space and system configuration can be monitored with the Windows Diagnostics. System-wide performance data, such as CPU usage, can be monitored using the Performance Monitor. Certain system-wide errors and events are monitored in the Event Log, which can be viewed with the Event Viewer. 

UNIX

Most UNIX vendors have a fsck program to check for unreferenced disk blocks, unreferenced inodes, and inconsistencies in operating system tables. Free disk space in your file systems is easily monitored with operating system tools such as df and du. The pstat (BSD) or sar (System V) UNIX commands have options to show the use and distribution of various operating system resources. Every vendor also provides a variety of system maintenance utilities that are menu-driven and easy to use, but which are generally specific to a particular operating system vendor. Make full use of any operating system tools such as these. 

VMS

VMS offers the analyze command which, among other operations, analyzes readability and validity of files and disk volumes. The show device command shows the amount of free disk space. The VMS Monitor Utility (MONITOR) monitors classes of system-wide performance data, such as CPU usage, at a specified interval. These are only a few of the system maintenance utilities that VMS provides. Consult the VMS Help facility and your VMS System Manager for more information on these and other useful operating system tools. 

Verifying Databases

The Verify Database dialog within Ingres/Visual DBA provides the DBA with several operations that can be executed to verify the integrity of a database and to repair certain table-related problems that may arise. To use this dialog, you must be the DBA for all the databases you want to verify, or a user with the security or the operator privilege (as described in the “Ensuring Access Security” chapter).

The Verify Database dialog allows you to verify one or more databases by specifying an operation, then choosing an appropriate scope and mode for that operation. This dialog is quite versatile, allowing you to perform several different operations, including:

- Checking specified tables for inconsistencies and recommending ways to repair them
- Checking database system catalogs for inconsistencies and recommending ways to repair them
- Purging temporary tables, which may be left on the disk inadvertently when the system does not have time to shut down in an orderly fashion (for example, if the machine is rebooted or stops due to power loss)
- Purging expired tables
- Dropping tables that cannot be dropped in the normal manner (for example, if the underlying disk file for the table was deleted at the operating system level) by removing all references to them from the database system catalogs
- Checking the specified databases to determine if they can be and indicates whether the user can connect to the database accessed

For details on how to specify these operations using the Verify Database dialog, see the online help topic *Verifying a Database*.

Avoiding User Errors

Follow these rules for databases that are shared among multiple users:

- Have users use only application programs to access data in the database. Discourage users from using Ingres tools, such as the Terminal Monitor or Ingres/Visual DBA, to access data. Permitting users to access data only by means of an application program guarantees that the executing queries were written by an application programmer and are not ad hoc queries that could damage or delete data, or cause lock contention delays.

- Ensure that reports are run with readlock = nolock. This may be done by including all reporting tools within application programs and setting readlock there, or by running all reports from operating system scripts which set lockmode before the report runs. This avoids locking contention problems that could lead to severe concurrent performance problems in the database.

See the Setting Readlock section in the “Ingres Locking” chapter of this guide for the advantages and drawbacks of using readlock=nolock.

Translating File Names into Table Names

A naming algorithm is used to assign underlying file names for tables. There are two columns in the iirelation table used to produce names:

- reltid, a unique table identifier assigned in sequential order
- reltidix, a unique index identifier associated with each base table

The algorithm for creating the name is as follows:

1. Convert reltid (for base tables) or reltidix (for secondary indexes) to an 8-digit hexadecimal number.
2. Assign letters to each of the resulting hexadecimal digits:
0,1, 2, ..., F is assigned to a, b, c, ..., p

For example, a reltid of 129 converted to an 8-digit hex number is “00000081”. Substituting letters gives a file name of aaaaaaib.tnn, where nn=00, 01, ..., for first (or only) location, second location, and so on.

As the DBA, you can use the view iifile_info to select the names of the disk files associated with tables, as shown in the example below:

```
select table_name, owner_name, file_name, file_ext
from iifile_info;
```

Retaining Templates of Important Tables

It is good practice to periodically generate copy scripts for important tables and views. The copy.in scripts produced will be useful in the future if you need to recreate new, empty tables, or the entire database. Generating copy scripts is accomplished using the Generate copy.in and copy.out dialog within Ingres/Visual DBA.

Maintaining Storage Structures

Note: Before making any changes to table structure or database design, please check with this product's local support center for clarification.

Maintaining good performance is one of the major responsibilities of the DBA. Performance issues with respect to storage structures include modifying the database tables, compressing storage structures, and managing overflow.

This chapter discusses how and when to use the Visual DBA modify procedures to change storage structures for tables and secondary indexes. Modify procedures should be used as part of regular system maintenance to get rid of overflow pages and recover disk space for deleted rows.

For additional information on database performance, see the chapter, "Improving Database and Query Performance," in this guide.

Storage Structures and Performance

The data for each table is stored in a file on disk. Tables consist of pages with a size that you define when you create the table. For example, you can specify a page size of 2 KB, 4 KB, and so forth, by powers of two up to 64 KB. Each page has a certain amount of overhead, which depends upon the page size.

Each page stores a number of rows. The number of rows per page varies, according to the row width, the storage structure of the table, whether or not the table is compressed, and how much data has been added or deleted since the table was last modified. Rows cannot span pages, limiting the maximum row width to the per-page data size.

The page is an important concept in understanding query performance because it affects the amount of disk I/O a query does, as well as the amount of CPU resources required to read through a table.

To see how many pages there are in a table, use Visual DBA to select a table, then select the Pages tab.

Note: For information on the SQL statement equivalent to accomplish this task, see the help table statement in the *SQL Reference Guide*.

A B-tree table is shown in the following example:

```
Name: emp
Owner: ingres
Created: 23-sep-1998 10:27:00
Location: ii_database
Type: user table
Version: II2.5
Page size: 2048
Cache priority: 0
Alter table version: 0
Alter table totwidth: 70
Row width: 70
Number of rows: 32
Storage structure: B-tree
Compression: none
Duplicate Rows: not allowed
Number of pages: 6
Overflow data pages: 0
Journaling: enabled
Base table for view: yes
Permissions: none
Integrities: none
Optimizer statistics: none
Column Information:
```

Column Name	Type	Length	Nulls	Defaults	Key Seq
name	varchar	20	no	no	1
title	varchar	15	no	yes	
hourly_rate	money		no	yes	
manager	varchar	20	yes	null	

Secondary indexes: none

Limitations of Heap Structure

Without help from the storage structure, when you want to retrieve a particular row from a table, you must search through every row in the table looking for rows that qualify. (Searching through every row is called *scanning* the table.) Stopping at the first row that qualifies is not enough, since multiple rows may qualify.

Consider the data shown in a sample heap table:

	empno	name	age	salary	comment
Page 0	17	Shigio	29	28000.000	
	9	Blumberg	33	32000.000	
	26	Stover	38	35000.000	
	1	Mandic	46	43000.000	
Page 1	18	Giller	47	46000.000	
	10	Ming	23	22000.000	
	27	Curry	34	32000.000	
	2	Ross	50	55000.000	
Page 2	19	McTigue	44	41000.000	
	11	Robinson	64	80000.000	
	28	Kay	41	38000.000	
	3	Stein	44	40000.000	

Page 3	20	Cameron	37	35000.000	
	12	Saxena	24	22000.000	
	29	Ramos	31	30000.000	
	4	Stannich	36	33000.000	
Page 4	21	Huber	35	32000.000	
	13	Clark	43	40000.000	
	30	Brodie	42	40000.000	
	5	Verducci	55	55000.000	
Page 5	22	Zimmerman	26	25000.000	
	14	Kreseski	25	24000.000	
	31	Smith	20	10000.000	
	6	Aitken	49	50000.000	
Page 6	23	Gordon	28	27000.000	
	15	Green	27	26000.000	
	7	Curan	30	30000.000	Fire
	24	Sabel	21	21000.000	
Page 7	16	Gregori	32	31000.000	
	8	McShane	22	22000.000	
	25	Sullivan	38	35000.000	

With this heap structure, a retrieval such as the following would have to look at every page in the emp table:

```
select * from emp where emp.name = 'Sullivan';
```

Although Shigio's record is the first row in the table, the following retrieval would also have to look at every row in the table:

```
select * from emp where emp.name = 'Shigio';
```

Since the table is not sorted, the entire table must be scanned in case there is another employee named Shigio on another page in the table.

A retrieval from a large table can be expensive in time and system resources. To understand the performance consequences of a scan of a large table, assume that the emp table is actually 300,000 pages, rather than 8. Further, assume the disks can manage approximately 30 disk I/Os per second. Assume one disk I/O per page. With a heap storage structure, the above select would take $300,000 / 30 = 10,000$ seconds (or 2 hours, 46 minutes) just in disk access time alone, not counting the CPU time it would take to scan each page once it was brought in from disk, and assuming no other system activity.

For a large table, a different storage structure is needed. A production system cannot tolerate a three-hour wait to retrieve a row. The solution is to provide a storage structure that allows for keyed access, like hash, isam, or B-tree.

Modifying Storage Structures

You can change tables to a more effective storage structure. To change a table from one storage structure to another, use the Modify Table Structure dialog. By enabling the Change Storage Structure radio button and clicking Structure, you activate the Structure of Table dialog, where you can specify the parameters for the storage structure type and other structure-specific characteristics. For secondary indexes, the Modify Index Structure dialog offers a similar option to enable the Structure of Index dialog. See the online help topic Modifying Storage Structures for more information.

Key columns must be specified for hash, isam, and B-tree (heap and heapsort tables do not have key columns). There is no limit to the number of key columns that can be specified, but there is a slight decline in performance as the key columns increase.

Note: For information on the SQL statement equivalents for accomplishing this task, see the modify statement in the *SQL Reference Guide*.

Modify Cautions

Keep in mind the following effects of the modify procedure when you are modifying the storage structure:

- **Locking**—During the modify procedure, the table is exclusively locked and inaccessible to other users.
- **Secondary Indexes**—Secondary indexes are destroyed when you modify the base table storage structure. For details, see the Modifying Secondary Indexes section.
- **Disk Space**—When a table storage structure is modified, temporary sort files are created. Before the old table can be deleted, a new table must be built. Once it is completely built, the old table is deleted, and the temporary file is renamed with the old table name.

Modify Options

The Structure of Table and Structure of Index dialogs provide several options: Min Pages, Max Pages, Allocation, Extend, Fillfactor, Leaffill, Nonleaffill, Unique, and Compression, which are discussed in the following sections. Some of these options take effect only during the modify procedure (Min Pages, Max Pages, Fillfactor, and Nonleaffill), while others are enforced for the life of the table (Allocation, Extend, Unique, and Compression).

Controlling the Number of Pages

Min Pages and Max Pages are valid options only when you are modifying the table to hash. These options allow you to control the hashing algorithm to some extent, extending the control offered by the Fillfactor option.

The Min Pages option is useful to specify if the table is going to be growing rapidly or if you want few rows per page in order to increase concurrency so multiple people can update the same table. (You can achieve nearly the same effect by specifying a low value for the Fillfactor option, but the fill factor is based on the current size of the table. See Specifying Alternate Fill Factors below.)

To force a specific number of main pages, use the Min Pages option to specify a minimum number of main pages. The number of main pages used will be at least as many as specified, although the exact number of Min Pages specified might not be used.

As an example, for the emp table, you could force a higher number of main pages by specifying the minimum number of main pages when you modify the table to hash. If you specify 30 main pages for the table, which has 31 rows, you have approximately one row per page. To modify the storage structure of the emp table, you would enter the following values in the Structure of Table dialog, assuming it is already open for the emp table. (See the online help for more information.)

1. Select Hash from the Structure drop-down list.
2. Enter **30** in the Min Pages edit control.
3. Enable the age column in the Columns list.

To specify a maximum number of main pages to use, rather than the system choice, use the Max Pages option. If the number of rows does not completely fit on the number of pages specified, overflow pages are allocated. If fewer pages are needed, the lesser number is used. Max Pages is useful mainly for shrinking compressed hash tables more than might otherwise happen. (You can achieve nearly the same effect by specifying a high value for the Fillfactor option, but the fill factor is based on the current size of the table. See Specifying Alternate Fill Factors below.)

The following example modifies the emp table, specifying a Max Pages value. The steps assume the Structure of Table dialog is already open for the emp table. (See the online help for more information.)

1. Select Hash from the Structure drop-down list.
2. Enter **100** in the Max Pages edit control.
3. Enable the empno column in the Columns list.

Remember that Max Pages controls only the number of main pages; it does not affect overflow pages. For example, assume your data takes 100 pages in heap. If you modify to hash and limit the number of main pages to 50, then the remainder of the data goes onto overflow pages.

Preallocating Space

Use the Allocation option to preallocate space. You can modify the table to an allocation greater than its current size to leave free space within the table. (The default is four pages if no allocation has been specified.) This allows you to avoid a failure due to lack of disk space, or to provide enough space for table expansion instead of having to perform a table extend operation (see the Extending a Table or Index section later in this chapter for more information).

For example, the following example specifies that 1000 pages be allocated to table inventory. The steps assume the Structure of Table dialog is already open for the inventory table. (See the online help for more information.)

1. Select B-tree from the Structure drop-down list.
2. Enter **1000** in the Allocation edit control.

The size must be in the range 4 to 8,388,607 (the maximum number of pages in a table). The specified size is rounded up, if necessary, to make sure the allocation size for a multi-location table or index is always a multiple of sixteen. For example, the actual space allocated in the preceding example is 1008.

Note: If the specified number of pages cannot be allocated, the modify procedure is aborted.

Once an allocation is specified, it remains in effect and does not need to be specified again when the table or index is modified.

Extending Space

The Extend option allows you to control the amount of space by which a table is extended when more space is required. (The default extension size is 16 pages.) The following example specifies that the table inventory be extended in blocks of 1000 pages. The steps assume the Structure of Table dialog is already open for the inventory table. (See the online help for more information.)

1. Select B-tree from the Structure drop-down list.
2. Enter **1000** in the Extend edit control.

The size must be in the range 1 to *max_size*, where the *max_size* is calculated as:
 $8,388,607 - allocation_size$

The specified Extend size is rounded up, if necessary, to make sure the size for a multi-location table or index is always a multiple of sixteen. For example, the actual extension space in the above example is 1008.

Note: If the specified number of pages cannot be allocated, the operation will fail with an error.

Once an extend size has been specified for the table or index, it remains in effect and does not need to be specified again when the table or index is modified.

When choosing an extend size, keep the following in mind:

- When extending a table, not only the physical extension must be performed, but the extension must also be recorded. Therefore, avoid an excessively small extend size that requires many additional small extensions.
- In an environment that is short of disk space, a large extend size may cause an operation to fail, even when there is sufficient disk space for the particular operation.

Windows

- On a file system that requires the underlying files to be written to when allocating disk space (such as Windows NT), a large extend size may be undesirable because it will affect the performance of the operation that causes the extend. ■

UNIX

- On a file system that requires the underlying files to be written to when allocating disk space (such as UNIX), a large extend size may be undesirable because it will affect the performance of the operation that causes the extend. ■

VMS

- On file systems that provide calls for allocating disk space (such as VMS), a large extend size will help reduce the amount of table fragmentation. ■

Keeping Default Fill Factors

Each storage structure has a different default fill factor. The term *fill factor* refers to the number of rows that are actually put on a data page divided by the number of rows that could fit on a data page for a particular structure.

The various fill factors enable you to add data to the table without running into overflow problems. Since the data pages have room to add data, you do not have to remodify.

For instance, a heap table fits as many rows as possible on a page; this is known as 100% fill factor. However, isam and B-tree data pages are filled only to 80% capacity, leaving room to add 20% more data before a page is completely full. The default data page fill factors are shown in the following table:

Storage Structure	Default Fill Factor	Multiply Heap Size by	Number of Pages Needed for 100 Full Pages
B-tree	80%	1.25	125 + index pages
compressed B-tree	100%	1	100 + index pages
hash	50%	2	200
compressed hash	75%	1.34	134
heap	100%	1	100
compressed heap	100%	1	100
isam	80%	1.25	125 + index pages
compressed isam	100%	1	100 + index pages

The default B-tree index page fill factors are shown in the following table:

Storage Structure	Default Fill Factor
B-tree leaf	70%
B-tree index	80%

The first table shows that if a heap table is 100 pages and you modify that table to hash, the table now takes up 200 pages, since each page is only 50% full. (Note that the number of pages may be approximate, depending on the system allocation for tracking used and free pages.)

Specifying Alternate Fill Factors

In the Structure of Table and Structure of Index dialogs, you control the fill factor of the data pages using the Fillfactor option.

You can tailor the fill factor for various situations. For instance, if the table is not going to grow at all, you might use a 100% fill factor for the table. On the other hand, if you know you are going to be adding a lot of data, you may wish to use a low fill factor, perhaps 25%. Also, if your environment is one where updates are occurring all the time and good concurrency is important, you may wish to set the fill factor low.

Note: Fill factor is used only at modify time. As you add data, the pages fill up and the fill factor no longer applies.

When specifying a fill factor other than the default, you should keep the following points in mind:

- Use a high fill factor when the table is static and you are not going to be appending many rows.
- Use a low fill factor when the table is going to be growing rapidly. Also, use a low fill factor to reduce locking contention and improve concurrency. A low fill factor distributes fewer keys per page, so that page level locks lock fewer records.

Specifying fill factor is useful for hash and isam tables; but because only data pages are affected, the Fillfactor option should be done in conjunction with the Leaffill or Nonleaffill options for B-tree tables. (See the next two sections.)

Normally for hash tables, a 50% fill factor is used for uncompressed tables. You can raise or lower this, but raising it too high may cause more overflow pages than would be desirable. You should always measure the overflow in a hash table when setting a high fill factor — fill factors higher than 90% are very likely to cause overflow.

This example sets the fill factor on a hash table to 25%, rather than the default of 50%, by modifying the emp table. The steps assume the Structure of Table dialog is already open for the emp table. (See the online help for more information.)

1. Select Hash from the Structure drop-down list.
2. Enter **25** in the Fillfactor edit control.
3. Enable the empno column in the Columns list.

If you are using compressed isam tables and are adding data, make sure you set the fill factor to something lower than the default 100%, or you immediately add overflow pages.

Normally, uncompressed isam tables are built with an 80% fill factor. You can set the fill factor on isam tables to 100%, and unless you have duplicate keys, you should not have overflow problems until after you add data to the table, as shown in the example below. The steps assume the Structure of Table dialog is already open for the emp table. (See the online help for more information.)

1. Select Isam from the Structure drop-down list.
2. Enter **100** in the Fillfactor edit control.
3. Enable the name column in the Columns list.

Specifying Leaf Page Fill Factors

It is possible to specify B-tree leaf page fill factors at modify time. This is the percentage of the leaf page that will be used during the modify procedure. The remaining portion of the page is available for use later when new rows are added to the table. The purpose of the fill factor is to leave extra room on the leaf pages to do inserts without causing leaf page splits. This is useful if you modify a table to B-tree and plan to add rows to it later. In the Structure of Table dialog, you control these values using the Leaffill options.

The Leaffill option specifies the percentage of each leaf page to be filled at the time the table is modified to B-tree or cB-tree. The Leaffill default is 70, which means that 70% of the leaf page is filled at modify time and 30% remains empty for future use.

For example, assume that the key-tid pair requires 400 bytes of storage. This means that five key-tid pairs should fit on a single 2KB B-tree leaf page. However, if the leaf page fill factor is specified at 60%, then only three key-tid pairs are allocated on each B-tree leaf page at modify time. If subsequent updates to the table cause two new rows on this leaf page, they are placed in the empty space on the leaf page. The key-tid pairs are reordered on the leaf page from min to max. If more than two new rows need to be added to this leaf page, then there is not enough space and the leaf page will have to split.

Specifying Index Page Fill Factors

It is possible to specify B-tree index page fill factors at modify time. This is the percentage of the index page that will be used during the modify procedure. The remaining portion of the page is available for use later when new rows are added to the table. The purpose of the fill factor is to leave extra room on the index pages to do inserts without causing index page splits. This is useful if you modify a table to B-tree and plan to add rows to it later. In the Structure of Index dialog, you control these values using the Nonleaffill options.

The Nonleaffill option specifies the percentage of each index page that is to be filled at the time the table is modified to B-tree. That is, it is similar to Leaffill, but for index pages instead of leaf pages. The Nonleaffill default is 80. This means that 80% of the index page is used at modify time and 20% remains empty for future use.

For example, assume that the key-tid pair requires 500 bytes of storage. This means that four key-tid pairs should fit on a single B-tree index page. However, if the index page fill factor is specified at 75%, then only three key-tid pairs are allocated on each 2KB B-tree index page at modify time. If subsequent updates to the table cause another leaf page to be allocated, then the empty space on the index page is used to hold a key-tid pair for that new leaf page. If there were enough new rows to cause two new leaf pages to be added to that index page, then the index page would have to split.

Setting a fill factor of lower than 60 on leaf pages can help reduce locking contention when B-tree leaf pages are splitting, since index splitting is reduced. Setting Leaffill low for small but quickly growing B-trees is advisable.

When you specify a high Leaffill, index splitting is almost guaranteed to occur because leaf pages immediately fill up when data is added. Thus you want to avoid a high fill factor unless the B-tree table is relatively static. Even in this case you might be better off with an isam table.

Ensuring Key Values Are Unique

Unique keys can be enforced automatically for hash, isam, and B-tree tables using the modify procedure. Unique keys can be specified as Row or Statement in the Unique group box in the Structure of Table and Structure of Index dialogs:

- Row indicates that uniqueness is checked as each row is inserted.
- Statement indicates that uniqueness is checked after the update statement is executed.

If you do not want to create a unique key, select the No option.

The following example prevents the addition of two employees in the emp table with the same empno. The steps assume the Structure of Table dialog is already open for the emp table. (See the online help for more information.)

1. Select Isam from the Structure drop-down list.
2. Enable Row in the Unique radio button group box.
3. Enable the empno column in the Columns list.

If a new employee is added with the same employee number as an existing record in the table, the row is not added, and you are returned a row count of zero.

Note: An error is not returned in this case; only the row count shows that the row was not added. Be aware of this if you are writing application programs using unique keys.

Benefits of unique keys are:

- A good database design that provides unique keys enhances performance.
- You are automatically ensured that all data added to the table has unique keys.
- The Ingres optimizer recognizes tables that have unique keys and uses this information to plan queries wisely.

The disadvantages of unique keys include a small performance impact in maintaining uniqueness. You must also plan your table use so that you will not add two rows with the same key value. In most cases, unique keys are a decided advantage in your data organization.

The following example modifies the emp table to hash and prevents the addition of two employees in the emp table with the same empno. The steps assume the Structure of Table dialog is already open for the emp table. (See the online help for more information.)

1. Select Hash from the Structure drop-down list.
2. Enable Row in the Unique radio button group box.
3. Enable the empno column in the Columns list.

The rows in the following example have unique keys. Although employee #17 and #18 have the same records except for their employee numbers, the employee numbers are unique, so these are valid rows after the modification:

Empno	Name	Age	Salary
17	Shigio	29	28000.000
18	Shigio	29	28000.000
1	Aitken	35	50000.000

The following two rows do not have unique keys. These two rows could not both exist in the emp table after modification to hash unique on empno:

Empno	Name	Age	Salary
17	Shigio	29	28000.000
17	Aitken	35	50000.000

Compressing Tables

All storage structures permit tables and indexes (where present) to be compressed, except the r-tree secondary index storage structure and the heapsort storage structure, which cannot be compressed. Compression is controlled using the Key and Data options in the Compression group box in the Structure of Table and Structure of Index dialogs. By default, there is no compression when creating or modifying, indicated by the fact that no Compression options are enabled.

Not all parts of all storage structures can be compressed, as summarized in the table below:

Storage Structure		Data	Key
B-tree	Base Table	Yes	Yes
	Secondary Index	No	Yes
hash	Base Table	Yes	No
	Secondary Index	Yes	No

Storage Structure		Data	Key
heap	Base Table	Yes	No
	Secondary Index	N/A	N/A
heapsort	Base Table	No	No
	Secondary Index	N/A	N/A
isam	Base Table	Yes	No
	Secondary Index	Yes	No
r-tree	Base Table	N/A	N/A
	Secondary Index	No	No

Note: Selecting Data in the Compression group box in the Structure of Table dialog does not affect keys stored in isam or B-tree index and leaf pages—only the data on the data pages is compressed. To compress index entries on B-tree index pages, select Key instead.

Isam index pages cannot be compressed.

Compression of tables compresses character and text columns. Integer, floating point, date, and money columns are not compressed, unless they are nullable and have a null value.

Trailing blanks and nulls are compressed in character and text columns. For instance, the emp table contains a comment column that is 478 bytes. However, most employees have comments that are only 20 to 30 bytes in length. This makes the emp table a good candidate for compression because 478 bytes can be compressed into 30 bytes or fewer, saving nearly 450 bytes per row.

Furthermore, as many rows are placed on each page as possible, so that the entire emp table (31 rows) that normally took eight 2KB pages as a heap, takes just one page as a compressed heap. In this example, pages were limited to four rows per page, but by using compression, many more rows could be held per page.

There is no formula for estimating the number of rows per page in a compressed table, as this is entirely data dependent.

When to Compress a Table

When a table is compressed, you can reduce the amount of disk I/O needed to bring a set of rows from disk. This can increase performance if disk I/O is a query-processing bottleneck.

For instance, having compressed the emp table from eight pages down to one page, the following query performs only one disk I/O, whereas prior to compression as many as eight disk I/Os may have been required:

```
select * from emp;
```

In a large table, compression can dramatically reduce the number of disk I/Os performed to scan the table, and thus dramatically improve performance on scans of the entire table. Compression is also useful for conserving the amount of disk space it takes to store a table.

But compression should be used wisely, as the overhead associated with compression can sometimes exceed the gains. If a machine has a fast CPU, disk I/O can be the bottleneck for queries. However, because compression incurs CPU overhead, the benefits must be weighed against the costs, especially for machines with smaller CPUs. Compression can increase CPU usage for a query because data must be decompressed before it is returned to the user. This increase must be weighed against the benefits of decreased disk I/O and how heavily loaded the CPU is. High compression further reduces disk I/O, but uses even more CPU resources.

There is overhead when updating compressed tables. As rows are compressed to fit as many as possible per page, if you update a row so that it is now larger than it was before, it must be moved to a new spot on the page or even to a new page. If a row moves, its tid, or tuple identifier, also changes, requiring that every secondary index on the compressed table also be updated to reflect the new tid.

For instance, if you change Shigio's comment from "Good" to "Excellent," Shigio's record length grows from 4 bytes to 9 bytes and does not fit back in exactly the same place. His record may need to be moved to a new place (or page), with updates made to any secondary indexes of this table (if the emp table was B-tree, the appropriate B-tree leaf page would be updated instead).

Compressed tables should be avoided when updates that increase the size of text or character columns occur frequently, especially if there are secondary indexes involved—unless you are prepared to incur this overhead. If you do compress and are planning to update, use a fill factor lower than 100% (75% for hash); the default fill factor for compressed tables is 75% for hash with data compression, 100% for the others. With free space on each page, moved rows are less likely to be placed on overflow pages. For more information, see the Fill Factor section, earlier in this chapter.

Page Size

Page size was discussed earlier in this chapter in the section entitled Storage Structures and Performance. In Visual DBA, you specify page size using the Page Size option in the Structure of Table and Structure of Index dialogs.

The default page size is 2 KB. The corresponding buffer cache for the installation must also be configured with the page size you specify or you will get an error.

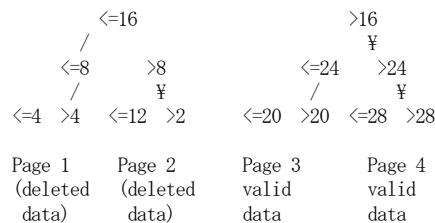
Shrinking a B-tree Index

In order to maintain good concurrency and performance, the B-tree index is not rebuilt after deletions. Deletions occur at the leaf and data page level, but an empty leaf page is not released. If your environment is one where many deletions are performed, you should occasionally update the index using the Shrink B-tree Index option in the Modify Table Structure and Modify Index Structure dialogs.

The Shrink B-tree Index option is also important for users with incremental keys, which may incur lopsided indexes after heavy appends to the end of the table.

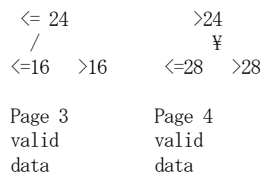
Not updating the index to reflect unused leaf pages can cause the index to be larger than it might otherwise be. For example, if the emp table is keyed on empno (ranging from 1 to 31), and you fire all employees with employee numbers less than 16, the B-tree index would not shrink but would be unbalanced. This is shown in the Before diagram that follows:

Before:



To rebalance the index level, you can use the Shrink B-tree Index option. It also reclaims unused leaf pages that would otherwise never be reused. This is illustrated in the After diagram that follows:

After:



Free page list: 1, 2

The index is rebuilt, and empty leaf pages are marked as free, but otherwise leaf and data pages remain untouched. Therefore, this procedure is neither as time-consuming nor as disk-space intensive as modifying the table structure using the Change Storage Structure option. Shrink B-tree Index, however, does not resort the data on the data pages. Modifying the structure to B-tree is the only option for resorting data on data pages.

Note: For information on the SQL statement equivalent to accomplish this task, see the modify statement in the *SQL Reference Guide*. The to merge clause is the same as the Shrink B-tree Index option.

Extending a Table or Index

Within Visual DBA, you can extend (add pages to) a table or index by enabling the Add Pages radio button in the Modify Table Structure or Modify Index Structure dialogs, then specifying the number of pages you want to add. Using this option does not rebuild the table or drop any secondary indexes.

Note: For information on the SQL statement equivalent to accomplish this task, see the modify statement in the *SQL Reference Guide*. The to extend clause is the same as the Add Pages option.

Modifying Secondary Indexes

Secondary indexes are destroyed by default when you modify the base table storage structure. They are destroyed automatically because secondary indexes use the tidp column to reference the row of the base table to which they are pointing. When you modify a table, all the tids of the rows in the base table change, rendering the secondary index useless.

Persistence Option

You can use the Persistence option when creating or modifying a secondary index to specify that the index be recreated whenever the base table is modified. This option is found in the Structure of Index and the Create Indexes dialogs. By default, indexes are created with no persistence.

Note: For information on the SQL statement equivalents to accomplish this task, see the create index and modify statements in the *SQL Reference Guide*. The [no]persistence clause is the same as the Persistence option.

For example, assuming the secondary index empidx was created without enabling the Persistence option, you could modify it as shown in the following example to enable this feature. The steps assume the Structure of Index dialog is already open for the empidx index. (See the online help for more information.)

1. Select B-tree from the Structure drop-down list.
2. Enable the Persistence check box.

Changing the Index Storage Structure

The default storage structure for secondary indexes is isam; you can choose a different structure when creating an index using the Create Indexes dialog. You can also modify the index to another storage structure after it has been created, using the Structure of Index dialog.

The following example creates a B-tree index for the emp table. The steps assume the Create Indexes dialog is already open for the emp table:

1. Enter an appropriate name in the Index Name edit control.
2. Select B-tree from the Structure drop-down list.
3. Select an appropriate key column in the Base Table Columns list box, and click the double-right arrow (>>) to add the column to the Index Columns list box.

The next example shows how you would modify an existing index to use the B-tree storage structure, assuming it was created using another storage structure. The steps assume the Structure of Index dialog is already open for the index. (See the online help for more information.)

1. Select B-tree from the Structure drop-down list.
2. Enable the appropriate column(s) in the Columns list.

If a secondary index is modified to B-tree, it will not contain any data pages. Instead, the leaf pages in the secondary index point directly to data pages in the main table.

Overflow can occur in secondary indexes, as well as base tables, and should be monitored. One way to handle overflow is to use B-tree as the default index structure. If overflow is not a problem, isam or hash may be preferable because the indexes are smaller, require less locking, and reuse deleted space.

Secondary indexes are smaller and can be modified more quickly than the base table. When they are used, overflow occurs less frequently since only key values are stored, rather than the entire row.

Because it is quicker to build secondary indexes than to modify the base table, it is easier to experiment with different choices of secondary indexes and different storage structures for them. Remember, however, that it will take longer to update a table with secondary indexes than one without them.

A high degree of duplication in a secondary key can lead to overflow in the secondary index. Repetitive keys are not recommended. Performance benefits can be derived by the inclusion of another column in the secondary index that makes the entire key less repetitive. The less repetitive key reduces the likelihood of overflow chains, resulting in better performance when updates made to the base table require updates to the secondary index. Because overflow chains are reduced, locking and searching overhead is lessened.

If the secondary index to be stored is isam or B-tree and the key is not unique, the tidp column is automatically included in the key specified when the index is modified. This achieves key uniqueness without any loss of functionality when the key is used for matches.

When to Remodify B-tree Tables

If you suspect that the data on the data pages is scattered over several data pages, you can modify the table to B-tree again. You can check this by retrieving the tids as well as the column values, and looking at the pages they reflect. Remodifying sorts the data and then builds the B-tree index, placing like keys on the same data pages, which can slightly reduce the number of disk I/Os required to access the data.

This type of modification is especially useful when the key size is small, the row size is large, and the data has not been appended in sorted order. Remodifying a B-tree is also useful when you have deleted many rows and wish to reclaim disk space.

Examples of a modification to B-tree follow. The first example represents the table before modification, and the second example shows it after modification.

Note that the following retrieval touches all three data pages before modification but only one page after modification:

```
select * from emp where emp.age = 35;
```

The following table shows the leaf and data pages prior to modification. Note that the records with a key of 35 are found on several data pages:

Leaf Page		
key	page, row	(tid)
35	1, 2	(514)
35	2, 2	(1026)
35	3, 3	(1539)
36	2, 3	(1027)
37	3, 2	(1538)

Data Pages		
Page 1	Page 2	Page 3
1, 1 (513) 29	2, 1 (1025) 29	3, 1 (1537) 30
1, 2 (514) 35	2, 2 (1026) 35	3, 2 (1538) 37
1, 3 (515) 30	2, 3 (1027) 36	3, 3 (1539) 35

The following example modifies the emp table, respecifying B-tree as its structure. The steps assume the Structure of Table dialog is already open for the emp table. (See the online help for more information.)

1. Select B-tree from the Structure drop-down list.
2. Enable the age column in the Columns list.

After you perform this modification, the table looks as follows. Note that all records with a key of 35 are clustered together on Page 2:

Page 1	Page 2	Page 3
1, 1 (513) 29	2, 1 (1025) 35	3, 1 (1537) 36
1, 2 (514) 29	2, 2 (1026) 35	3, 2 (1538) 37
1, 3 (515) 30	2, 3 (1027) 35	

Common Errors During the Modify Procedure

When using the modify procedure in Visual DBA, the most common errors include:

- A “duplicate key” error message when you use the Unique option (or the to unique clause of the modify statement). To resolve this problem, determine which rows have duplicate keys and delete these rows.

You can locate these rows with the following query:

```
select key_col, count(*) as repeat_number
  from table_name
 group by key_col
 having count(*) > 1;
```

- An error when modifying a table. You may be completely out of disk space on the file system the modify procedure is trying to use. Clear up disk space on this file system.

Overflow

Overflow chains can considerably slow down performance and should be monitored and, as much as possible, prevented. The sections that follow explain how to monitor overflow, describe and give examples of some of the causes of overflow, and discuss overflow in relation to the different types of storage structures.

Preventing or reducing overflow requires you to do the following:

- Carefully monitor overflow in both primary tables and secondary indexes
- Avoid the use of repetitive keys, including both primary keys and secondary index keys
- Modify table structure to redistribute poorly distributed overflow
- Understand the overflow implications when choosing a particular storage structure

Measuring the Amount of Overflow

To monitor overflow in Visual DBA, select a table or secondary index in the Database Object Manager window, and click the Pages tab. For tables, overflow data is displayed in red in the pie chart, as indicated in the legend. Heap tables are considered as one main page, with an overflow chain attached to the main page. For B-tree tables, overflow occurs only at the leaf level and only with duplicate keys.

Note: For information on the SQL statement equivalent to monitor overflow, see the help table statement in the *SQL Reference Guide*.

The iitables catalog (a view into the iirelation catalog) includes one row for each table in the database. It contains pertinent information for evaluating overflow. For example, the following query results in the information shown in the table:

```
select table_name, storage_structure,
       number_pages, overflow_pages
from iitables
```

table_name	storage_structure	number_pages	overflow_pages
manager	hash	22	4
department	B-tree	5	0
parts	B-tree	5	0
orders	heap	3	0

The above figures are approximate; they are updated only when they change by a certain percentage (5%) to prevent performance degradation by continuously updating these catalogs. Also, if transactions that involve many new pages are backed out during a recovery, then the page counts may not be updated. Page counts are guaranteed to be exact only after modification.

In evaluating overflow, if the number of overflow pages is greater than 10-15% of the number of data pages, expect performance degradation. Overflow must be regularly monitored to ensure that performance does not degrade as rows are appended to tables.

Repetitive Key Overflow

Storage structures other than heap that have a high degree of duplication in the key values are likely to have overflow since duplicate keys are stored in overflow pages. Keys with a high degree of duplication are not recommended. This applies to secondary index keys as well as primary keys.

Repetitive key overflow occurs, for example, if the emp table is keyed on sex, resulting in two primary pages for the values "M" and "F." The remainder of the pages would be overflow pages to these two primary pages.

Consider if the following query is run:

```
select * from student
where student.sex = 'F'
and student.name = 'Baker' ;
```

The key is used to find the first primary page. Then the search goes down the entire overflow chain for “F” looking for all names Baker (there might be more than one Baker, so every page is checked). Since this query looks restrictive, the locking system would probably choose to page level lock. The query would lock 10 pages and then eventually escalate to a table level lock. You may have had to wait for the table level lock if other users were updating. Finally, the search finishes scanning the overflow chain and returns the row.

Retrieval performance with a duplicate key is still better than for a heap table because only half the table is scanned.

However, update performance suffers. If a user wants to append a new female student, the locking system starts by exclusively locking pages in the “F” overflow chain. If another 10 pages need to be locked eventually, the locking system attempts to escalate to an exclusive table level lock. If only one user is updating the table, the lock is easily obtained. If multiple users are trying to update the table at the same time, deadlock is likely.

User1 and User2 would both exclusively hold 10 pages in the table. User1 wants to escalate to an exclusive table level lock so the query can continue, but User1 cannot proceed until User2 drops the exclusive page level locks User2 holds. User2 also wants to obtain an exclusive table level lock, but cannot proceed until User1 releases the locks. This is deadlock, which can seriously degrade update performance.

Poorly Distributed Overflow

Overflow that is not uniformly distributed, that is, it is concentrated around one or two primary pages, is poorly distributed. A classic example of poorly-distributed overflow occurs when new rows are added to a table with a key that is greater than all the keys that already exist in the table (for example, a time stamp). If this table has an isam structure, the table builds up overflow in the last primary page, and all operations involving this overflow chain will exhibit poor performance. This type of table is best stored as a B-tree or hash.

Overflow and Storage Structures

This section discusses overflow issues as they relate to specific storage structures.

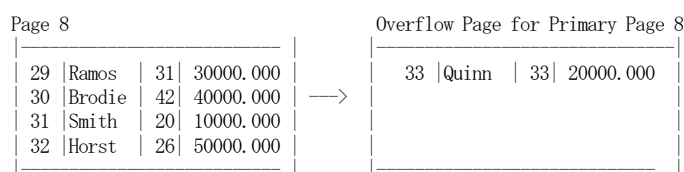
Isam and Hash Tables

In hash and isam tables that have had a large amount of data added and have not been remodified, overflow and the resulting performance degradation is easy to understand. A keyed retrieval that normally would touch one page, now has to look through not only the main data page, but every overflow page associated with the main data page. For every retrieval, the amount of disk I/O increases as the number of overflow pages increases.

Overflow pages are particular to a main data page for isam and hash tables, not to the table itself. If a table has 100 main pages and 100 overflow pages, it is likely that the overflow pages are distributed over many main data pages (that is, each main data page has perhaps one overflow page). A keyed retrieval on such a table would possibly cause only one additional I/O rather than 100 additional I/Os.

For more information on overflow in hash tables, see *Specifying Alternate Fill Factors* earlier in this chapter.

For isam tables, because the isam index is static, if you append a large number of rows, the table may begin to overflow. If there is no room on a page to append a row, an overflow page is attached to the data page. For example, if you wanted to insert empno #33, there is no more room on the data page, so an overflow page is allocated for the data page as shown in the following diagram:



For hash and isam tables, one way of looking at overflow is by looking at the tids of rows and analyzing the way the tids grow in a sequential scan through the table.

As an example, the sample code shown below can be customized to show overflow distribution. Each time a primary page is encountered, the tid's value grows by 512. If a primary page has associated overflow pages, the tid's value jumps by more than 512. So if the embedded SQL/C program shown below in *Sample Code to Show Overflow* is run, the output looks like that shown in *Output from Sample Code*.

Sample Code to Show Overflow

```
page_val = 0;
exec sql select key, tid
      into :key_val, :tid_val
      from tablename
exec sql begin;
```

```

if (tid_val == page_val)
{
    printf("Primary Page %d, tid = %d,", (page_val/512)+1, tid_val);
    printf(" Starting key value = %d0", key_val);
    page_val = page_val + 512;
    old_tid_val = tid_val;
    overflow_page = 0;
}
else
{
    if (tid_val > old_tid_val + 1)
    {
        overflow_page++;
        printf("\n Overflow page %d, tid = %d0", over_page, tid_val);
    }
    old_tid_val = tid_val;
}
exec sql end;

```

Output from Sample Code

```

Primary Page 1, tid = 0, Starting Key Value = 123
Overflow page 1, tid = 2048
Overflow page 2, tid = 2560
Overflow page 3, tid = 3072
Overflow page 4, tid = 3584
Primary Page 2, tid = 512, Starting Key Value = 456
Overflow page 1, tid = 4096
Overflow page 2, tid = 4608
Overflow page 3, tid = 5120
Overflow page 4, tid = 5632

```

B-tree Tables

Eliminating overflow is one of the major benefits of the B-tree storage structure. Overflow in a B-tree occurs only at the leaf level, and then only if you have a significant number of duplicate keys.

For instance, if 30 new employees all joined the company and all had the last name Aitken, the attempt is made to add their records to leaf page 1. In this case, since leaf page 1 can hold only 8 keys (remember that the leaf page can actually hold $2000/(key_size + 6)$), an overflow leaf page would be added to hold all the duplicate values. This is different than splitting the leaf page, because the same index pointer could still point to the same leaf page and be accurate. There would be no additional key/leaf page entry added to the index.

In B-tree tables, you can look at overflow in the leaf level by running a query of the following type, substituting your B-tree table name for *t*, your B-tree keys for the *keycol* values, and the width of the key for *key_width*:

```

select keycol1, keycol2, overflow =
    (count(*)/keys_per_page)-1
from tablename t
group by keycol1, keycol2;

```

Notes:

- This query is not needed for a B-tree index, in which the automatic inclusion of the tidp column in the key prevents overflow.

- For B-tree tables with key compression selected, in the select statement you can substitute an estimate of the average key size for *key_width*.

The results of this query give an approximation of the amount of overflow at the leaf level, per key value. The query works by calculating the number of keys that fit on a page and dividing the total number of particular key incidents—grouped by key—by this value. For instance, if there are 100 occurrences of a particular key and 10 keys would fit on each page, there would be nine overflow pages at the leaf level.

Other tables may incur overflow pages for reasons other than duplicate keys; hence, overflow distribution may involve more than simply running a query.

Secondary Indexes

Overflow should be monitored in secondary indexes, as well as in the primary tables. Even if the base table has a low overflow percentage, the secondary indexes may have badly overflowed. Except when the base table is a heap or B-tree table, the base table generally overflows before the secondary index.

Secondary indexes need to be monitored and modified at interim points—even between base table modifications—to ensure a low percentage of overflow pages. See *Changing the Index Storage Structure in the Modifying Secondary Indexes* section earlier in this chapter for more information.

Backup and Recovery

It is important to back up your database regularly, so that you can recover your data, if necessary. Databases, or individual tables, can be damaged accidentally by hardware failure or human error. For instance, a disk crash, power failure or surge, operating system bugs, or system crashes, can destroy or damage your database or tables in it.

This chapter describes the following backup and recovery features of Ingres:

- Checkpointing and journaling to back up a database or selected tables
- Unloading a database
- Copying a database to back up particular tables or all of the objects you own in a database
- Operating system backups to replace current or destroyed tables in a database
- Roll forward of a database to recover a database or selected tables from checkpoints and journals

Ingres allows you to perform *full recovery*, which involves recovering an entire database, or *partial recovery*, which recovers selected tables in a database. Partial recovery entails recovering data from a backup copy at a level of granularity finer than the entire database. In the event of failure, Ingres will be able to, if possible, mark less than the whole database physically inconsistent. The advantage of partial recovery is that it reduces recovery times by requiring only recovery of logically or physically invalid data.

The Logging System

The Ingres logging system keeps track of all database transactions automatically. It is comprised of the following facilities and processes:

- Logging facility, which includes the transaction log file
- Recovery process (dmfrcp)
- Archiver process (dmfacp)
- Cluster server process (dmfcsp, VAXcluster configurations only)

The Logging Facility

Each installation has an installation-wide transaction log file that keeps track of all transactions for all users. The log file can be distributed among up to sixteen partitions (locations), although Ingres treats the files as one logical file. These log files are identified by the configuration file parameter `II_LOG_FILE_n`, where *n* is the partition number from 1 to 16.

With dual logging enabled, the installation has an alternate log file. With dual logging, a media failure on one of the logs will not result in the loss of data or the interruption of service. If one of the log file disks fail, the logging system automatically switches over to access the other log without interrupting the application. The dual log file is identified by the configuration file parameter `II_DUAL_LOG_n`.

When log files are properly configured, the use of dual logging has a negligible impact on system performance.



In a VAXcluster installation, there is one log file per cluster node. Dual logging is also provided on VAXclusters. ■

Log Space Reservation

During normal online processing, space is reserved in the transaction log file for possible use during recovery when it is rolling back transactions. The reserved space is used to write Compensation Log Records (CLRs), which describe the work performed during the rollback.

Generally, the logging system reserves approximately as much log file space to perform the rollback as was required to log the original operation. Exceptions are insert and update operations, which require less reserved space than the original log.

In the Log File page in the Performance Monitor window, you can see a close approximation of the log file space required for both normal log writes and for CLRs. Also displayed is the number of log file blocks reserved for use by the recovery system at any point in time.

To access the Log File page, you click on the Log Information branch in the Performance Monitor window, and click the Log File tab in the Properties pane.

Note: For information on the Ingres system command equivalents to accomplish these tasks, see the `sysmod` and `set log_trace` commands in the appendix, "Ingres Commands," in this guide.

The `set log_trace` statement is also described in the Tracing with Set Log_Trace section in this chapter.

The Recovery Process

The recovery process (dmfrcp) handles online recovery from server and system failures. The logging system writes consistency points into the transaction log file to ensure that all databases are consistent up to that mark and to allow online recovery to take place when a problem is detected. While a transaction is being rolled back, users may continue working in the database.

The recovery process is a multi-threaded server process, similar to a normal DBMS server. However, the recovery process does not support user connections. The process must remain active whenever the installation is active.

The Archiver Process

The archiver process (dmfacp) removes completed transactions from the transaction log file and, for journaled tables, writes them to the corresponding journal files for the database. Each database has its own journal files, which contain a record of all the changes made to the database since the last checkpoint was taken. The archiver process “sleeps” until sufficient portions of the transaction log file are ready to be archived or until the last user exits from a database.


The Cluster Server Process



The cluster server process (DMFCSP) is resident only on VAXcluster configurations, and is responsible for the management of installations running on different cluster nodes. The process manages the startup, shutdown, and recovery of all cluster nodes. The cluster server has capabilities of both the recovery process and archiver process, as well as special-purpose logging and locking capabilities.

The cluster server maintains a history of important actions in its own message log file, resident in:

`II_SYSTEM:[INGRES.FILES]IICSP.LOG`

Relevant parts of this file should be included in any communication with Technical Support regarding this process. 

Verifying Data Accessibility Before Backup

As the DBA, you should know that the data in your database is good before backing it up. This will assure that a successful recovery can be made should it become necessary to restore the database from the backup copy.

One method of verifying the accessibility of your tables would be to write a script that automatically checks each of the tables and system catalogs in your database. Otherwise, you can use one of the following suggested methods with Visual DBA:

- Modify system tables to predetermined storage structures using the System Modification dialog.
- Modify user table storage structures using the Modify Table Structure dialog.
- Use any procedure that will touch all the rows in each table being backed up. (For example, select all the rows from the tables using an SQL Test window.)

If rows in a table are not accessible, you will receive an error message. If this happens, restore the table from an earlier checkpoint before doing a new backup.

- Check the integrity of specific tables using the Verify Database dialog. (For each table, specify report for Mode, table for Operation, and a table name.)

The detailed steps for performing these procedures can be found in the Procedures section of the online help for Visual DBA. See the following topics:

- Optimizing System Tables
- Modifying Storage Structures
- Specifying a Query
- Verifying a Database

Note: For information on the Ingres system command equivalents to accomplish these tasks, see the `sysmod`, `modify`, `select`, and `verifydb` commands in the appendix, "Ingres Commands," in this guide.

Backing Up a Database with Checkpoints

Using checkpoints, you can make a static (“snapshot”) backup of your entire database, or selected tables. For a dynamic backup of your database, you use checkpointing in combination with journaling. These backup methods enable you to restore data up to the last checkpoint, or the last journaled transaction, respectively.

This section primarily discusses the use of checkpointing for static backups. For details on dynamic backups, see the Using Journaling System section for more information.

To checkpoint a database or tables, you must be a privileged user (operator privilege or Ingres system administrator).

Note: For information on the Ingres system command equivalent to accomplish this task, see the ckpdb command in the appendix, Ingres Commands," in this guide.

Checkpointing a Database

You can create a new checkpoint for a database by using the Checkpoint dialog, invoked by the Database Checkpoint menu command in Visual DBA. Each time you perform this operation, a new checkpoint of the database is taken.

A record of up to 99 checkpoints can be maintained at any point. It is recommend that at least one database-level checkpoint be included in this record.

The use of the Database Infodb menu command to verify the status of the database and checkpoints is encouraged. This will help ensure that a valid database checkpoint is always available.

The detailed steps for performing this procedure can be found in the Procedures section of the online help for Visual DBA. See the Setting Checkpoints topic.

Running a checkpoint does not affect the current state of journaling for the database. For details on how to enable and disable journaling with a checkpoint, see the Starting Journaling and Stopping Journaling sections.

Tables that have had journaling enabled since the previous checkpoint will have their journaling status changed from “enabled after next checkpoint” to just “enabled.”

Checkpointing Tables

In the Checkpoint dialog, you can specify that checkpointing be done for selected tables. Table-level checkpoints should only be used as a supplement to database-level checkpoints, and never as a substitute for them.

Database versus Table-Level Checkpoints

Use caution in the area of table-level checkpoints and recovery. Generally, full database checkpoints are recommended over table-level checkpoints. When using table-level checkpoints and restores, it is important—at the very least—to back up all dependent tables with a full checkpoint.

Table-level checkpoints are restricted in their recoverability when the checkpointed table has been dropped or the table has been modified through any DDL statement. In these cases, the table-level checkpoint is rendered unusable. There is also danger in compromising the referential integrity of the database when rolling forward a table without journaling.

Performing table-level checkpoints on system catalogs is not permitted. Frequent database checkpointing of the iidbdb database is strongly encouraged.

Roll Forward of Tables

Whenever a database is rolled forward, it is recommended that a new checkpoint be taken to allow subsequent table-level roll forward activities.

When a roll forward is performed at the table level, you may choose either to roll forward the table excluding or including all secondary indexes. You cannot specify a secondary index name as a table.

If it is necessary to do a roll forward with the No Secondary Index option, the base table's secondary index in the RDF cache will become inconsistent. To clear the inconsistency, do one of the following:

- Drop or recreate the inconsistent secondary index
- Restart Ingres to refresh the RDF cache

Call Computer Associates Technical Support if additional assistance is required.

Checkpoint Template File

A file called the checkpoint template file, `cktmpl.def`, drives the checkpoint and roll forward operations. The `cktmpl.def` file allows you to customize backup and recovery processes and provides additional information tracking. It is possible to modify the backup process so that the names of the tables that are specified during a table-level backup are written to a text file.

The `IL_CKTMPL_FILE` environment variable overrides the default `cktmpl.def` file for a particular user. This should be used when testing modifications to the `cktmpl.def` file before it is made available to the entire installation so that other users in the installation are not affected.

See the Altering the Checkpoint Template File section for checkpoint template codes and parameters.

Online and Offline Checkpoints

Taking Online Checkpoints

An *online* checkpoint is one taken when users are using the database. This is the default when you take a checkpoint.

Taking Offline Checkpoints

A checkpoint taken *offline* is performed when no one is using the database. To take an offline checkpoint, enable the Exclusive Lock check box in the Checkpoint dialog.

When you specify the Exclusive Lock option, you can also specify the Wait option to wait for the database to be free before performing the checkpoint.

The behavior with or without the Wait option specified is described as follows:

- If specified, the wait will be as long as necessary for the database to become free before taking the checkpoint.
- If not specified, an error message is returned if the database is busy.

Locking During a Checkpoint

By default, an exclusive lock is not taken on the database when you take a checkpoint. Other users who may be using the database at the time of the checkpoint can continue working online. During this time, transactions in progress are placed in the dump file for the database.

When you perform a roll forward, the dump files are used to restore the database to its state when the checkpoint was taken. It then updates the database from journals, if the database is journaled.

There are two cases, however, in which checkpointing takes an exclusive database lock. These are if either of the following options in the Checkpoint dialog are used:

- The Exclusive Lock option is specified to take the checkpoint offline
- The Enable Journaling or Disable Journaling options are specified to enable/disable journaling


If you want to continue the present journaling status, you do not use either journaling option.

Cleaning Up Outdated Checkpoints


To delete previous checkpoints and journals after you take a new checkpoint, specify the Delete Previous option in the Checkpoint dialog.

Up to 98 checkpoints can be deleted in this way. If you have taken more than 98 checkpoints since the last time you created a checkpoint with the Delete Previous option, you need to delete the additional old checkpoints manually using an operating system command:

Windows

Use the Windows `del` command from the `II_CHECKPOINT\ingres\ckp\dbname` directory. 

UNIX

Use the UNIX `rm` command from the `ii_checkpoint/ingres/ckp/dbname` directory, where `ii_checkpoint` is the value of `II_CHECKPOINT` as displayed by the `ingrenv` command. 

VMS

Use the VMS `delete` command. 

Observe the following cautions when manually deleting checkpoints:

- Do this only **after** creating a checkpoint with the Delete Previous option.
- Be sure that you do not delete the most recent checkpoint. You can identify the most recent checkpoint by its version number, described below.

When you checkpoint a database, a checkpoint file is created for each location on which the database is stored. The names of the checkpoint files are in the format shown by the following example:

`C000v001.ckp`

where *v* shows the version number of the checkpoint sequence and *l* shows the location number of the data directories. The most recent checkpoint file has the highest version number. To obtain this number, select the database and choose the Infodb command from the Database menu. View the information in the Infodb dialog.

Deleting the Oldest Checkpoint

You can delete the oldest available full database checkpoint, along with associated journal and dump files, by enabling the Delete Oldest Checkpoint check box in the Database Characteristics dialog, invoked by the Operations Alter DB menu command.

This option operates only for full database checkpoints, not partial checkpoints. For additional information about the Database Characteristics dialog, see the Altering Database Characteristics section in this chapter.

Checkpoints and Destroyed Databases

Important! A checkpoint is a backup of an existing database. If you destroy the database (with the Edit Drop menu command), you will not be able to recreate it from a checkpoint, because this deletes a database's associated checkpoints as well.

If you want to destroy your database and then recreate it, use the Unloaddb menu command, appearing off the Database Generate Scripts submenu.

Parallel Checkpointing in UNIX

UNIX

In UNIX, you can checkpoint to a disk or a tape in parallel, as described in this section.

Checkpointing to Disk

To checkpoint a multi-location database to disk in parallel, issue the ckpdb command with the #m flag followed by the number of parallel checkpoints to be run. For example, to save two data locations at a time to the II_CHECKPOINT location, the command would be as follows:

```
ckpdb #m2 dbname
```

Checkpoint to Tape

To checkpoint a multi-location database to tape in parallel, in the Checkpoint dialog, specify multiple table devices to be used in the Tape Device edit control. For example, enter the following:

```
/dev/rmt/0m,/dev/rmt/1m
```

This saves one location per tape—the first location will be stored on device 0m; the second on device 1M. The third location will be stored on whichever device is finished first. The remaining locations will be stored on the next free device. The operator is prompted to insert a new tape for each location.

Some points to be aware of when performing parallel checkpointing to tape in UNIX include:

- Recovery does not have to be in parallel if a checkpoint was done in parallel.

- Each tape label should include the checkpoint number, database name, and location number.
- Each tape device must be the same medium, that is, all 4mm or all 8mm; mixing is not permitted.
- The maximum number of devices that can be used is limited by the system's input and output bandwidth.

Putting Checkpoints on Tape in Windows

Windows

In Windows, the backup system uses the Windows backup utility to create checkpoints on tape. This utility allows you to back up on multiple tapes. The program prompts you for more tapes as needed during the checkpoint procedure.

The backup uses the commands in the following batch file:

```
%II_SYSTEM%\ingres\bin\ckcopyt.bat
```

You can tailor these commands in any way you wish (for example, to meet local conventions such as tape labeling).

For detailed information on backing up to tape, please see your Windows documentation on backup utilities. 

Putting Checkpoints on Tape in UNIX

UNIX

In UNIX, the backup system uses an operating system utility, such as tar (Berkeley UNIX) or cpio (System V), to create checkpoints. Both cpio and tar are limited to handling files that will fit on a single tape. Since checkpoints of larger databases will abort at the end of the first tape, you must estimate both the checkpoint size and the tape capacity before checkpointing these databases. If you estimate that the checkpoint will exceed the tape size, follow instructions in the Checkpointing to Multiple Tapes in UNIX section.

The following sections provide instructions for estimating checkpoint and tape size, checkpointing to a single tape, and checkpointing to multiple tapes.

Estimating Checkpoint File Size in UNIX

A separate checkpoint file is created for each location to which a database has been extended. To estimate the size of checkpoint files:

1. Issue the following command at the operating system prompt:

```
du ii_database/ingres/data/default/dbname
```

where *ii_database* is the value of the environment variable `II_DATABASE` displayed by the `ingprenv` command.

For other locations, substitute the name of the directory associated with the location name.

2. If your operating system uses tar, increase the resulting block size of the directory by 5%.
3. The du command displays the directory size in blocks. To get the file size in bytes, multiply the block size by the number of bytes in a block on your operating system.

See your operating system manual for information on the number of bytes in a block on your system.

Estimating Tape Capacity in UNIX

The capacity of a tape depends on the following:

- Density at which the tape is written
- Length of the tape
- Size of the blocks written on the tape
- Length of the inter-record gap (IRG)

Standard 9-track tape drives write at either 800, 1600, or 6250 bits per inch (bpi), so the bits per inch specification is the same as bytes per inch. The standard tape length is 2400 feet.

Block sizes, which are not standardized, are important because of what is between the blocks—the IRG. A typical IRG is .75 inches of empty tape separating each block from the next. With this information, you can use the following formula to estimate the size of the file in bytes that a tape can accommodate:

$$F = (B + (I * D)) / (12 * B * D * L)$$

where:

- F is the file size in bytes
- B is the block size in bytes
- D is the density in bits per inch
- L is the length of the tape in feet
- I is the IRG in inches

The sample file sizes in the following table were calculated for a standard 2400 foot tape, assuming an IRG of .75:

Tape Size	IRG	Block Size	Density	File Size (MB)
2400	.75	512	1600	13.8
2400	.75	512	6250	17.7
2400	.75	8192	1600	40.2
2400	.75	8192	6250	114.5

After using this formula to calculate the file size, you need to add an arbitrary amount to allow for miscalculations. You do not want a tape to run off the reel because you miscalculated the size of the file that should fit. A reasonable amount to add is 5% of a tape's capacity.

If your system uses a cartridge tape or other storage media, contact the vendor for the specifications that will allow you to make the calculations described above.

Checkpointing to a Single Tape in UNIX

To checkpoint a database to a single tape:

1. Mount a tape reel.
2. In the Checkpoint dialog, enter the name of the tape drive in the Tape Device edit control.

The equivalent ckpdb command at the operating system prompt would be as follows with a tape drive named "/dev/rmt8":

```
ckpdb -m/dev/rmt8 dbname
```

The backup created by this checkpoint writes over everything that was previously on the tape.

Checkpointing to Multiple Tapes in UNIX

There are two cases to consider when checkpoint files exceed the tape size:

Case 1

In Case 1, the checkpoint file exceeds the size of the tape, but will fit on a disk. In this case, follow this procedure:

1. Follow normal procedures for checkpointing to disk.
2. Have your operating system administrator move the checkpoints from disk to tape. Use a standard system backup method, such as `cpio` or `dump`.

If some of the database's tables are stored in alternate locations, separate checkpoint files are created for them in the checkpoint location. These files may be small enough to move to single tapes.

Caution! *To System V Users: It is possible that large checkpoints will exceed the `ulimit` on your system. (The `ulimit` is a tunable operating system parameter that sets a limit on file size.)*

Case 2

In Case 2, the checkpoint file exceeds the size of the tape and will **not** fit on a disk.

In this case, you must checkpoint the database using the operating system. To successfully checkpoint a database, you have to lock all users out during the entire process.

To lock out all users and take the checkpoint, follow this procedure:

1. To synchronize journaling, checkpoint the database to a null device by specifying the following options in the Checkpoint dialog:
 - Exclusive Lock
 - Wait
 - Delete Previous
 - Tape Device: `/dev/null`

The Wait option causes the checkpointing to wait until all user locks have been released before beginning the checkpoint.

The Delete Previous option removes all previous checkpoints and journals.

The Tape Device specification causes the checkpoint to be placed in `/dev/null`, which is a nonexistent device. This makes the database “think” it is being checkpointed and causes journaling to be correctly synchronized. At this time, all changes to the database are guaranteed to be on disk.

2. To lock the database, start a new process:
 - C shell:
After the first message from the checkpoint is printed, press `Ctrl+Z`.
 - Bourne shell:
Log in at another terminal immediately after the checkpoint begins.
 - Start the new process by issuing the following command at the operating system prompt:

```
ingres -l +w dbname
```

The `+w` flag causes a wait until that lock is granted.

3. After the checkpoint finishes:

For the C shell:

If the checkpoint process is stopped (csh job control), put the job back in the foreground; wait for the process to complete.

For the Bourne shell:

Wait for the process to complete.

4. Have your operating system administrator use standard system backup methods to back up the database directory to tape.

Make sure that the backup method used allows you to save the files and recover them to their original places on the system. Some backup methods have limitations. The volcopy command, for instance, requires that the database disk device be unmounted and unavailable for use by any users during the copy. Additionally, it saves files by saving the entire file system.

5. For the C shell:

Leave the second process stopped (csh).

For the Bourne shell:

Leave the second process at the SQL prompt (*) until the backup is complete.

6. Quit from the SQL prompt held by the second process. ■

Putting Checkpoints on Tape in VMS

VMS

To initiate a checkpoint in VMS, ready the tape and issue the ckpdb command with the -m option. See the appendix, "Ingres Commands", in this guide, for information about the ckpdb command.

The backup system uses the VMS BACKUP utility to create checkpoints. This utility allows you to back up on multiple tapes. The program will ask for more tapes as needed during the checkpoint procedure.

The backup uses the following command in the script:

```
IL_SYSTEM:[INGRES.FILES.CHECKPOINT]CKP_TO_TAPE.COM
```

You can tailor this command in any way you wish, for example, to meet local conventions, such as tape labeling. ■

Using the Journaling System

For a dynamic backup of your database, journals are used in combination with checkpoints. Checkpoints provide you with a snapshot of the database at the time you took the checkpoint. Journals keep track of all changes made to journaled tables since the last checkpoint.

When you are journaling a database:

- Take regular checkpoints of your database to minimize recovery time.
- Periodically verify that your journaling data is correct by auditing the database. See the Producing Audit Trails with Journals section.

The detailed steps for performing the procedures in this section can be found in the Procedures section of the online help for Visual DBA. See the following topics:

- Setting Checkpoints
- Altering Database Characteristics

Note: For information on the Ingres system command equivalents to accomplish these tasks, see the ckpdb and alterdb commands in the appendix, "Ingres Commands," in this guide.

Starting Journaling

Journaling may be selected for an entire database or on a table-by-table basis, as described below:

- Journaling on the entire database

The recommended approach is to journal the entire database. Tables in journaled databases are created "with journaling" if that is the default_journaling setting of the server class used by the DBMS server you are connected to.

You can then disable journaling on specific tables only if a rollforward recovery of those tables is not important. You should exercise caution when creating non-journaled tables in journaled databases. Non-journaled tables cannot be audited when the database is audited, in addition to their lack of roll forward recovery. Following a roll forward recovery, the relationship between journaled and non-journaled tables can be very confusing.

- Journaling table-by-table

If you choose to journal selected tables, you are responsible for ensuring that all related objects are also journaled (for example, that all tables associated with a view are journaled).

Journaling an Entire Database

To journal an entire database, use one of the following methods:

Enable journaling on the database using the Enable Journaling option in the Checkpoint dialog.

Note: The only tables that will be enabled are those whose journaling status is “enabled after next checkpoint.” Tables whose journaling status is “disabled” will not be enabled.

Journaling New Tables

The journaling of new tables begins:

- If you have enabled journaling on the database and the table is created with journaling enabled, the new tables begin journaling immediately.
- If you have not enabled journaling on the database, the new tables begin journaling after you take a checkpoint with the Enable Journaling option in the Checkpoint dialog (although tables created with journaling disabled are never enabled even after journaling is enabled for the database as a whole).

Starting Journaling and Online/Offline Checkpoints

To start journaling on a database that has not yet been checkpointed, invoke the Checkpoint dialog (by choosing the Database Checkpoint menu command) and set the Enable Journaling option.

Please note the following:

- The first time journaling is turned on in a particular database, you must checkpoint the database by setting the Enable Journaling option in the Checkpoint dialog. This ensures that the checkpoint is taken offline and with an exclusive lock on the database.
- Once you have enabled journaling by checkpointing offline with the Enable Journaling option, you can maintain the “journaling on” status and take online checkpoints by not subsequently setting the Enable Journaling option when you take a checkpoint. Online checkpoints permit users to continue using the database while the checkpoint is being taken.
- Once you have enabled journaling for the database by checkpointing offline with the Enable Journaling option, you can take an offline checkpoint to start journaling of tables for which journaling is enabled after the next checkpoint.

Any explicit journaling option causes the checkpoint to be taken offline, exclusively locking the database.

Disabling Journaling

To disable journaling, disable the Journaling check box in the Options dialog invoked from the Create Table dialog.

Stopping Journaling

To stop journaling a particular table, issue the following statement from the query language monitor:

```
set nojournaling on tablename;
```

You can stop journaling all the tables in a database using either of the following methods in Visual DBA:

- Creating a checkpoint, specifying the Disable Journaling option in the Checkpoint dialog.
- Altering a database using the Database Characteristics dialog. Note that this will take effect immediately, therefore, it should only be used for emergencies. See the next section, Disabling Journaling When Altering a Database.

To re-enable journaling on a table or database that has had journaling disabled, use the Checkpoint dialog, as described in the Starting Journaling section.

Disabling Journaling When Altering a Database

When you alter a database with the Disable Journaling option (using the Database Characteristics dialog), journaling of a database is halted immediately, regardless of whether users are connected to the database.

This option is provided as a method for recovering from journaling system problems that prevent the archiver from moving transaction log file records to the database journal files. This might be the case, for example, if the disk partition containing the journal files is not periodically purged of obsolete journal files and the partition becomes entirely full. If the logging system is unable to move records from the log file to the journal files, the transaction log file will eventually fill up, causing a LOGFULL condition. When this occurs, no database activity can proceed until the LOGFULL state is cleared.

Important! Using this option to disable journaling will make the displayed value for the journaling status inconsistent. Tables will show as “journaling enabled,” even though journaling is disabled for the database as a whole and you would expect to see “enabled after next checkpoint.”

See the Altering Database Characteristics topic in the Procedures section of the Visual DBA online help for step-by-step instructions.

The following example shows the steps that must be performed by the DBA of the database. It does not require a database lock and can be run even while the log file is full (LOGFULL).

1. To disable journaling on a database, select the database in the Database Object Manager and choose the Alter DB command from the Operations menu.

The Database Characteristics dialog appears.

2. Enable the Disable Journaling check box and click OK.

At this point the database is no longer journaled.

Caution! Do not roll forward a database that has journaling disabled. Any transactions committed after the alter database operation, or that were still in the transaction log file at the time journaling was disabled, will be lost.

3. To check the database state, choose the InfoDb command from the Database menu.

Using the Infodb dialog, you can determine whether journaling has been disabled.

4. Schedule a new checkpoint to re-enable journaling as soon as it is possible. To do this, select the database in the DOM window and choose the Checkpoint command from the Database menu.

The Checkpoint dialog appears.

5. Enable the Enable Journaling check box and click OK.

Disabling Journaling When Checkpointing

When you choose the Disable Journaling option in the Checkpoint dialog, journaling is stopped for all tables in a database.

This causes a checkpoint of the specified database to be taken and then journaling on it stopped. After stopping journaling, you can still take periodic checkpoints of the database.

Altering Database Characteristics

As previously discussed, you can disable journaling from the Database Characteristics dialog. This dialog also allows you to change several database characteristics, including:

- Change journal block settings
- Delete oldest checkpoint
- Set verbose mode

In order to perform this operation, you must be the owner of the database or have the operator privilege.

Resizing the Journal File

Journal files are created by the archiver process by the first journal write after a checkpoint takes place. Additional journal files are created as prior files are filled.

By default, journal files are created with:

- A target number of journal blocks of 512
- A block size of 16,384 bytes
- An initial allocation of 4 blocks

This results in a target journal file size of 8 MB (16,384 * 512 bytes). Although most users will find these parameters satisfactory, all three may be modified via the Database Characteristics dialog, using the Block, Size, and Initial edit controls.

Setting the Target Journal Size

The Database Characteristics dialog specifies the target journal size in the Block edit control. The possible values are between 32 and 65536.

A journal file is closed and a new one is created when either a checkpoint is taken (actually, when the first write after a checkpoint is taken) or when the journal file fills.

The Block edit control allows some control over when the logging system declares a journal file full. This parameter is known as the “target journal file size” because the exact size of a journal file cannot be easily predicted. The archiver closes off journal files, if they grow larger than the target number of blocks, only at the completion of an archive cycle. Longer archive cycles imply more variation in journal file sizes.

Upon successful completion of this command, a message is written to the `errlog.log`. The updated block value can be observed as the `infodb` parameter “Target journal size”.

The command takes effect immediately (or more accurately, the next time the archiver reads the configuration file).



This option has no effect on journal files created as part a “VAXcluster merge.”

The initial journal size (specified in the Initial edit control) may be affected by this command.

Setting the Journal Block Size

The Database Characteristics dialog allows you to specify the journal block size in the Size edit control. Valid journal block sizes are 4096, 8192, 16384, 32768, and 65536 bytes.

Archiver (`dmfacp`) performance is affected by the journal file block size. You would normally change the block size (Size edit control) in conjunction with the number of target journal blocks (Block edit control). This allows you to target the creation of journal files of a given size. Changing the block size without also changing the number of blocks in a journal file will change the target size of the file.

You would typically change the journal block size immediately after the database is created, before the initial checkpoint is taken with the journaling option. Thereafter, changing the journal block size is generally required only for installations with a relatively high volume of journaled data. You can only change the journal block size when journaling is not currently enabled.

To change the journal block size on a database that is currently journaled, perform the following operations:

- Take a checkpoint and disable journaling using the Checkpoint dialog
- Set the journal block size using the Database Characteristics dialog
- Take a checkpoint and enable journaling using the Checkpoint dialog

Upon successful completion of this operation, a message is written to the `errlog.log`. The updated journal file block size can be observed as the `infodb` “Journal block size” parameter.

Setting the Initial Journal Size

The Database Characteristics dialog allows you to specify the initial journal size in the Initial edit control. Valid journal block sizes are from 0 to the current target journal size (which can be obtained using `infodb`).

The Initial option allows a measure of control over when journal file disk space allocation takes place, but only for the first journal file created after a checkpoint is taken.

You can change the initial journal size at any time, and it takes effect when the next database journal file is created. In the case of an offline checkpoint, this may be some time after a checkpoint is taken. In the case of an online checkpoint, the file allocation occurs during execution of the checkpoint.

Upon successful completion of this command, a message is written to the `errlog.log`. The updated block value can be obtained from the "Initial journal size" parameter in the `Infodb` dialog (invoked by the Database `Infodb` menu command).

Resize Considerations

Preallocating space in journal files using the Database Characteristics dialog can reduce the likelihood of running out of journal file disk space. Filling a journal file causes the archiver to stop, and if left untreated will eventually cause the log file to fill, which will bring the system to a halt.

With the `alter database` operation you can, for example, request creation of journal files of a given size and also request preallocation of the entire file. If the file is sufficiently large, this eliminates the possibility of running out of journal disk space during normal online processing.

This may, however, cause unused journal space to be wasted. If excessive space is allocated during journal file creation, that disk space will be made unavailable when a subsequent checkpoint operation takes place.

UNIX

On UNIX systems, disk space must be physically written when a journal file is extended. Since consecutive journal files are created as prior ones fill, it is undesirable to affect performance caused by file initialization occurring at unplanned intervals during the processing day. The `alter database space preallocation` features can be used to manage when the allocation takes place, allowing control over when the allocation time delay occurs. A significant amount of journal file I/O may occur when the first journal file is created, with the archiver being unavailable during this time. This will be observed as an online checkpoint taking a long time to complete, or the archiver performing a large amount of work when the first journal write after an offline checkpoint takes place. ■

If it is necessary to more accurately control journal file size, the archiver must be awakened more frequently. This can be accomplished with smaller consistency point (CP) intervals, allowing more frequent archiver “wake-ups.” The consistency point interval can be configured using the Ingres Configuration Manager. (For details on the Ingres Configuration Manager utility, see the Using the Ingres Configuration Manager topic in the Visual DBA online help.) Smaller CP intervals can affect system performance, although the processing involved is for a short interval of time.

Producing Audit Trails with Journals

In addition to using journals for recovery, you can use journals to produce audit trails of changes to a database. You must be the DBA for the database or have the security privilege to perform an audit on a database.

You should audit your database periodically to verify that your journals are correct.

The detailed steps for performing this procedure can be found in the Procedures section of the online help for Visual DBA. See the Auditing a Database topic.

Note: For information on the Ingres system command equivalent to accomplish this task, see the `auditdb` command in the appendix, “Ingres Commands,” in this guide.

Auditing a Database

The audit database operation in Visual DBA can be performed on a database by using the Audit Database dialog, invoked by the Operations Audit menu command.

The Audit Database dialog enables you to produce a listing or file of changes made to journaled tables since the last checkpoint. This listing may not include all changes made since the last checkpoint for the following reasons:

- Since the audit database does not exclusively lock the database, other users may complete a transaction while the audit is running.
- If other users are using the database when you perform an audit, a completed transaction may not have been moved to the journal files.

The audit database operation scans journal files twice. A prescan is performed to filter out undesired information (for example, aborted transaction data). The second scan outputs journal records of interest. To improve program performance, the Before edit control value terminates both scans when an End Transaction record is found that has a time later than that specified.

When the Inconsistent check box is enabled, you are allowed to view journals that the database has marked as inconsistent. Note, however, that audit database operation will still fail if core catalogs are inconsistent.

When the Wait check box is enabled, the audit waits until journals are current. "Current" in this context means either of the following:

- No further archiving is required on the database.
- The archiver has copied all log file information up to the log file end-of-file when the audit database request was initiated.

Note that if a large amount of unarchived information remains in the log file when this request is initiated, there may be significant delay in processing.

Loading an Audit Trail as a Table

To make querying the data easier, you can create an audit trail as a file in your current directory and then load the file into a table in your database. To do this:

1. In the Audit Database dialog, enable the Output Files check box and specify the corresponding files to create an audit trail file in the current directory. (Note that you must have first specified at least one table. Also, you can specify file(s) only if the table you are auditing has fewer than 1940 bytes per row.)

For example, the audit database operation could extract a record of the changes to an employee table from the journal for a particular database. The changes in the current directory could then be placed in a file named empaudit.trl.

2. To copy the file into a database table, create a table to hold the audit trail data.

When creating the table, include the audit trail and employee table columns shown below. Enter the audit trail columns before the table's columns, in the order shown. If you do not, the copy operation will fail when you try to copy the audit trail data into the table.

Column Name	Data Type	Description
date	date not null with default	Date and time of the beginning of the multi-query transaction that contained the operation
username	char(32) not null with default	User name of the user who performed the operation
operation	char(8) not null with default	Insert, update, or delete operation
trandid1	integer not null with default	Transaction identification number. Concatenated with trandid2.
trandid2	integer not null with default	Transaction identification number. Concatenated with trandid1.
table_id1	integer not null with default	Table identification number. Corresponds to value in table_reltid column of iitables system catalog for specified table.
table_id2	integer not null with default	Table identification number. Corresponds to value in table_reltidx column of iitables system catalog for specified table.
name	varchar(20)	Employee name

Column Name	Data Type	Description
age	integer	Employee age
salary	money	Employee salary
dname	varchar(10)	Department name
manager	varchar(20)	Employee manager

3. Use the copy statement to load the new table with the data from the file from Step 2.

In the following example, the data in the empaudit.trl file is copied to the empaudit table:

copy empaudit() from 'C:\users\joe\empaudit.trl';

copy empaudit() from '/usr/joe/empaudit.trl';

copy empaudit() from '[usr.joe]empaudit.trl';

The table created from the audit trail (in this example, the empaudit table) will contain:

- A row for each row added to the employee table
- A row for each row removed
- Two rows for each update: one showing the row before the update and the other showing the row after the update

Note: See the appendices in this guide for more information on the copy statement.

Windows

UNIX

VMS

Copying a Database

You can copy a database to back up the tables, views, and procedures that you own in a database. By default, all of the tables, views, and procedures that you own in the database are copied. If you specify table names, only those tables will be copied.

Since any user authorized to use a database can use the copy database operation, this is a useful backup method for a non-DBA, who can use it to back up his or her own tables, views, and procedures.

Creating Copy Scripts

To create scripts that will be used to copy objects from a source database to a target database, you use the Generate copy.in and copy.out dialog, invoked from the Database Generate Scripts Copydb menu command.

The detailed steps for performing this procedure can be found in the Procedures section of the online help for Visual DBA. See the Creating Copy Database Scripts topic.

Note: For information on the Ingres system command equivalent to accomplish this task, see the copydb command in the appendix, "Ingres Commands," in this guide.

Copying a Database

To perform the copy database operation, you run the scripts that have been created using the Generate copy.in and copy.out dialog.

The detailed steps for performing this procedure can be found in the Procedures section of the online help for Visual DBA. See the Executing Copy Database Scripts topic.

Unloading a Database

Unloading a database is a time-consuming method for backing up and recovering your database, because all of your database's files must be unloaded and then reloaded. For this reason, it is recommended that you use checkpointing instead.

However, unloading a database can be useful as a backup tool because it enables you to:

- Generate copy scripts, which can be used to recreate your database.
- Recover particular tables by editing the copy.in scripts.

The detailed steps for generating these scripts can be found in the Procedures section of the online help for Visual DBA. See the Creating Unload and Reload Scripts topic.

Note: For information on the Ingres system command equivalents to accomplish this task, see the `unloaddb` command in the appendix, "Ingres Commands," in this guide.

Recovering Databases

This section tells you how to recover:

- A non-journaled database from a checkpoint
- A journaled database from checkpoints and journals
- A database from a tape checkpoint
- Selected tables

Recovering Databases from Checkpoints and Journals

To recover a database from checkpoints and journals or from checkpoints only, you use the *roll forward* operation. Performing a roll forward of a database overwrites the current contents of the database being recovered.

When you roll forward a database, the database is locked to prevent errors from occurring. If the database is busy, the roll forward operation waits for the database to be free before recovering it. (If you specify the Wait option, the rollforwarddb operation pauses until all users have left the database. If you do not specify the Wait option, you get a message that the database is in use.)

If the target checkpoint was taken online (when the database was in use), the roll forward operation:

- Restores the database from the checkpoint location to the database location.
- Applies the log records in the dump location to the database, which restores it to the state when the checkpoint began. The log records contain the transactions that were in progress when the checkpoint was taken.

Since there were no transactions in progress during an offline checkpoint, this step is not performed when restoring a database from an offline checkpoint.

- Applies the journal records to the database, if the database is journaled.

Performing a Roll Forward of a Database

You can recover a database by using the Roll Forward DB dialog, invoked by the Database Rollforward DB menu command in Visual DBA. To perform a roll forward, you must be the DBA for the database or have the operator privilege.

A roll forward may write Compensation Log Records (CLRs) to the transaction log file while executing the rollback phase of a roll forward recovery. This happens rarely, only if incomplete transaction histories are written to the journals. This is an unlikely condition except when the transaction log file is lost (or, if running with dual logging, when both copies are lost). In this case, it is possible for journal files to grow in size as a consequence of performing a roll forward.

The detailed steps for performing this procedure can be found in the Procedures section of the online help for Visual DBA. See the Recovering a Database from Checkpoints topic.

Note: For information on the Ingres system command equivalent to accomplish this task, see the rollforwarddb command in the appendix, "Ingres Commands," in this guide.

Recovering a Journalled Database

To recover a specific database from the last checkpoint and journal, where both the checkpoints and journals are stored online, select the default options in the Roll Forward DB dialog.

Recovering a Non-Journalled Database

To recover a non-journalled database from the last checkpoint, enable the From Last Checkpoint check box in the Roll Forward DB dialog.

Recovering a Database from Tape Checkpoints

To recover a database whose checkpoints are on tape, mount the tape reel containing the checkpoints. Then, select the default options in the Roll Forward DB dialog and also specify the tape drive in the From Tape Device edit control.

The checkpoint is read from the tape and then the journal files are applied, if the database is journalled, to bring your database up to date.

Parallel Roll Forward from Disk

UNIX

To roll forward a multi-location database to disk in parallel, issue the `rollforwarddb` command the `#m` flag followed by the number of parallel restores to be run. For example, to restore two data locations at a time from the `IL_CHECKPOINT` location, the command would be as follows:

```
rollforwarddb #m2 dbname 
```

Parallel Roll Forward from Tape


UNIX

To roll forward a multi-location database from tape in parallel, specify the devices to be used in the From Tape Device edit control. For example, the following tape device may be specified:

```
/dev/rmt/0m,/dev/rmt/1m
```

This restores one location per tape – the first location will be restored from device 0m; the second location will be restored from device 1M. The third location will be restored from whichever device is finished first. The remaining locations will be restored from the next free device. The operator is prompted to insert the numbered tape into the free device.

Some points to be aware of when performing parallel roll forward from tape in UNIX include:

- Recovery does not have to be in parallel if a checkpoint was done in parallel.
- Recovery can be in parallel if a checkpoint was not done in parallel.
- Each tape label should include the checkpoint number, database name, and location number.
- Each tape device must be the same medium, that is, all 4mm or all 8mm; mixing is not permitted.
- The maximum number of devices that can be used is limited by the system's input and output bandwidth. 

Recovering Tables Using Roll Forward

You can specify that only certain tables are recovered during a roll forward database operation. (Journals of tables in the database must be enabled.) When doing table-level recovery, you may optionally move the table to a new location. (Note that the database must be extended to the new locations before the rollforward.)

In the Roll Forward DB dialog, some of the available options that relate to specifying tables include:

- Specifying individual tables
- Relocating tables
- Continue processing on error conditions
- Inhibiting automatic recovery of secondary indexes

Note that table recovery is not allowed if structural changes have been made to the table since the checkpoint. This means table recovery is not allowed if you have modified the table, created indexes, or altered the number of columns in the table.

Retracting Changes Using Roll Forward

If a user makes a serious error in a table that is being journaled, the changes can be retracted. Use roll forward to restore the database up to the beginning of the transaction in which the error occurred.

For example, to restore a database from the previous checkpoint to its condition at 8:00 A.M. on August 17, 1998, you would specify the following options in the Roll Forward DB dialog:

- Verbose check box enabled
- From Last Checkpoint check box enabled
- From Journal check box enabled
- Before: 17-aug-1998:08:00:00

This retracts all changes made to the database after this time, not just those made to the table with the error.

To ensure that the error is not reintroduced when you perform a roll forward in the future, take a new checkpoint to reset the journals.

Recovering a Subset of Data Using Roll Forward

The roll forward end time option (specified in the Before edit control) permits the recovery of a subset of data in the journal file. The option is useful when problems have been encountered in a full roll forward database operation or when, for example, a critical piece of data has been inadvertently deleted.

There is one important consideration when using this option. As this form of recovery does not restore the database to the state reflected by the full set of journals, it is critical that a checkpoint of the database be performed after the recovery completes. If not, another roll forward performed later may leave the database in an inconsistent state.

The only recommended courses of action that may be taken is to roll forward a database with the Before value specified:

- Roll forward the database again
- Checkpoint the database, preferably specifying the Delete Previous option (to delete previous checkpoints)

Note also that the Before and End options operate on End Transaction timestamps, not on the time that a user may associate with an update. The audit database End and Before options (in the Audit Database dialog) also operate on End Transaction timestamps, and can be used to check anticipated roll forward results.

Recovering a Database from an Old Checkpoint

If the most recent checkpoint has been damaged or is unreadable, it is possible to recover from an older checkpoint. You can use either a specific checkpoint number or the most recent usable checkpoint.

To recover the database from a particular checkpoint and apply all journals since that time, enable the From Specified Checkpoint check box and enter the checkpoint number in the corresponding spin box.

The checkpoint sequence number must be a valid checkpoint number. You can verify this number in the Infodb dialog, invoked by the Database Infodb menu command.

If the most recent checkpoint is unfinished and you want to recover using the most recent usable finished checkpoint, enable the From Specified Checkpoint check box and do not enter a corresponding checkpoint number.

The From Specified Checkpoint option can also be used in conjunction with the After and Before edit controls if you want to restore a database to its state at some previous moment in time. You should exercise extreme caution with the After and Before options. As these commands roll the database forward to a point in time other than that fully represented by the journals, transactions that were performed after the Before time or before the After time are lost. Partially completed transactions will be backed out by the roll forward process.

Furthermore, a checkpoint should always be performed after completion of such a roll forward, thereby ensuring that obsolete journal data is not inadvertently reused in a subsequent recovery (or by an audit database operation to produce inaccurate auditing results). Note that audit database After and Before options behave as do the equivalent roll forward flags, and can be used to predict roll forward results.

For a full explanation of the options associated with roll forward operation, see the Roll Forward DB dialog topic in the Visual DBA online help. It can be accessed through the Recovering a Database From Checkpoints topic in the Procedures section.

Recovering from the Loss of the Transaction Log File

In the unlikely event of a loss of the transaction log file (or, if dual logging is enabled, loss of both file copies), the following recovery procedure may be used to restore as much database information as is possible.

1. Create a new transaction log file.
2. At this point the action differs, depending on whether offline or online backups take place. Included in the latter class of systems are those that employ journaling capabilities.

- For offline backups

Installations using their own backup and recovery mechanisms (implying no use of online checkpoint or journaling facilities) only need to restore database directories and bring the system back up. No directed recovery is needed, since backups are done during a period when there is no system activity, and when all database information is resident on disk.

- For online backups and roll forward

If you are using online checkpoints and journaled databases, bring the installation back up with the newly initialized log file. All databases open at the time of the failure will be marked inconsistent by the recovery process. Each must be recovered in turn by the roll forward database operation. The Enable Journaling option with roll forward is specified for journaled databases; this option is not specified for those databases that are not journaled.

Note that a roll forward operation restores databases to a consistent state even if incomplete transaction histories have been copied to the journal files.

Altering the Checkpoint Template File

The checkpoint template file drives the checkpoint and roll forward operations. This section describes this command file in detail so that, if needed, you can tailor it to meet the requirements of your site.

For example, if the database exists on multiple locations, checkpointing backs up each location to a separate tape or disk and, in turn, roll forward restores each location one at a time. If you want to use a different backup method or only one tape for all locations, you can edit this command file.

Checkpoint Template Codes

In the checkpoint template file, a four-character uppercase code at the beginning of each line provides the following information:

First Character	<p>The first character indicates when the command is to be used. Valid characters are:</p> <ul style="list-style-type: none">■ B (Begin)—the command is to be executed before the device is used. It indicates setup work done prior to the execution of the command.■ P (Pework)—the command is to be executed before the work is executed.■ I—the command begins table-level recovery (initializes only).■ W (Work)—the command activates the device. It indicates the execution of the command.■ F—the command ends table-level recovery (comments only).■ E (End)—the command is executed after the device is used. It indicates cleanup work done after the operation is complete.
Second Character	<p>The second character indicates whether the command specifies several types of checkpointing and roll forward options. Valid characters are:</p> <ul style="list-style-type: none">■ S—the command is for checkpointing only.■ R—the command is for roll forward only.■ E—the command is for both checkpointing and roll forward.■ D—the command is for delete file processing.■ C—the command checks if a database checkpoint exists before the roll forward.■ J—journals are to be applied, for a roll forward.■ U—dumps are to be applied, for a roll forward.
Third Character	<p>The third character specifies the device. Valid characters are:</p> <ul style="list-style-type: none">■ T—the command on that line refers to reading from or writing to a tape.■ D—the command refers to disk operations.■ E—the command applies to both types of devices.
Fourth Character	<p>The fourth character specifies the data. Valid characters are:</p> <ul style="list-style-type: none">■ D—the command is for a database.■ A—the command is for all databases.■ T—the command is for table(s).■ E—the command is for either a database or table.
Examples	<p>Some examples of a checkpoint template code follow:</p>

WSTD identifies the command line to use during the working (W) phase of a checkpoint, which is saving (S) a database to tape (T), for a database (D).

BRDT identifies the command line to use during the begin (B) phase of a roll forward operation that is restoring (R) from disk (D) for a table (T).

Substitution Parameters

The checkpoint template file can optionally include substitution parameters that will be filled in at run time, to specify things like:

- Which database directory to back up
- Which tape device the user specified in the Checkpoint dialog

The parameters consist of a “%” and a single uppercase character. These parameters have the following meanings:

%T	The type of operation: 0 if to tape, 1 if to disk.
%N	The total number of locations being written.
%M	For the begin or end operations, the incremental/current location number. For save or restore operations, this starts at 1 and is incremented after each save or restore command.
%D	The path to the database directory being saved or restored.
%C	The path to the checkpoint directory of disk files or the device name if to tape.
%F	The name of the checkpoint file created or read from.
%A	%C prepended to %F in a form to produce a fully specified file (that is, %A = %C/%F).
%X	The name of the table, pertinent to the work commands executed under table processing.
%B	Expanded during execution to represent the list of internal files that are associated with a table checkpoint. This parameter is pertinent to the work commands executed under table processing.

The “%” parameters in the commands are replaced by ckpdb and/or rollforwarddb when the command is executed.

Valid Code Combinations

The valid code combinations are shown here:

B	[S,R,E,J,U] [T,D,E] [T,D,E,A]
P	[S,R] [T,D] [D,T]
W	[S,R,E,J,U,D,C] [T,D,E] [T,D,E,A]
I	[,R,E] [T,D,E] [T,E]
F	[,R,E] [T,D,E] [T,E]
E	[S,E] [T,D] [D,T,E]

For every entry with a first character of B, there must be an accompanying entry beginning with E.

This section demonstrates how the codes are used in the checkpoint template file to perform checkpointing and roll forward operations in a variety of ways.

Checkpointing

The checkpointing operation (ckpdb command) executes the following sequence of codes in the cktml.def file:

Bsxy	Beginning checkpoint
Wsxy	Executed once for each location
Esxy	Ending checkpoint

where:

x denotes D for disk, T for tape, or E for both.
y denotes D for database, T for table, or E for both.

Roll Forward

The roll forward operation (rollforwarddb command) processes the following codes in the cktml.file:

WCxA for each location

If table processing is specified, the following codes are executed:

BRxT	once per location
IRxT	once per location
WDxT	for each table
WRxT	for each table
FRxT	once per location
EExE	once (note that ERxT is executed if available)

If an entire database is being recovered (rather than specific tables), the following codes are executed:

BRxD once for each location
 WDxD once for each location
 WRxD once for each location
 EExE once (note that ERxD is executed if available)

For all roll forward operations, the following codes are executed:

BUxA if dumps are to be applied
 WUxA
 EExE
 BJxA if journals are to be applied
 WJxA
 EExE

Format of the Checkpoint Template File in Windows

Windows

The checkpoint template file uses the two batch files, ckcopyd.bat (for checkpointing to disk) and ckcopyt.bat (for checkpointing to tape).

The checkpoint template file, cktmpl.def, can be found in the following folder:

%II_SYSTEM%\ingres\files

Each line contains a command preceded by a four-character code that tells when to use the command.

By altering this file, or the two batch files that it calls, you can change how checkpoints are performed. You can add or delete flags to the underlying operating system commands, or you can supply your own batch files to perform the backup and restore steps.

Windows Examples

For example, the command:

BSTD: echo Beginning checkpoint to tape %C of %N locations

indicates what is done initially, before the device is used (B), when checkpointing is used to save (S) a database location to tape (T), for a database (D).

As another example, when executing a checkpoint on a database which spans multiple locations, one of the following commands is executed once for each location (WSTD for backup to tape, WSDD for backup to disk):

WSTD: ckcopyt %N %D BACKUP

WSDD: ckcopyd %D %A BACKUP

The commands instruct the checkpoint operation to call either the ckcopyt.bat or ckcopyd.bat batch command file to do the actual backup.

The checkpoint utility automatically substitutes the appropriate values for "%N," "%D," and "%A."

The ckcopyt.bat batch file calls the Windows backup backup command and passes it the name of the directory for the location, and other operating system flags. ■

Format of the Checkpoint Template File in UNIX

UNIX

The checkpoint template file, cktmpl.def, uses the UNIX tar command. This file can be found in:

\$II_SYSTEM/ingres/files

Each line is a command preceded by a four-character code that instructs the checkpoint operation when to use the command.

By altering this file, you can change how checkpoints are performed. You can add or delete flags from the tar commands or you can supply your own shell scripts to perform the backup and restore steps.

UNIX Examples

For example, the command:

BSTD: echo beginning checkpoint to tape %C of
 %N locations

indicates what is done initially, before the device is used (B), when the checkpoint operation is used to save (S) a database location to tape (T) for a database (D).

As another example, when executing a checkpoint on a database that spans multiple locations, the following command is executed once for each location:

PSTD: echo mount tape %N and press return;
 read foo;

WSTD: cd %D; /bin/tar cbf 20 %C *

The command instructs the checkpoint operation to save each location on a tape and to use the tar command with the parameter cbf 20. The checkpoint utility automatically substitutes the appropriate value for "%N," "%D," and "%C." ■

Format of the Checkpoint Template File in VMS

VMS

The checkpoint template file, cktmpl.def, can be found in:

\$II_SYSTEM:[INGRES.FILES]

The checkpoint template file uses the four-letter key described above to begin each line. A line can specify an individual tape and disk handling command or the name of a user-written command file to provide more complex processing such as backing up all of a database's locations concurrently.

Defaults are provided so that sites using standard processing do not need to alter the checkpoint template file.

VMS Example

Here are some example lines from a cktmpl.def file:

```
WSTD: @ckp_to_tape "%N" "%D" "%C" "%F"
WSDD: @ckp_to_disk "%D" "%A"
WRTD: @rollfwd_from_tape "%N" "%D" "%C" "%F"
WRDD: @rollfwd_from_disk "%A" "%D"
```

Each of these example lines specifies the name of a command file and establishes requests for run-time information. ■

An Alternate Checkpoint Template File in UNIX

UNIX

The alternate checkpoint template file, cktmpl_cpio.def, uses the UNIX cpio command to backup and restore the database files. This file can be found in:

\$II_SYSTEM/ingres/files

To use this template file, enter the following command to override the default cktmpl.def template file:

```
ingsetenv II_CKTMPL_FILE $II_SYSTEM/ingres/cktmpl_cpio.def ■
```

Backup and Recovery of the Master Database (iiddb)

The iiddb database is your Ingres installation's master database. It contains information about your installation as a whole, such as:

- Which databases exist in this installation
- Where user databases are located
- Which locations may be used for files
- Which users may access databases

The iiddb also contains information about groups, roles, and database privileges defined for your site.

Checkpointing the
iiddb

The iiddb is journaled by default.

It is important to regularly checkpoint and journal the iiddb database just as you would do for a user database. Ckpdb and rollforwarddb are the supported utilities for recovering the iiddb if it is lost or damaged for any reason. The system catalogs containing the installation information for groups, roles and database privileges are stored in the iiddb database and can only be recovered from backups.

The detailed steps for performing these procedures can be found in the Procedures section of the online help for Visual DBA. See the following topics:

- Setting Checkpoints
- Recovering a Database From Checkpoints

Tracing with Set Log_Trace

You can use the log_trace option of the set statement to start and stop tracing of logfile writes. Using this option requires the trace privilege.

Caution! You should use set log_trace only as a debugging or tracing tool. Set log_trace output is not guaranteed to remain the same across versions, so you should not base applications on it. You are not guaranteed of the support of set log_trace in this or future versions.

To start tracing log writes, issue the following statement:

```
set log_trace;
```

To stop tracing log writes, issue the following statement:

```
set nolog_trace;
```

When you use set log_trace during a session, you receive a list of the log records written during execution of your query, along with other information about the log. Set log_trace output includes:

- The length of the log and the amount of space reserved for its CLR. For more information on CLRs, see the Log Space Reservation section.
- If the log write is a normal log record (do/redo) or a CLR.
- If the log record will be copied to the journal file.
- If the log is associated with a special recovery action.

Improving Database and Query Performance

This chapter contains information and tips on how to improve and optimize query and database performance. Good performance requires planning and regular maintenance by the DBA.

By following the techniques and procedures in this chapter, you may find that you can often solve the problem yourself. If it is still necessary to call your Technical Support representative, you should be able to accurately define the problem, in this way contributing substantially to a rapid solution.

Note: This chapter assumes that Ingres is running satisfactorily.

Locking and Concurrency Issues

Use this section if your performance problem occurs in a multi-user environment or if the query runs slowly or hangs intermittently.

Concurrency problems occur when several users access the same tables, and at least one is a writer. If your query needs to access objects that are locked, that session will wait indefinitely for locks to be released unless the lockmode timeout is set, or a deadlock occurs.

The sections below suggest some reasons for concurrency problems.

Identifying Lock Waits

Use the Lock Information branch of the Performance Monitor window within Visual DBA to monitor lock waits. For details on using this window, see the online help topic, *Viewing Performance Information*. If you find lock waits, identify the queries that are holding locks on the resources you are waiting to access. You may need to modify your locking strategy to avoid future problems.

Pay particular attention to:

- maxlocks
- readlock = nolog
- timeout
- set lock_trace command

If the lock being waited on was created as the result of lock escalation, your system may be configured with too few system-wide locks. This is a configuration issue.

If lock escalation occurs because too many locks are taken on a given table's pages, a set lockmode statement can be issued to increase this threshold. The default is 10 before escalation occurs.

Multi-Query Transactions (MQTs)

Remember that a transaction accumulates locks on resources until you roll back or commit. A transaction that is waiting for lock(s), or that is not waiting for a lock but nevertheless seems unusually slow, may be using excessive server or system resources.

Here are suggestions:

- Keep your transactions as short as possible.
- Commit your transactions quickly:
 - You will create large MQTs unless you use set autocommit on or commit after each statement. Statements accumulate as one multi-query transaction until you commit.
 - MQTs should not include prompts that hang the transaction until a user responds; or sleeps that prevent your transaction from being released quickly.
- Avoid bottlenecks in your transaction such as:
 - Insert to heap table with secondary indexes
 - Counter table updates
 - Iterative deletes
 - Unbounded long iterations

Managing Overflow

Overflow chains slow concurrent performance. Overflow pages are attached to the main data page if a record must be added to a full main page. Then a query that should touch one main data page must now touch that page plus each associated overflow page. This will increase I/O, cause concurrency problems, and use up locking system resources.

Here are suggestions:

- Monitor overflow chains.

Check the number of overflow pages for your tables and secondary indexes. To monitor overflow in Visual DBA, select a table or secondary index in the Database Object Manager window, and click the Pages tab. Use the legend to interpret the information displayed.

If the number of overflow pages is greater than 10-15% of the number of data pages, expect performance degradation.

- Overflow problems are often caused by duplicate keys.

- Some table structures create long overflow chains when much new data is added. If you have problems with overflow, you may want to consider trying a different storage structure:

- Heap

Heap tables are created as one main page with an overflow chain. There is no overflow management.

- Hash

Overflow pages occur in a newly modified table if the key is repetitive; this is normal but undesirable. Check a freshly modified table. If there is overflow, consider using isam instead.

- Isam

Has a fixed index that can cause long overflow chains. Modify frequently or use B-tree for a non-static table. Use heap structure for large bulk updates, then modify back to isam to avoid update performance problems.

- B-tree

No overflow if there are no duplicate keys, so consider making keys unique. Overflow occurs only at the leaf level. Use the Shrink B-tree Index option to reorganize it. Use heap structure for bulk loads, then modify to B-tree.

- Decrease overflow

There are several ways to decrease overflow and improve concurrency:

- Use unique keys.

- Modify the table to reorganize it; with a B-tree structure, simply specify the Shrink B-tree Index option.

- Consider tailoring the table's fill factor.

Set Statements

There are a variety of set statements you can use to manage your locking strategy. Be sure you are using user-defined lockmodes and isolation levels to their fullest for concurrency and deadlock avoidance. See your query reference guide sections for command syntax. Pay particular attention to:

- Deadlock
- Lock_trace flag
- Maxlocks
- Readlock = nolock
- Timeout

Database Maintenance

If your query used to run quickly and recently it slowed down, or the speed of the query changes depending on the constants specified in the where clause, your problem may be poor database maintenance. To optimize performance, set up maintenance procedures to include running DBA utilities.

The following features are especially useful in tracking performance problems:

- Optimization—perform using the Optimize Database dialog within Visual DBA or the `optimizedb` command

This feature collects the statistics that are used by the Ingres query optimizer to determine the best query execution plan (QEP) to use for your queries.

 - Periodically run optimization on all your databases to generate statistics for columns that are keys or indexed. List the other columns you need as an argument to this command.
 - Run full optimization statistics on columns referenced in where clauses of strategic queries that are having problems.
 - For very large tables, create statistics based on sample data.
 - When there are significant changes to your data distribution, run optimization on the affected columns.
 - Do not collect excessive statistics, because you will build up large optimizer tables with unused data.
 - Always run system modification after optimization.
- Modification—perform using the Modify Table Structure and Modify Index Structure dialogs within Visual DBA or the `modify` statement

You can modify a table or index to:

- Reorganize data on new data pages
- Free deleted record space
- Reduce overflow chains
- Adjust the fill factor

Use the Shrink B-tree Index option (or modify to merge statement) to:

- Reorganize index pages of B-tree tables
- Reduce overflow chains

Use the Change Location option (or modify to relocate statement) to move your tables to balance disk access.

- **System Modification**—perform using the System Modification dialog within Visual DBA or the sysmod command

This feature modifies system catalogs to predetermined storage structures.

- Run system modification on the iistatistics system catalog after optimization.
- Run system modification on ii_rcommands if you create and update a lot of Report-Writer reports.
- Regularly using system modification reduces overflow in your system catalogs. Run it often if catalog changes are frequent due to development or if you use many create or drop statements in your applications.

- **Verification**—perform using the Verify Database dialog within Visual DBA or the verifydb command

Use this utility to:

- Destroy or list unrequired disk files, expired tables, or temporary tables in your database
- Clean up fragmented disk space

See the appendix, "Ingres Commands," in this guide, for more information on the commands mentioned in this section. See the Visual DBA online help for more information on the dialogs mentioned in this section. For discussions of maintenance issues, see the chapters "Maintaining Databases" and "Maintaining Storage Structures," in this guide.

Design Issues

Good query performance requires planning. Poor performance can be caused by design problems if the following are not considered; these subjects are covered in sections below:

- Storage structures and index design
- Key design
- Query design

Other important design issues are:

- Database design
- Validation checks and integrities
- Grants and views
- Application design

Diagnostic Hierarchy

The diagnostic hierarchy begins with the area of greatest gain. A thorough performance analysis must include each item in the following list. Notice that the highest priority items are the design issues covered in the sections below. For instance, if your database design is flawed, perfect server configuration will not help you avoid query performance problems.

Storage Structures and Index Design

Choosing the correct table storage structure for your needs can improve concurrency and query performance. Remember that there is no substitute for testing and benchmarking your queries.

For modifying and compressed storage structures and a discussion of overflow, see the chapter, “Maintaining Storage Structures,” in this guide.

Keys

Key design can be a very complex subject. The following material presents guidelines only.

Evaluate Your Keys

Here are guidelines on good keys and bad keys:

Good keys have the following features. They:

- Use columns referenced in the where clauses and joins of your queries
- Are unique or as unique as possible

Document reasons for maintaining non-unique keys.

All keyed storage structures can enforce unique keys. They are:

- Short
- Static
- Non-nullable

Bad keys have the following features. They are:

- Wide
 - Use wide keys with caution.
 - You get fewer rows per page.
 - Evaluating the hash function takes more time with wide keys.
 - A wide key deepens the index level of B-tree and isam logarithmically, with respect to key width. B-tree is the least affected table structure.
 - Consider using a surrogate key as an alternative.
- Non-static

Updating the index will slow performance.
- Non-uniform duplication

A mix of high and low duplication will cause inconsistent query performance.
- Sequential keys
 - These should be used with care.
 - Isam tables will be lopsided, and the overflow chains will cause concurrency problems.
 - Control sequential key problems with a frequent modify schedule.

Using Multi-Column Keys

Multi-column keys have special issues. If used improperly in your query, the key will not be used and the search will do a full-table scan.

Keep the following in mind:

- Use the most unique and frequently used columns for the left member of a multi-column key.
- Searches on B-tree and isam tables must use at least the leftmost part of a multi-column key in a query, or a full-table scan will result.
- Searches on hash tables must use an exact match for the entire key in the query, or a full-table scan will result.
- Optimizer statistics are approximated by adding the statistics of the columns making up a multi-column key.

Surrogate Keys

When you use a short surrogate or internal key to replace a bad key, or because there is no good key, consider the performance trade-offs. The set processing of data includes the overhead of deriving the key. There are three surrogate key types:

- Natural
Universal, a social security number, or zip code are examples.
- Environmental
These are local to an organization, like an employee number.
- Design artificial. These are:
 - Local to an application
 - Hard to remember
 - Hard for users to understand
 - Can be hidden from users

Query Design

Query design can be a very complex subject. The following material presents guidelines only.

The following tips will improve the performance of your queries.

- Conversion joins are joins where two columns of different data types are joined in a query, either explicitly or implicitly. These joins are frequently the result of database design problems and should be avoided.
- Avoid using function joins.
 - Functions in the where clause force a full-table scan.
 - Control uppercase and lowercase, and so on, at input time.
- Some complex OR queries may be rewritten as unions.
- Evaluate QEPs for critical queries:
 - Can large table scans be avoided?
 - Is an additional index needed?
 - Are Cartesian products with large tables used?
 - Are function joins used?
- Use repeated queries for queries that are used many times.
- Do not forget to commit.
Consider using set autocommit on.

Before Calling Technical Support

If you have worked through the query performance evaluation and your problem is still not resolved, then it is time to call Technical Support. Before you call Technical Support there are two procedures to follow:

- Isolate and analyze the suspect query
- Create a test case

Isolate and Analyze the Suspect Query

Use the following procedure to determine if you have a problem with the user interface, the query itself, or a software bug.

1. Isolate a poorly performing query from your user interface using the trace flag set `printqry`, which prints queries before they are optimized and executed. Identify the query that seems to hang.

Execute the query in the Terminal Monitor or from within the Visual DBA SQL Test window, and determine if performance is the same. If performance is only a problem when the query is executed from the user interface, you have identified an application problem. If performance is the same, continue.

2. In the Terminal Monitor, issue the following statements to display the QEP without running the query:

```
set qep;  
set optimizeonly;
```

Now, execute your query and save the output to a file for examination. After running the query, exit the Terminal Monitor session or turn query execution back on using:

```
set nooptimizeonly;
```

3. Review the Design Issues section and evaluate the QEP for your query. For example, you can look for:
 - Large table scans that can be avoided
 - An additional index that is needed
 - Cartesian products with large tables
 - Function joins

If you are not able to identify your problem and suspect a software bug, submit your query to Technical Support along with a test case.

Create a Test Case

First verify that you are on the most recent version of Ingres available for your platform. Next, collect the information Technical Support needs to duplicate your problem, and then call.

Technical Support needs the following information in ASCII files that you can send by email, UUCP, or on a tape:

- The exact query that causes the error to occur
- The QEP generated by the problem query
- Dump optimizer statistics for all the tables in the query (use the Direct Output to Server File option in the Display Statistics dialog within Visual DBA, or the `statdump` command with the `-o` flag)

- The help table *tablename* information for all the tables that the query references (or equivalent information obtained from within Visual DBA)
- The help index *indexname* information for all secondary indexes of tables in the query (or equivalent information obtained from within Visual DBA)
- The help permit on table *tablename* information for grants on all the tables in the query (or equivalent information obtained from within Visual DBA)
- A query of the system catalogs for information about each table. Look at *iirelation* and select *relpages*, *reltups*, *relmain*, and *relprim*, where the *relid* is equal to each table and index in the query.
- The create scripts and data for all the tables, indexes, and grants that the query references. When generating the scripts, you should specify the Create Printable Data Files option.

Now you are ready to contact Technical Support about your problem.

Ingres Commands

This appendix is intended for programmers and users of Ingres who have a basic understanding of how relational database systems work. In addition, the reader should have a basic understanding of the operating system.

Issues you should be aware of before using this guide include the following:

- C2 Security—Ingres installations can be administered in compliance with the C2 security standard.
- Enterprise Access Compatibility—If you are working through an Ingres/Enterprise Access product (formerly called “Gateways”), refer to your Ingres/Enterprise Access documentation for information about syntax that may differ from that described in this guide.

Audience

This guide will be useful to the person who has responsibility for the operation of Ingres and needs a quick reference to Ingres commands and system utilities used when monitoring and troubleshooting Ingres.

Special Considerations

UNIX Variations

In this guide, command formats for the following UNIX shell variations are shown:

- C shell
- Bourne shell

For the Korn shell, use the Bourne shell syntax. Refer to your operating system shell documentation for any variations required on your particular system.

Command formats for the following UNIX operating system variants are shown where needed:

- BSD
- System V

Query Languages

The industry standard query language, *SQL*, is used as the standard query language throughout this guide. Ingres is compliant with ISO Entry SQL92. For details about the settings required to operate in compliance with ISO Entry SQL92, refer to the online *SQL Reference Guide*.

This chapter lists Ingres commands that are executed at the operating system level. Complete syntax and descriptions of the commands used for normal operation are listed. Many of these commands operate on the database as a whole. Other commands invoke various Ingres querying and reporting tools and preprocessors.

Command Syntax

You enter system-level commands to invoke an Ingres command or utility at the operating system prompt. A command consists of one or more required command words, usually followed by one or more parameters or flags, including database name(s):

command *dbname* | *vnode::dbname*[/*server_class*] [*flags*] [*parameters*]

In general, you can enter command options in any order. However, a few commands require certain ordered parameters.

Standard Command Line Flags and Parameters



A *flag* is a letter preceded by a hyphen (-) that determines various options for commands. Use the following standards:

- *Parameter* – A flag may stand alone (-f) or be followed by a parameter (-f*parameter*). Generally, there is no intervening space entered between a flag and parameter.
- *Case-sensitivity* – Flags are shown in lowercase unless they are required to be passed as uppercase. Uppercase flags may need special input syntax if the host operating system is case-insensitive. See the Uppercase Flags section.

A parameter is any other command line option that is not a flag. It can be the name of a database, a table or other object, or a value that specifies a particular use for a command.

The following table describes the parameters and flags that are commonly used in many commands. You can specify these options if they are shown in the syntax sections of this chapter.

Flag	Description
<i>dbname</i>	The name of a database. This parameter must precede all other non-flag parameters (with the exception of <i>vnode::dbname</i>).
<i>vnode</i>	The name of the computer on which your database is stored, as known to Net. If you are accessing a database on a remote node, you must specify <i>vnode</i> , the name of the remote host where the database resides. It must be followed by two colons (::) and the <i>dbname</i> parameter, with no intervening space.
<i>server_class</i>	The name of one of the Ingres servers or Ingres/Enterprise Access products (for example, <i>db2</i>). If you are accessing a distributed database or a non-Ingres database through an Enterprise Access product, you must specify the <i>server_class</i> . For a complete description of server types, see your Ingres/Enterprise Access documentation.
[-f] <i>product</i>	<p>The name of a product parameter. In selected commands, the catalog modules for one or more products may be specified. The user interface catalogs are grouped into modules. Each Ingres tool requires a set of modules in order to operate. If you omit the product, the command reads the installation's authorization string and specifies all products that the authorization string permits.</p> <p>The product parameter must be one of the following:</p> <p><i>ingres</i>—processes catalogs for the Ingres tools (ABF, QBF, RBF, and VIFRED).</p> <p><i>ingres/dbd</i>—processes catalogs for DBD.</p> <p><i>vision</i>—processes catalogs for Vision.</p> <p><i>windows_4gl</i>—processes catalogs for OpenROAD.</p> <p><i>nofeclients</i>—directs the command not to process catalogs for any user interface products. You cannot use the <i>nofeclients</i> name in conjunction with the name of any valid user interface product; <i>nofeclients</i> is valid only in specified commands.</p>
-u <i>username</i>	<p>Specifies the effective user name for the session. Valid only for a privileged user, DBA, or sessions that have the <i>db_admin</i> database privilege. (Some commands, including <i>ckpdb</i>, <i>rollforwarddb</i>, <i>verifydb</i>, <i>createdb</i>, and <i>destroydb</i>, restrict the use of the <i>-u</i> flag to privileged users.)</p> <p>Note: The <i>-u</i> flag does not assume the group of the effective user. Use the <i>-G</i> flag to distinguish between the real and effective user.</p>

Flag	Description
-Ggroupid	<p>Specifies the group identifier for the session. After the Ingres system administrator defines a group identifier, a DBA can grant database permissions to the group. When you issue a command, specifying group ID (using the -G flag), the group's permissions are applied to the session.</p> <p>In order to specify a group, you must be a member of the specified group identifier's user list, an Ingres system administrator, the DBA of the specified database, or a user that has the db_admin privilege.</p> <p>If you omit this flag and there is a default group identifier specified for you, the default group identifier is assigned to the session. (Default group identifiers are assigned using accessdb.)</p>
	You must enclose this parameter in double quotation marks ("-Ggroupid").
-Rroleid	<p>Specifies the role identifier for an application image. After the Ingres system administrator defines a role identifier, a DBA can grant database permissions to the role ID. When you invoke an application and specify role ID (using the -R flag), the role permissions are applied to your session.</p> <p>The <i>roleid</i> must be an existing role identifier. If the role identifier requires a password, you are prompted for the password. If you specify the -R flag, but omit both the role identifier and password, you are prompted for both. If no password is defined for the specified <i>roleid</i>, press the Enter key when prompted for the password.</p> <p>Neither <i>roleid</i> nor password is validated if you are an Ingres system administrator, DBA for the specified database, or a user that has the db_admin privilege.</p>
	You must enclose this parameter in double quotation marks ("-Rroleid").

Uppercase Flags

Flags that must be entered in uppercase may need special input syntax when the host operating system is case-insensitive.

Windows

The Windows operating system passes uppercase flags with no special formatting needed. For example, to invoke Ingres Interactive Terminal Monitor with a group of sales, you could enter:

isql dbname -Gsales

UNIX

UNIX is case-sensitive and passes uppercase flags with no special formatting needed. For example, to invoke Ingres Menu with a group of sales, you could enter:

ingmenu dbname -Gsales 

VMS

OpenVMS is case-insensitive and requires the addition of double-quotation marks around the uppercase flags. In OpenVMS, be sure to enclose all uppercase Ingres flags in double quotation marks.

For example, to invoke Ingres Menu with a group of sales, use double quotes around the -G designation:

ingmenu dbname "-Gsales" 

Using Schemas for Owner Qualification

A *schema* is a collection of database objects, such as tables. Each table, view, and synonym belongs to a schema that is determined when the object is created. (A *synonym* is a “redefinable” label for a table name. The schema name corresponds to the user who owns the object. The schema name allows you to distinguish between objects with identical names but different owners.

You can specify a schema name for a table, view, or synonym on the command line. You use the following syntax to specify ownership:

schema.objectname

For example, to specify the table named “empinfo” having a schema name of dave, you would specify the table name as:

dave.empinfo

The period (.) must immediately follow the schema name and precede the object name, with no intervening spaces. Both the schema name and the object name can be delimited identifiers.

You do not use a schema name when referencing a table, view, or synonym, for example, you specify the table name as:

empinfo

The search looks first for an object with a schema corresponding to the current user; then it looks for an object owned by the DBA to which you have access. Lastly, if the object name begins with ii, the search looks for a system catalog with that name.

Delimited Identifiers on the Command Line

Delimited identifiers are database object names that are identical to reserved words, words that contain spaces, and non-alphanumeric characters that are disallowed in a regular identifier. If the installation allows mixed case names, you can also use delimited identifiers to distinguish among identical names with different case (for example, SALES and Sales).

On the command line, you use delimited identifiers if needed for names of tables, views, synonyms, schema, and authorization names (users, groups, and roles). For more information on allowable characters in delimited identifiers, see the *SQL Reference Guide*.

To create a delimited identifier, you must enclose the name in double quotation marks ("), dereference any embedded quotes, and use the appropriate number and type of delimiting quotes to pass it through your operating system. Use delimited identifiers on the operating system command line to specify database object names:

report my_database "Jane's table"

You must observe any operating system requirements for specifying quoted parameters, parameters containing embedded quotes, and parameters containing other characters that could be interpreted differently by the operating system. Depending upon your operating system, you add delimiting and dereferencing quotes to a delimited identifier on the command line in order to pass it through the operating system with its own delimiting and embedded quotes (if any).

The names of the tables used in the following examples are shown in this table:

Table Stored in Database	Delimited Identifier
Jane's table	"Jane's table"
"Expert" Table	""""Expert"" Table"



In the Windows environment, surround delimited identifiers and their delimiting quotes with double quotes on the command line, and dereference the delimited identifier quotes, preceding them with a backslash(\):

report my_database "\\Jane's table\\"
report my_database "\\\"Expert\" table\\"



In UNIX, surround delimited identifiers and their delimiting quotes with double quotes on the command line, and dereference the delimited identifier quotes, preceding them with a backslash (\):

report my_database "\\Jane's table\\"
report my_database "\\\"Expert\" table\\"

VMS

In OpenVMS, you must surround delimited identifiers with a set of dereferenced double quotes on the command line. Also, you must dereference each embedded quote by doubling it (including any quotes required to dereference an embedded quote):

```
report my_database ""Jane's table""
report my_database ""Expert"" table""
```

Authorization
Parameters

You can use delimited identifiers to specify a *username* for the **-u** flag, a *groupid* parameter for the **-G** flag, or a *roleid* for the **-R** flag on the command line. A general example is:

```
sreport my_database myfile -u"user 5" -G"group 2"
```

Specific operating system examples are:

Windows

```
sreport my_database myfile -u"user 5" -G"group 2"
```

UNIX

```
sreport my_database myfile -u"user 5" -G"group 2"
```

VMS

In OpenVMS you must also enclose the entire **-G***groupid* parameter within double quotes:

```
sreport my_database myfile-u""user 5""-G""group2""
```

The following is an Windows NT-specific example:

```
sreport my_database myfile -u' " user5' " " -G' " group 2' "
```

Case Sensitivity

By default, identifiers are forced to lowercase, and are therefore case-insensitive. The casing rules can be specified at installation time for delimited identifiers. The following settings are allowed:

- Ingres setting: lowercase (case-insensitive; forces all letters to lowercase).
- ISO Entry SQL92 standard: mixed case (case-sensitive; preserves case for delimited identifiers); regular identifiers are uppercase (case-insensitive; forces all letters to uppercase).

For compliancy with ISO Entry SQL92 standards, the Ingres system administrator should set delimited identifiers to mixed case.

abf

Invokes Ingres/ Applications-By-Forms (ABF).



Syntax

```
abf dbname [vnode::dbname[/server_class] applname [-w] [+wopen] [-5.0]  
[-username] [-Ggroupid]
```

Description

The **abf** command invokes ABF, a forms-based interface for creating forms applications.

The following table lists valid flags and parameters for this command:

Parameter	Description
<i>dbname</i>	The name of the database. The <i>vnode</i> and <i>server_class</i> are specified if required; for details, see the Standard Command Line Flags and Parameters section.
<i>applname</i>	The name of an ABF application. If omitted, ABF prompts for the name of the application.
-w	Causes an application's procedure names to be checked for conflicts with system function names.
+wopen	Generates warnings if ABF detects statements that are not compatible with OpenSQL.
-5.0	Causes 4GL to be invoked in 5.0 compatibility mode.
-username	Specifies the effective user for the session. Files that are created by ABF when using this flag are not owned by <i>username</i> , but rather by the user actually running the ABF process. See the Standard Command Line Flags and Parameters section.
-Ggroupid	Specifies a group identifier. For details, see the Standard Command Line Flags and Parameters section.
	You must enclose this parameter in double quotation marks (" -Ggroupid "). 

accessdb

Authorizes access to a database.

Syntax

```
accessdb [-uusername] [-vnode=vnode]
```

Description

The accessdb command invokes a forms-based interface by which the Ingres system administrator or another privileged user can authorize access to Ingres and individual databases. Accessdb is also used to extend databases to new locations.

The following table lists valid flags and parameters for this command:

Parameter	Description
-uusername	Specifies the effective user for the session. Refer also to the Standard Command Line Flags and Parameters section.
-vnode=vnode	Specifies a vnode name as described in the Standard Command Line Flags and Parameters section.

alterdb

Sets database characteristics.

Syntax



```
alterdb dbname[/server_class]  
  [-target_jnl_blocks=n |  
  -jnl_block_size=n | -next_jnl_file |  
  -init_jnl_blocks=n | -disable_journaling |  
  -delete_oldest_ckp] [-verbose] [-help]
```

Description

The alterdb command sets journaling block characteristics for a database. You can use alterdb to halt journaling for a specified database.

Alterdb can be run only by the DBA or a privileged user running as the DBA.

The following table lists valid command flags and parameters:

Parameter	Description
<i>dbname</i>	<p>The database to be operated on by alterdb. A single database name must be specified.</p> <p>The <i>server_class</i> is specified if required; for details, see the Standard Command Line Flags and Parameters section.</p>
-target_jnl_blocks=<i>n</i>	<p>Specifies the number of journal blocks to be used for the database's journal file, where:</p> <p>$32 \leq n \leq 65536$</p> <p>One and only one database name can be specified.</p> <p> In OpenVMS, this option has no effect on journal files created as part of a OpenVMSccluster merge. </p> <p>The current size can be obtained by the infodb Target journal size parameter.</p>
-jnl_block_size=<i>n</i>	<p>Specifies the size of each journal file block for the database, where:</p> <p>$n = 4096, 8192, 16384, 32768, \text{ or } 65536 \text{ (bytes)}$</p> <p>One and only one database name can be specified. Journaling must be off when this command is issued.</p> <p>The current size can be obtained by the infodb Journal block size parameter.</p>
-next_jnl_file	<p>Causes Ingres to start a new journal file for this database.</p>
-init_jnl_blocks=<i>n</i>	<p>Specifies the size of the <i>first</i> journal file created after a checkpoint is taken (with the ckpdb command), where:</p> <p>$0 \leq n \leq \text{current target journal size}$</p> <p>The target journal size is displayed by infodb and is the parameter set by the -target_jnl_blocks flag.</p>
-disable_journaling	<p>Halts journaling immediately, regardless of whether users are connected to the database.</p> <p>Note: A side effect of using alterdb to disable journaling is that incorrect journaling status is displayed for tables. Tables display journaling as <i>enabled</i> – though journaling is disabled for the database – where <i>enabled after next checkpoint</i> is what would be expected.</p>

Parameter	Description
-delete_oldest_ckp	Deletes the oldest available checkpoint, including related journals and dump files. The request fails if you attempt to delete the only remaining valid checkpoint.
-verbose	Displays system commentary to the standard output device as the alterdb operation continues. This may be used with any one other alterdb parameter.
-help	Displays syntax online.


To restart journaling, you must use the ckpdb +j command.


arcclean

Purges unneeded records from the Replicator shadow and archive tables.

Syntax



arcclean [vnode::]dbname "before_time" [--udba_name] 

arcclean [vnode::]dbname 'before_time' [-udba_name] 

The parameters of the arcclean command are explained in the following table:

Parameter	Description
[vnode::]dbname	Name of the database to be cleaned.
"before_time"	Indicates that all records in the shadow and archive tables dated before the specified date and time are to be purged. Provide the date and time in standard Ingres date and time format. Be sure to place single quotes around the date and time for UNIX and double quotes for OpenVMS and Windows NT. <i>Caution! The date you specify should be before the last successful checkpoint or backup.</i>
udba_name	Name of the database owner.

Description

Use the arcclean command to reclaim disk space and improve performance. After the Replicator server processes data in the local queues, the records remain in the shadow and archive tables. This data continues to grow until you perform cleanup. The arcclean command purges records that are no longer needed. Access to the shadow and archive tables is greatly improved after arcclean is run. You must execute the arcclean command from the DBA account or from an account that has DB_ADMIN privilege on the databases against which you want to run arcclean.

Replicator needs records of the last transactions on each row in order to detect and resolve collisions. Therefore, after arcclean is run, there will still be at least one shadow record for each record in the base table that has been touched by a replicated transaction. However, if there are any records in the input or distribution queues, the associated transactions in the shadow and archive tables also remain. To ensure that records still remaining in input and distribution queues are not removed, the arcclean command selects records eligible for deletion and places them in a temporary table. If arcclean aborts, this temporary table should be removed automatically. If it is not removed, you can safely drop the table manually and then rerun arcclean.

You can modify the storage structures and locations of the shadow table, the shadow table index, and the archive table. The arcclean command remodifies the shadow and archive tables back to their current storage structures after it purges unneeded records.

The arcclean command should be run at intervals using the operating system's job scheduling mechanism, such as cron under UNIX and a batch job under OpenVMS. In UNIX, you can place the arcclean command in a cron file if \$II_SYSTEM/ingres/bin is in the path. In OpenVMS, you can run the arcclean command from a batch job if you have executed the ingdbadef.com command file that defines the arcclean symbol.

Caution! Do not use arcclean if you do not have checkpoints and journals or other types of recovery mechanisms. The data arcclean purges from the shadow and archive tables can be used to aid recovery in the event of a disaster.

Examples

In this example, invoking the arcclean command would require the preparation described in the following steps:

1. Make sure that you have valid checkpoints or backups of all databases to be cleaned. The date you specify for arcclean (in a later step) should be before the checkpoint or backup date.

Removing records without a valid checkpoint could hinder recovery in the event of a system failure. See *reconcil* in this chapter for further details.

2. Deny user access to all databases involved.

There should be no new transactions during the cleaning process. Also, the shadow and archive tables need exclusive locks to the base tables during the remodification procedure.

3. Make sure the input and distribution queues are empty. You can do this by allowing the servers to run until they complete processing of all pending transactions.

If records remain in the queues, the arcclean process will retain the relevant records in the shadow and archive tables on the local database. However, arcclean has no way of knowing which transactions are pending on other databases, and could thus remove records on the local database that are required for an outstanding transaction on a target database. This situation may generate collisions when the outstanding transactions are distributed.

4. Type the arcclean command at the operating system prompt:



```
arcclean nyc::hq -unyc_dba "20-feb-01" ▣
```

```
arcclean nyc::hq -unyc_dba '20-feb-01' ▣
```

auditdb

Audits a database.

Syntax


```
auditdb dbname[/server_class] [-a][-all] |
    [-table=tablename {,tablename}][-file[=filename {,filename}]]]
    [-bdd-yyyymm-dd[:hh:mm:ss]] [-edd-yyyymm-dd[:hh:mm:ss]]
    [#cn] [-iusername] [-inconsistent] [-wait] [-uusername]
    [-help]
```

Description

The auditdb command enables the user to print selected portions of the journal for a database. The user can also create an audit trail of the changes made to particular table(s).

Only the database's DBA or the Ingres system administrator can run the auditdb command on a database.

VMS

If you are using this command against a database in a group level installation, you must have the VMS CMKRNL privilege to run the command. 

The following table lists valid flags and parameters for this command:

Parameter	Description
<i>dbname</i>	<p>The database that is to be audited. A single database name must be specified.</p> <p>The <i>server_class</i> is specified if required; for details, see the Standard Command Line Flags and Parameters section.</p>
-a	Prints journal entries for the system catalogs.
-all	Prints everything in the journal file.
-table= <i>tablename</i> <i>{tablename}</i>	<p>Specifies a particular table or tables for which journal entries are to be printed. Up to 64 <i>tablenames</i> (and 64 <i>filenames</i> if the -file flag is also used) may be specified on the command line. No spaces are allowed in the table list. If this flag is omitted, all tables in the database are audited.</p> <p>This flag is not valid for system catalogs (-a flag).</p> <p>The table name may be qualified with a valid schema name in the format <i>schema.tablename</i>. See the Using Schemas for Owner Qualification section.</p>
-file [= <i>filename</i> <i>{filename}</i>]	<p>Specifies that audit output is to go to one or more files. To use this option, you must specify the -table option on tables of fewer than 1948 bytes per row.</p> <p>If a file list is specified, the number of files must match the number of tables. The audit output of the first <i>tablename</i> goes to the first <i>filename</i>, etc. No spaces are allowed in the file list.</p> <p>If the -file flag is present without a list of file names, auditdb creates default <i>filenames</i> of the form "<i>tablename.trl</i>" (the file extension is an abbreviation of 'trail').</p> <p>If a list of tables is specified without a list of files, output is presented to the standard output device.</p> <p>This flag is not valid for system catalogs (-a flag).</p>

Parameter	Description
-bdd-<i>mmm-yyyy</i> [<i>hh:mm:ss</i>]	Prints journal entries for transactions committed after the specified date and time. If you specify a date and omit the time, the time defaults to 00:00:00 (midnight). If you omit this parameter, auditdb lists transactions starting from the date and time of the most recent checkpoint.
-edd-<i>mmm-yyyy</i> [<i>hh:mm:ss</i>]	Prints journal entries for transactions committed before the specified date and time. If you specify a date and omit the time, the time defaults to 00:00:00 (midnight). If you omit this parameter, auditdb lists transactions through the current system date and time.
#<i>cn</i>	Prints journal entries for transactions committed starting from an older checkpoint. The checkpoint number <i>n</i> must be a valid checkpoint number (as shown by the infodb command). If you omit this parameter, auditdb lists transactions starting from the most recent checkpoint.
-i<i>username</i>	Prints journal entries for actions taken by the specified user.
-inconsistent	Allows you to view journals that the database has marked as inconsistent. The audit will still fail if core catalogs are inconsistent.
-wait	Specifies that auditdb is to wait until journals are current before starting the audit. Auditing begins after all archiving is completed on the database, or after the archiver has copied all log file information up to the log file end-of-file when the auditdb request was initiated.
-u<i>username</i>	Specifies the user for which journal entries are to be printed. See the Standard Command Line Flags and Parameters section.
-help	Displays command options.

The audit files specified with the **-file** flag are in binary (bulk copy) format and contain rows appended to, deleted from, or copied into the tables specified with the **-table** flag. You can copy such a file into a database table that has been created as in the following example:

```
create table auditrel
(date      date not null with default,
username  char(32) not null with default,
operation  char(8) not null with default,
tranid1    integer not null with default,
tranid2    integer not null with default,
table_id1  integer not null with default,
table_id2  integer not null with default,
{columns of tablename})
```

To copy the file audit.trl into the table auditrel, use the following command:

Windows

copy table auditrel () from 'C:\WINNT\Profiles\user1\audit.trl' ▀

UNIX

copy table auditrel () from '/usr/dir/audit.trl' ▀

VMS

**copy table auditrel () from 'dev:[directory]
audit.trl' ▀**

When the copy is finished, auditrel will have a row for each operation against the specified table. The values in each row, corresponding to the columns in the table, are:

Value	Description
date	The date and time of the beginning of the multi-query transaction that contained the operation.
username	The user name of the user who performed the operation.
operation	Contains one of the following: append, repold, repnew, or delete.
transaction id	An 8-byte value composed of two 4-byte integers concatenated. The tranid1 column holds the high order 4 bytes and the tranid2 column holds the low order 4 bytes of the transaction id.
table id	Table_id1 and table_id2 are two 4-byte integers whose values correspond to the values in the columns table_reltid and table_reltidx, respectively, from the itables standard catalog for the table specified.

Auditdb does not necessarily give you a complete list of all transactions since the last checkpoint. There are two reasons for this:

- Since auditdb does not exclusively lock the database, other users may complete a transaction while auditdb is running.
- In some cases, a completed transaction might not yet have been moved from the log files to the journal files.

If you need an absolutely accurate list of transactions since the last checkpoint, make sure all users exit the database before you run auditdb or use the `-wait` flag. Note that if a large amount of unarchived information remains in the log file when auditdb with `-wait` is requested, there will be a delay before the request can be completed.

Examples

To audit the empdata database, use the following command:


auditdb empdata

To audit empdata, creating an audit trail for the employee table, use the following command:


auditdb empdata -table=employee -file sql empdata

```
create table empaudit
(date          date not null with default,
username      char(32) not null with default,
operati       char(8) not null with default,
tranid1       integer not null with default,
tranid2       integer not null with default,
table_id1     integer not null with default,
table_id2     integer not null with default,
eno           I2,
ename         char(10),
age           I1,
job           I2,
salary        money,
dept          I2);
```


Windows

copy table empaudit () from
'C:\WINNT\Profiles\user1\employee.trl' 

UNIX

copy table empaudit () from **"/usr/directory/employee.trl";** 

VMS

copy table empaudit () from **"dev:[directory]employee.trl";** 

Auditing empdata creates audit trails for the employee and address tables. The default files are employee.trl and address.trl:

auditdb empdata -table=employee,address -file

To use specified files, the specified files are created as aud2.trl and aud3.trl:

**auditdb empdata -table=employee,address
-file=aud2.trl,aud3.trl**

ckpdb

Checkpoints a database or selected tables in a database.

Syntax

```
ckpdb dbname[/server_class]  
      [-d] [+j | -j] [-l] [#m[n]] [-mdevice {, device}]  
      [-table=tablename {, tablename}]  
      [-v] [+w | -w] [-uusername] [-help]
```

Description

The ckpdb command creates a new checkpoint for the specified database. If a table list is specified, only the tables on the table list are included in the checkpoint. If journaling is enabled for the database, all journal entries up to this checkpoint are marked as expired. Checkpointing takes place online (while the database is in use) and is transparent to the users. Ckpdb creates the checkpoint, then copies (to the dump file) the log records of any changes to the database that occurred during the checkpoint procedure. Rollforwarddb uses the dump file when it recovers a database that was checkpointed online.



Ingres keeps track of whether the checkpoint is for a table or a database, and will prevent an attempt to roll forward an entire database from a table checkpoint. For table checkpoints, an infodb display of the mode field of the Journal Checkpoint History and Dump Checkpoint History will indicate TABLE.

You can checkpoint a database if you are the Ingres system administrator, DBA, or any Ingres user with the operator privilege.

VMS

If you are checkpointing a database in a group level installation, you must have the VMS CMKRNL privilege to run the command. ■

The following table lists the valid command flags and their parameters:

Parameter	Description
<i>dbname</i>	Specific database to be checkpointed. A single database name must be specified. The <i>server_class</i> is specified if required; for details, see the Standard Command Line Flags and Parameters section.
-d	Destroys all previous checkpoint and journal files.
+j -j	Enables/disables journaling for a database. When this flag is not specified, current journaling status of the database is maintained. If you specify this flag, the checkpoint is performed offline.
-l	Takes an exclusive lock on the database. When you specify the -l flag, you may use the +w or -w flag. If you specify this flag, the checkpoint is performed offline.
#m[n]	For a multi-location database, checkpoint <i>n</i> locations at a time to disk.
-mdevice {, device}	Writes the checkpoint to the specified tape device. If a list of tape devices is supplied, parallel checkpointing will be used for a multi-location database.
	The -m option is not supported on Windows NT. 
-table=tablename {, tablename}	A list of tables to be checkpointed. If multiple tables are specified, no space is allowed between the tables listed. Table checkpoint is not allowed for system catalogs. The database must be journaled to use this parameter. Do not use +j -j with -table .
-v	Verbose mode; displays interim messages as checkpointing proceeds.
+w -w	Flag to wait/don't wait for the database to be free (not in use). Valid only if you have specified the +j , -j , or -l flag. The default is -w .
-uusername	Specifies the effective user for the session. For details see the Standard Command Line Flags and Parameters section.
-help	Displays syntax online.

Use the **-m** flag to write the checkpoint to a specified tape device. You can only write one checkpoint per tape. When you use this flag, it is not necessary to mount the tape device. (When you restore a checkpoint that was created using the ckpdb **-m** command, you must use the rollforwarddb **+c** command.)

If you want to checkpoint a database offline, that is, while it is not in use, you must specify the **-l** flag. Checkpointing a database offline requires the database to be locked.

In an interactive session, if you specify the **-l** flag to perform the checkpoint offline, then you can specify the **+w** | **-w** flag also. This flag, **+w** | **-w**, cannot be used if the backup is performed online.

The **+w** | **-w** flag directs ckpdb to wait (**+w**) or not wait (**-w**) for the database to be free before performing the checkpoint. Since an offline checkpoint requires the database to be locked, this flag allows you to decide whether ckpdb will wait or not for the database to be free if it is in use. If you specify "wait," ckpdb will wait as long as necessary for the database to become free for locking and checkpointing. If you specify "not wait," an error is returned if the database is busy. The default is **-w** (not wait).

By default, ckpdb will sequentially checkpoint data locations one at a time. A database with more than one data location can be checkpointed in parallel.

Examples

To checkpoint and initiate journaling on the empdata database, use the following command:

```
ckpdb +j empdata;
```

To checkpoint the tables employee and dept, use the following command:


```
ckpdb empdata -table=employee,dept
```

To checkpoint the empdata database retaining only the newest checkpoint, use the following command:

```
ckpdb empdata -d;
```

To checkpoint the empdata database to tape, use the following:

UNIX

```
ckpdb -m/dev/rmt0 empdata 
```

VMS

```
ckpdb -mMTA0: empdata 
```

convrep

Converts the Ingres/Replicator data dictionary for use with the current release of Ingres.

Syntax

```
convrep [vnode::] dbname [-udba_name]
```

copyapp

Copies an application, created with Ingres/ Applications-By-Forms (ABF) or Ingres/Vision, from one database to another.

Syntax

```
copyapp out dbname [vnode::dbname/server_class]  
          applname [-ddirname] [-tintfilename] [-lfilename] [-uusername]
```

```
copyapp in newdbname [vnode::dbname/server_class]  
          [-nnewapplname] [-ddirname] intfilename [-lfilename]  
          [-c] [-p] [-q] [-r] [-s[dirname]] [-a[dirname]] [-uusername]
```

Description

The copyapp command copies an application from one database to another. The copyapp out command copies information about the application and its objects to an intermediate text file. The copyapp in command transfers the information from the text file into a database.

Valid parameters for copyapp out are listed in the following table:

Parameter	Description
<i>dbname</i>	The name of the database containing the <i>applname</i> . The <i>vnode</i> and <i>server_class</i> are specified if required; for details, see the Standard Command Line Flags and Parameters section.
<i>applname</i>	The name of the application to be copied.

Parameter	Description
-d <i>dirname</i>	Specifies the directory in which to create the intermediate text file; the default is the current directory.
-t <i>intfilename</i>	Specifies the intermediate text file filename; the default file name is iicopyapp.tmp.
-l <i>filename</i>	Creates a file containing the names of the source files to be copied. For Vision applications, the list includes only custom frames.
-u <i>username</i>	Specifies the effective user for the session. For details, see the Standard Command Line Flags and Parameters section.

Valid flags for copyapp in are listed in the following table:

Flag	Description
-u <i>username</i>	Specifies the effective user for the session. For details, see the Standard Command Line Flags and Parameters section.
<i>newdbname</i>	The name of the database into which the application is to be copied. The <i>vnode</i> and <i>server_class</i> are specified if required. For details, see the Standard Command Line Flags and Parameters section.
-n <i>newapplname</i>	Specifies the name to be assigned to the application in the new database. The default is the same name as in the old database.
-d <i>dirname</i>	Specifies the directory where the intermediate text file is located; the default is the current directory.
<i>intfilename</i>	The name of the intermediate file previously created by the copyapp out. This will be iicopyapp.tmp unless a different intermediate file name was designated with the -t flag of copyapp out.
-l <i>filename</i>	Creates a file containing a list of source files that were copied (or processed if the -a flag was specified). For Vision applications, the list includes only custom frames.
-c	Specifies that the intermediate text file should be deleted.

Flag	Description
-p	Suppresses messages about name conflicts. The default is to display messages.
-q	Specifies that copyapp will be performed as a single transaction. If there is a duplicate name conflict, all changes are rolled back. If you specify the -q flag, copyapp locks system catalogs; for this reason, you should not specify -q when users are connected to the database. In addition, the transaction is logged in the log file; you should be sure that the log file is large enough to accommodate the copyapp transaction.
-r	Specifies that objects with the same name should be replaced (overwritten). By default, duplicate names are not overwritten; instead, the copy is not completed and terminates with an error message.
-s [dirname]	<p>Specifies a new directory for source files. If <i>dirname</i> is omitted, the current working directory is used as the new application's source directory. This flag transfers 4GL source for custom Vision frames, but does not transfer source for non-custom Vision frames. Instead, it marks these frames as new, and source for these frames is regenerated on the next Go or Image operation.</p> <p>This flag is intended for Vision applications. You cannot specify both the -a and the -s flags.</p>
-a [dirname]	<p>Specifies a new source directory for the application, but does not copy source files. If <i>dirname</i> is omitted, then the current working directory is used as the new application's source directory. Any Vision frames are marked as new; source for these frames is regenerated on the next Go or Image operation.</p> <p>This flag is intended for Vision applications. You cannot specify both the -a and the -s flags.</p>

For Vision applications, when performing copyapp in on applications that contain non-custom frames, be sure to specify **-a** or **-s**; if you omit the **-s** or **-a** flag, all frames are marked as custom frames.

See the *Using Forms-Based Application Development Tools* for a further description of this command.

Examples

Copy the Vision new_emp application from the employee database to the employee2 database. Use the default intermediate text file, and use the current working directory as the new application's source directory.

```
copyapp out employee new_emp  
copyapp in -a employee2 icopyapp.tmp
```

copydb

Creates command files to copy and restore a database.

Syntax

```
copydb [-param_file=filename] | [dbname | vnode::dbname[/server_class]]  
[-c] [-row_labels] [-username] [-Ggroupid] [-parallel] [-P]  
[-source=dirname] [-dest=dirname] [-ddirname]  
[-with_tables] [-with_modify] [-with_data] [-all] [-order_ccm]  
[-with_index] [-with_constr] [-with_views] [-with_synonyms]  
[-with_events] [-with_proc] [-with_reg] [-with_rules] [-with_alarms]  
[-with_comments] [-with_roles] [-add_drop] [-infile=filename]  
[-outfile=filename] [-with_permits] [-relpath] [-no_loc] [-no_perm]  
[-no_int] [-no_persist] {tablename | viewname}
```



Description

The copydb command creates command files containing the SQL statements required to copy and restore the tables, views, and database procedures owned by the user. The command creates the following two command files in the current directory:

- copy.out contains SQL commands to copy all tables, views, and procedures owned by the user into files in the specified directory.
- copy.in contains SQL commands to copy the files into tables, recreate views, procedures, and indexes, and perform modifications.

To copy the database, you must execute the SQL commands in the copy.in and copy.out command files.

The following table lists valid command flags and parameters:

Parameter	Description
-param_file=filename	Directs copydb to read <i>filename</i> for all other command line flags, database names, and any other command line arguments. This file must contain only one flag per line (see the examples that follow this table). If this flag is specified, no other flags or arguments can appear on the command line; they must, however, appear in the specified file.
<i>dbname</i>	The name of the database. The <i>vnode</i> and <i>server_class</i> are specified if required; for details, see the Standard Command Line Flags and Parameters section.
-c	Directs copydb to create a printable data file. This is useful for transporting databases between computer systems whose internal representations of non-ASCII data differ. (When you restore a database from a file created using the -c flag, the copy command automatically converts data stored in this format back to the appropriate type.) Copydb cannot represent the following types of data using printable characters: (1) binary data stored in varchar columns, and (2) user-maintained logical keys.
-row_labels	This flag copies the row labels.
-uusername	Specifies the effective user for the session. For details, see the Standard Command Line Flags and Parameters section. Also, see the Using Schemas for Owner Qualification section.
-Ggroupid 	Specifies a group identifier. You must enclose this parameter in double quotation marks (" -Ggroupid "). For details, see the Standard Command Line Flags and Parameters section. You must enclose this parameter in double quotation marks (" -Ggroupid "). 
-parallel	Creates indexes using the parallel index creation syntax (to build multiple indexes concurrently).
-P	Prompts for password if the session requires a password.

Parameter	Description
-source=dirname	<p>The source directory from which the database will be copied in, that is, the directory that contains the data files and from which copy.in will be run. An empty <i>dirname</i> specification ("") denotes the current directory. The -source specification overrides a -d specification for the copy in file.</p> <p>If a source is specified without a destination (no -d or -dest), the default copy out directory is used.</p> <p>The source directory specification is not checked for validity or existence. This allows the scripts to be moved to another machine.</p>
-dest=dirname	<p>The destination directory into which the database will be copied out, that is, the directory where the data files created by copy.out will be stored. An empty <i>dirname</i> specification ("") denotes the current directory. The -dest specification overrides a -d specification for the copy out file.</p> <p>If a destination is specified without a source (no -source) then the default copy in directory is used.</p> <p>The destination directory specification is not checked for validity or existence. This allows the scripts to be moved to another machine.</p>
-ddirname	<p>Stores the copy.in and copy.out files in the specified directory instead of the default current directory. It must be a full filename specification.</p>
- with_tables	<p>Print only the create statements.</p>
- with_modify	<p>Print only the modify statements.</p>
- with_data	<p>Print only the copy statements.</p>
- all	<p>Print all the statements related to the database.</p>
- order_ccm	<p>This flag will order the copy and modify statements for the table. Default is to modify first and then copy; if -CCM is specified, copy first and then modify.</p>
- with_index	<p>Print statements only related to index.</p>
- with_constr	<p>Print statements only related to constraints.</p>
- with_views	<p>Print statements only related to views.</p>
- with_synonyms	<p>Print statements only related to synonyms.</p>

Parameter	Description
- with_events	Print statements only related to event.
- with_proc	Print statements only related to procedure.
- with_reg	Print statements only related to registration.
- with_rules	Print statements only related to rules.
- with_alarms	Print statements only related to a security alarms.
- with_comments	Print statements only related to a comments.
- with_roles	Print statements only related to roles.
- add_drop	This flag will also write a drop statement, before writing the create statements. This will help when the scripts are run repeatedly in case of errors, and tables were already created once.
-infile=	The user can specify an input file name for the copy.in file, so user can run copydb with different options and give different names for infile.
-outfile=	The user can specify an output file name for the copy.out file.
-with_permits	Print statements only related to permits.
- relpath	This removes the paths from the filenames, the files will thus be created and copied from the current directory.
- no_loc	This flag sets all the locations to the default location.
- no_perm	This flag will not print any grant statements.
- no_int	This flag will run the copydb statement uninterrupted for all the tables.
-no_persist	This flag will not write any create table statements for tables which are created with “with persistence” clause.
<i>tablename viewname</i>	<p>The table(s) to be copied. If omitted, all tables are copied. This could also be a list of views; in that case the given views are only copied.</p> <p>The table name may be qualified with a valid schema name in the format <i>schema.tablename</i>. See the Using Schemas for Owner Qualification section.</p>

The name of a file created by copy.out consists of the name of the table followed by an extension made up of the first three letters of the owner's login name. If file names collide, a unique digit replaces the last character of the table name segment.

Windows

The destination directory must be different from the database's actual directory [II_DATABASE]\ingres\data\default\dbname, because the files have the same names as the table files. ■

UNIX

The destination directory must be different from the database's actual directory, \$II_DATABASE/ingres/data/default/dbname, because the files have the same names as the table files. ■

VMS

The destination directory must be different from the database's actual directory, II_DATABASE:[INGRES.DATA.DBNAME], because the files have the same names as the table files. ■

Note: It is important that the database be recreated with copy.in before doing any work (for example, creating tables, forms, applications, or reports) in the new database. After recreating a database, be sure to run sysmod in order to optimize storage structures.

System catalogs cannot be copied using copydb. Use unloaddb to copy a complete database, including system catalogs.

Examples

Windows

To make a copy of olddb, use the following example replacing the named directory (\mydir\backup) with one of your own choice:

```
cd \mydir\backup
copydb olddb
sql olddb<copy.out
```

To create a new database newdb, use the following example:

```
createdb newdb
sqlnewdb<copy.in
sysmod newdb
```

To run copydb with parameters supplied in a file called flagfile, use the following command:

```
copydb -param_file=flagfile
```

where flagfile may contain the following entries:

```
dbname
-order_ccm
-relpath
-no_loc
-all
```

This is equivalent to the command:

```
copydb dbname -order_ccm -relpath -no_loc -all ■
```

To copy mydb to tape, use the following example replacing the named directory (/usr/mydir/backup) with one of your own choice:

```
cd /usr/mydir/backup
copydb mydb /usr/mydir/backup
sql mydb <copy.out
tar c
rm *
```

In order to copy a tape to mydb, use the following example replacing the named directory (/usr/mydir/backup) with one of your own choice:

```
cd /usr/mydir/backup
tar xvpf /dev/rmt0
sql mydb <copy.in
sysmod mydb ■
```

To make a static copy of olddb, use the following example replacing the named default directory (mydir.backup) with one of your own choice:

```
createdb newdb
set default [mydir.backup]
copydb olddb
sql olddb <copy.out
```

Next, to make a copy backup of olddb into newdb, use the following example replacing the named default directory (mydir.backup) with one of your own choice:

```
set default [mydir.backup]
sql newdb <copy.in
sysmod newdb ■
```

UNIX

VMS

createdb

Creates a database.

Syntax

```
createdb dbname [vnode::dbname]/server_class
           [cdbname]
           [-dlocationname] [-clocationname] [-jlocationname]
           [-blocationname] [-wlocationname] [-f product {product}]
           [-language] [-p] [-S][-username]
           [-rlocationname]
```


Description

The createdb command creates a new database. The user who creates a database becomes the Database Administrator (DBA) for that database. The DBA has special privileges and responsibilities regarding the database.

Important! When a database is created, system catalogs are created with the server default page size.

The following table lists valid command flags and parameters:

Parameter	Description
<i>dbname</i>	<p>The name of the database to be created. It must be unique among database names, begin with an alphabetic character, and can have a maximum of 24 alphanumeric characters (the underscore is also allowed).</p> <p>The vnode and server_class are specified if required; for details, see the Standard Command Line Flags and Parameters section.</p> <p>If you are using Ingres/Star, you must specify /star as the <i>/server_class</i>. For examples, see the <i>Star User Guide</i>.</p>
<i>cdbname</i>	Optional parameter for use with Ingres/Star. Overrides the default coordinator database name stored in the Star catalogs. The default name of the coordinator database is the <i>dbname</i> you already specified, prefixed with ii.
-dlocationname	Specifies the location of the database files; the default is the location to which II_DATABASE points.

Parameter	Description
-c <i>locationname</i>	Specifies the location of the checkpoint files; the default is the location to which II_CHECKPOINT points.
-j <i>locationname</i>	Specifies the location of the journal files; the default is the location to which II_JOURNAL points.
-b <i>locationname</i>	Specifies the location of the dump files; the default is the location to which II_DUMP points.
-w <i>locationname</i>	Specifies the location of the work files; the default is the location to which II_WORK points.
-f <i>product</i>	Names specific user interface product(s) for which you want to create catalogs. Allowable product names are ingres, ingres/dbd, vision, windows_4gl, and nofeclients. For more details, see the Standard Command Line Flags and Parameters section. The default is to include all product names. Use nofeclients if you want to copy from an existing database to an empty database using the reload.ing script created by unloaddb.
-l <i>language</i>	Specifies the collating sequence for the database. This sequence must exist in the installation when the createdb statement is issued.
-p	Creates a private database. Only the DBA and those named specifically with the accessdb command have access to the database. Do not use with Ingres/Star.
-S	This flag is used only when you are creating an iidbdb. You must be a privileged user to use this flag. Do not use with Ingres/Star.
	You must enclose this flag in double quotation marks ("-S").
-u <i>username</i>	Specifies the effective user for the session. For details, see the Standard Command Line Flags and Parameters section.
-r <i>locationname</i>	Specifies the new location of the read-only database; typically this will be the CDROM drive where the read-only database is located.

Private Database




By default, all users have access to a database although access to tables in the database must be explicitly granted. To create a private database, use the -p flag.

Locationnames	Before you can reference them in the createdb command, a directory for locationnames must exist. A default location is assumed for any file location that you do not specify. The default locations are created during installation. Additional locations can be created for database, checkpoint, journal, and dump files.
Collating Sequence	The -l flag specifies the collating sequence for the database. A database's collating sequence determines the order in which data is sorted. The collating sequence, specified by the language parameter, must exist in the installation when the createdb statement is issued.

The language parameter can be:

multi	DEC Multinational Character Sequence
spanish	Spanish alphabet's character sequence
<i>collation_name</i>	A custom collation sequence.

The available collation sequences can be viewed by examining the contents of the collation file:

Windows	%IL_SYSTEM%\ingres\files\collation 
UNIX	\$IL_SYSTEM/ingres/files/collation 
VMS	IL_SYSTEM:[INGRES.FILES.COLLATION] 

If the -l flag is not specified, the collating sequence is determined by the value of IL_COLLATION (if this is set). If IL_COLLATION is not set, the default collating sequence is assigned to the database. The default is the native sequence of the ASCII or EBCDIC character set, depending on which is present in your computer.

Note: If createdb fails for any reason, destroy the partially created database using destroydb.

Examples

To create a private database on the default device(s), use the following command:

```
createdb -p mydb
```

To create the public database `ericsdb` using a different user name, use the following command:

createdb ericsdb -ueric

To create a database with its database, checkpoint, and journal files on different devices, use the following command:

createdb bigdb -ddb_ingres -cnewdev_ingres -jotherdev_ingres

To create a database with catalogs for Ingres and OpenROAD, use the following command:

createdb testdb -f ingres windows_4gl

To create a distributed database for use with Ingres/Star:

createdb connie/STAR

dereplic

Removes the Ingres/Replicator database objects (queues and tables, events, and database procedures) from a replicated database.

Syntax

dereplic [*vnode::*] *dbname* [**-udba_name**]

where [*vnode::*] *dbname* is the name of the database to be dereplicated.

destroydb

Destroys an existing database.

Syntax



destroydb *dbname* [*vnode::dbname*
[**-p**] [**-l**] [**-uusername**]

Description

The `destroydb` command removes an existing database. The directory of the database and all files in that directory are removed. You cannot destroy the `iidbdb` using `destroydb`.

If you are using Ingres/Star, the `destroydb` command destroys the distributed database, the coordinator database, and all the Star objects that make up the distributed database. Data in underlying tables in non-coordinator local databases registered in the distributed database are not affected.

To execute this command, you must be the DBA for the database or the Ingres system administrator. The following table lists valid command flags and parameters:

Parameter	Description
<i>dbname</i>	The name of the database. The <i>vnode</i> and <i>server_class</i> are specified if required; for details, see the Standard Command Line Flags and Parameters section.
-p	Requires a prompt to be displayed, asking if you are sure that you want to destroy the database.
	If you want to be prompted for confirmation automatically when you execute the <code>destroydb</code> command, use the following command: <code>destroydb=="ii_system:[ingres.bin]destroydb.exe -p"</code> This command eliminates the need to use the <code>-p</code> flag to obtain a confirmation prompt. 
-l	Confirm if the database is in use, and if in use, return with an error message.
-uusername	Specifies the effective user for the session. For details, see the Standard Command Line Flags and Parameters section.

Examples

To destroy the `empdata` database, use the following command:

`destroydb empdata`

To destroy the `video` database and specify the user name Brad, use the following command:

`destroydb video -uBrad`

esqla (ESQL Preprocessor)

Invokes the specified Ingres embedded SQL (ESQL) preprocessor.

Syntax

```
esqla | esqlc | esqlcc | esqlcbl | esqlf
      [flags] [filename]
```

Description

The ESQL preprocessor commands are used to invoke the preprocessing of source files containing embedded SQL statements.

The languages supported by ESQL preprocessors are listed in the following table:







Language	Preprocessor Command
ADA	esqla
C	esqlc
C++	esqlcc
COBOL	esqlcbl
FORTTRAN	esqlf





The following table lists valid command parameters:







Parameter	Description
<i>flags</i>	The flags that may be passed to individual ESQL preprocessors. These flags are listed in the table below. Note that some flags are applicable only to particular preprocessor languages.
<i>filename</i>	Specifies a filename

The following table lists valid command flags; language-specific flags are noted:

Flag	Description
-{# p}	(esqlc) Generates # line directives to the C compiler (by default, they are in comments). This flag is helpful when debugging the error messages from the C compiler.

	Flag	Description
Windows	-?	(Hyphen and question mark) – Lists the valid command line options. 
UNIX	- -	(Two hyphens) – Lists the valid command line options. 
VMS	-?	(Hyphen and question mark) – Lists the valid command line options. 
	-a	(esqlcbl) Generates output in ANSI format. Use this flag if your source code is in ANSI format and you want to compile the program with the COBOL command line qualifier ansi_format.
		If this flag is omitted, the preprocessor generates output in Compaq COBOL terminal format. 
VMS	-avads	(esqla) Required flag to generate code for VADS on systems that have both an OpenVMS preprocessor and VADS Ada preprocessor. 
	-[no]blank_pad	<p>(esqlc) Tells the preprocessor to pad (-blank_pad) or not pad (-noblack_pad) with blanks any data selected at run time into fixed-length char host variables. Padding is done p to the declared length of the variable, less one byte for the C null terminator.</p> <p>The setting -blank_pad complies with ANSI data retrieval rules for fixed-length char variables.</p> <p>The setting -noblack_pad complies with current data retrieval rules. Instead of blank padding, char data will be null terminated to the length of the data retrieved; -noblack_pad is the default.</p>
	-[no]check_eos	<p>(esqlc) Tells the preprocessor to check (check_eos) or not check (-nocheck_eos) for an end-of-string null terminator on char variables.</p> <p>The setting -check_eos is provided for ANSI SQL92 conformity. It raises an error if a null terminator is missing.</p> <p>The setting -nosqlcode is the default. It turns off the above checking.</p>
	-d	Adds debugging information to the runtime database error messages generated by ESQL. The source file name, line number, and the erroneous statement are printed along with the error message.

Flag	Description
-f [<i>filename</i>]	Writes preprocessor output to the specified file. If the filename variable is omitted, the output is sent to standard output, one screen at a time. If the -f flag is omitted, output is written to a file that has the same base name as the input file, and contains an extension corresponding to the language preprocessor you invoked. For information about filename extensions, consult your host language companion guide.
-iN	(esqlb, esqlc) Sets the default size of integers to <i>N</i> bytes. <i>N</i> must be 1, 2, or 4. 4 is the default setting. (esqlf) Sets the default size of integers to <i>N</i> bytes. <i>N</i> must be either 2 or 4 (the default setting).
	If <i>N</i> =2 is used, the -i2 flag must be specified with the FORTRAN compiler. 
	If <i>N</i> =2 is used, the noi4 qualifier must be used with the FORTRAN compiler. 
-l	Writes preprocessor error messages to the preprocessor's listing file, as well as to the terminal. The listing file includes preprocessor error messages and your source text in a file named <i>filename.lis</i> , where filename is the name of the input file.
-lo	Like -l, but the generated code also appears in the listing file.
-multi	Generates a thread safe code for use in multi-threaded ESQL applications.
-o	If no extension is specified, the include file preprocessor does not create an output file. This does not affect the inclusion of files in the main program. The preprocessor generates a default extension for the translated include file statements unless you specify the -o. <i>ext</i> flag.
-o [. <i>ext</i>]	Specifies the extension of the files to which include files are output after being preprocessed by ESQL. If no extension is given, output files are not generated for include files.
-rN	(esqlb) Sets default size of reals to <i>N</i> bytes. <i>N</i> must be 4 or 8; 4 is the default setting.

Flag	Description
-s	<p>Reads embedded commands from standard input and generates resulting code to standard output. If the -l option is specified with this flag, the listing file is called stdin.lis.</p> <p>To terminate the interactive session, type Ctrl + C. </p> <p> To terminate the interactive session, type Ctrl + D. </p> <p> To terminate the interactive session, type Ctrl + Z. </p> <p></p>
-[no]sqlcode	<p>Tells the preprocessor to assume (-sqlcode) or not assume (-nosqlcode) the existence of a status variable named SQLCODE to receive status information from SQL statements.</p> <p>The -sqlcode setting is provided for ANSI SQL92 conformity.</p> <p>The -nocheck_eos setting is the default.</p>
-w	Prints warning messages.
-wsql = entry_SQL92 open	<p>Issues a warning if the preprocessor detects an embedded SQL statement that does not follow the specified syntax.</p> <p>entry_SQL92 specifies the ANSI.SQL92 entry level standard.</p> <p>open specifies OpenSQL syntax. This flag is useful if you intend to port an application across different Enterprise Access products. Warnings do not halt or affect the success of compilation. This flag does not validate the statement syntax for any SQL Enterprise Access whose syntax is more restrictive than that of OpenSQL.</p>

For a complete description of the SQL preprocessor requirements for your host language, see the related host language companion guide.

extenddb

Extend databases to new or existing locations.

Syntax

```
extenddb -llocation [-uuser] [dbname... | -nodb] [-aarea_dir]  
[-Udata,ckp,jnl,dmp,work | awork] [-rraw_pct] [-drop] [-alter]
```

Description

The **extenddb** command provides a command line interface to extend databases to new locations.

The following table lists valid flags and parameters for this command:

Parameter	Description
-llocation	The name of the location to be extended to.
-uuser	Specifies the effective user for the session. See also the Standard Command-Line Flags and Parameters section in this chapter.
<i>dbname...</i> -nodb	The list of databases to be operated on by extenddb. If no databases are to be extended, the -nodb flag should be passed. This allows for the creation of a location without extending any databases to the new location. The vnode is specified if required; for details, see the Standard Command-Line Flags and Parameters section in this chapter.
-aarea_dir	Specifies the directory the new location will point to. This option is used only when creating a new location and should not be passed when extending a database to an existing location.
-Udata,,ckp,jnl,dmp,work <i>awork</i>	Specifies the usage for the new location. Valid usages include database, checkpoint, journal, dump, work, and auxiliary work.
-rraw_pct	For Raw locations, specifies the percentage of the Raw Area to be allocated to this location. This option is used only when creating a new location and should not be passed when extending a database to an existing location.

Parameter	Description
-drop	Drop the specified location.
-alter	Modify a location's usage to add data, ckp, jnl, dmp, or work areas.

Examples

To extend the stockdb database to use the directory /disk1/loc1 as new data and work areas:

```
extendddb -lextraloc1 -a/disk1/loc1 -Udata,work
```

To create a location without extending it to any databases:

```
extendddb -lextraloc2 -a/disk2/loc2 -Uckp,jnl -nodb
```

To extend the employeedb database to an existing location:

```
extendddb -lextraloc3 -Udata employeedb
```

fastload

Loads binary format files into the database.

Syntax

```
fastload dbname -file = filename -table= tablename
```

If the filename does not exist in the current directory, specify the full path of the file.

Description

The following requirements must be met to use fastload:

- The fastload command must be able to obtain an exclusive lock on the table; otherwise fastload exits.
- The file's data format must match the table's data format.

If the formats do not match, incorrect data will be loaded into the table. For example, if each record in a file contains a 5-byte char and a 4-byte integer and this file is loaded into a table that has a 4-byte char field followed by a 4-byte integer field, fastload would read 8-bytes of the file and load it as a row into the table. This means that the integer field will not contain the actual integer in the original file because the last byte of the 5-byte char field plus 3-bytes of the integer field will be interpreted as a 4-byte integer. The problem is incremented as more data is read since the data will be off by one more byte for each row.

You should calculate the record length in the file and check that it matches the record length specified in the fastload statement. The record length can be found in the row width field output from the SQL help table statement. For more information about this statement, see the *SQL Reference Guide*.

In many cases, fastload is unable to determine the record size of a binary file (on all UNIX platforms this is the case); in these cases, fastload generates a message warning that no format checking will be performed. The warning also contains the expected size of records in the binary file.

- Be aware of the extra data added by the Ingres varchar data type and all nullable fields. The fastload command expects to read a two-byte integer at the beginning of a varchar field that contains the length of the varchar data. All nullable fields should be terminated with a single byte null indicator that indicates the field is null.
- The fastload command supports all standard Ingres table structures when loading into empty tables. It can also load data into heap tables that already contain rows.

All other table types that contain data require a data sort that merges the loaded data with the existing data. Fastload does not perform this function. The data always loads fastest into a heap table with no secondary indexes.

- The fastload command does not support complex data types such as intlist, ord_pair, or udfs.

Examples

To use Fastload, perform the following steps:

1. Make a backup of the table's content.

You need a backup because it may be difficult to fix or eradicate loaded data that is incorrect.

Always check manually that the data has been loaded correctly.

2. Generate a copy of the file you want to fastload by copying it from an Ingres table that has the same format as the target table, or by creating it programmatically.

Do the copy in binary format, for example:

copy test() into 'test.out'

3. Enter the fastload command, for example:

fastload fload -file=test.out -table=test

The table test in the database fload is loaded from the file test.out.

4. Observe that a summary of the load displays, showing the row size, number of rows loaded, and the number of bytes read.
5. Verify manually that the data has been loaded correctly.

Additional Considerations

Binary format files may not be transported to/from byte-swap from/to non-byte swap machines. The data can be generated programmatically, but you should be careful to generate the correct record format, taking into account additional bytes needed for some field types as described in the fastload command description section.

genxml

Exports Ingres tables data into xml format.

Syntax

```
genxml dbname | vnode::dbname[/server_class] [-uuser] [-P] [-GgroupID]
      [-dest=dir] [-xmlfile=filename] [-dtdfile=filename]
      [-metadata_only] [-internal_dtd] [-referred_dtd]
      [{tablename}] [title_doctype=title]
```

Description

This utility allows bulk transfer of Ingres data from an Ingres database into xml format. The generated xml documents refer to generic Ingres DTD (Document Type Definition). Ingres DTD is a controlled document describing the Ingres data. DTD can be internal or external.

When the genxml command is executed on an Ingres database, an xml file is generated, containing the metadata and the data for the tables in xml format, as specified by the Ingres DTD. A DTD file is also generated by genxml by default, unless other options are specified.

For external DTDs, the dtd file may be printed in the same directory as the xml file. Alternatively, external DTDs may be referenced in the xml file to the DTD location. This location will be either a URL or a static location like \$II_SYSTEM/ingres/files/ingres.dtd. Reference to the ingres dtd is made using the DOCTYPE declaration in the generated xml file.

The following table lists valid command flags and parameters:

Parameter	Description
<i>dbname</i>	The name of the database being exported. The <i>vnode</i> and <i>server_class</i> are specified if required; for details, see the Standard Command Line Flags and Parameters section.
-u <i>user</i>	Specifies the effective user for the session. For details, see the Standard Command Line Flags and Parameters section.
-P	Password if the session requires a password.
-G <i>groupID</i>	Specifies a group identifier. For details, see the Standard Command Line Flags and Parameters section.
-dest= <i>dir</i>	The destination directory into which the xml file is generated. An empty <i>dirname</i> specification ("") denotes the current directory. The generated Ingres dtd will also be placed in this directory.
-xmlfile= <i>filename</i>	The name of the output xml file. By default the xmlfile is called <i>xmlout.xml</i> .
-dtdfile= <i>filename</i>	The name of the output dtd file. By default this file is called <i>ingres.dtd</i> .
-metadata_only	If this flag is specified, only the metadata information is printed out.
-internal_dtd	This flag prints the DTD inline inside the xml doc.
-referred_dtd	This flag places a reference to the <i>ingres.dtd</i> in Ingres files directory (<i>\$II_SYSTEM/ingres/files</i>).
<i>tablename</i>	Table name or names that the user wants the xml to output. Table names must be separated by spaces. If omitted, all tables are copied. The table name may be qualified with a valid schema name in the format <i>schema.tablename</i> . See the Using Schemas for Owner Qualification section.
-title_doctype= <i>title</i>	This flag allows the users to change the doctype or the document name of the XML file. The default doctype is <i>IngresDoc</i> . This flag should be used with caution. If the document name is changed, the <i>referred_dtd</i> option should not be used as the referred generic Ingres DTD in <i>\$II_SYSTEM/ingres/files</i> still has <i>IngresDoc</i> as document type.

The xml generation utility *genxml* copies out the Ingres data in xml format. This allows Ingres data on the web to communicate with other third party products.

Examples

To generate a copy of a data base testdb in xml format, run:

genxml testdb

This produces files “xmlout.xml” and “ingres.dtd” in the current directory.

xmlout.xml contains the metadata and the data of all the tables of the database testdb.

To output only table tab1 and produce the files in directory (/tmp/mydirectory) with the filename myxml.xml and with DTD file placed inline with the xml file type the following command:

genxml testdb -dest="/tmp/mydirectory" -xmlfile=myfile.xml -internal_dtd tab1

Only information about the table tab1 will be generated.

The -metadata_only flag may be useful in generating the metadata information in xml form in cases when you wish to recreate objects at any other location or installation without copying what may be a large amount of data.

The generated xml file can be input to the xmlimport utility for importing into another Ingres installation. (See the xmlimport section for more details on xmlimport utility).

infodb

Provides a variety of information about a database.

Syntax

infodb [dbname[/server_class]][#c[n]][-username][-help]

Description

The infodb command displays information on the status of the database, a history of checkpoints and journaling, as well as information from the database's configuration file 'aaaaaaa.cnf' concerning the location of database files. If no database is specified, infodb prints a report for each database.

You must be a privileged user or the DBA of the specified database to use this command. If you are a privileged user, you can use the `-u` flag to impersonate another user.

VMS

If you are using this command against a database in a group level installation, you must be a privileged user (VMS CMKRNL, SYSPRV, and PHY_IO privileges) to run the command. ■

The following table lists valid parameters and command flags:

Parameter	Description
<i>dbname</i>	The name of the database. The <i>server_class</i> is specified if required; for details, see the Standard Command Line Flags and Parameters section.
#c[n]	Provides detailed information about a specific checkpoint for the database. The checkpoint number <i>n</i> must be a valid checkpoint number. If <i>n</i> is omitted, information about the most recent completed checkpoint is displayed.
-uusername	Specifies the effective user for the session. For details, see the Standard Command Line Flags and Parameters section.
-help	Displays syntax online.

Example

The following example shows the output from `infodb`:

```
=====11-NOV-2001 14:05:55.02 Database Information=====
Database : (doc,kbref) ID : 0x2D5BF682          Collation: default
Extents : 5    Last Table Id : 171
Config File Version Id : 0x00050001    Database Version Id : 5
Mode: DDL ALLOWED, ONLINE CHECKPOINT ENABLED
Status  : VALID, JOURNAL, CKP, DUMP, ROLL_FORWARD, CFG_BACKUP

    The Database has been Checkpointed.
    The Database is Journalled.

    Journals are valid from checkpoint sequence : 1

-----Journal information-----
Checkpoint sequence :      3    Journal sequence :      3
Current journal block :      2    Journal block size :    16384
Initial journal size :      4    Target journal size :     512
Last Log Address Journalled : <760829464:19666:188>
-----Dump information-----
Checkpoint sequence :      3    Dump sequence :      1
Current dump block :      1    Dump block size :    16384
Initial dump size :      4    Target dump size :     512
```

Last Log Address Dumped : <760829464::21100:100>						
-----Checkpoint History for Journal-----						
Date		Ckp_sequence	First_jnl	Last_jnl	valid	mode

11-NOV-2001 13:23:43.57		1	1	1	1	OFFLINE
11-NOV-2001 13:24:50.40		2	2	2	1	ONLINE
11-NOV-2001 13:58:52.65		3	3	3	1	ONLINE, TABLE
-----Checkpoint History for Dump-----						
Date		Ckp_sequence	First_dmp	Last_dmp	valid	mode

11-NOV-2001 13:58:52.65		2	1	1	1	ONLINE
11-NOV-2001 13:58:52.65		3	1	1	1	ONLINE, TABLE
-----Cluster Journal History-----						
Node ID	Current Journal	Current Block	Last Log Address			

0	1	1	<760829462:4731:1592>			
1	3	10	<760829464:35000:188>			
None.						
-----Extent directory-----						
Location	Flags	Physical_path				

ii_database	ROOT, DATA	c:\IngresII\Ingres\data\default\iiidbdb				
ii_journal	JOURNAL	c:\IngresII\Ingres\jnl\default\iiidbdb				
ii_checkpoint	CHECKPOINT	c:\IngresII\Ingres\ckp\default\iiidbdb				
ii_dump	DUMP	c:\IngresII\Ingres\dmp\default\iiidbdb				
ii_work	WORK	c:\IngresII\Ingres\work\default\iiidbdb				

The following example shows the output from the database information section. See the table that follows for a description of each field in this output:

- (1)=====11-FEB-2001 14:05:55.02 Database Information=====
- Database : (doc,kbref) ID : 0x2D5BF682
 Extents : 5 Last Table Id : 171
 Config File Version Id : 0x00040001 Database Version Id : 5
 Status : VALID, JOURNAL, CKP, DUMP, ROLL_FORWARD, CFG_BACKUP
- (2) The Database has been Checkpointed.
 The Database is Journalled.
- Journals are valid from checkpoint sequence : 1

The following table describes the fields in the preceding database information output:

Field	Description
At (1)	Date and time the infodb operation was run.
Database	The name (doc) and owner (kbref) of the database.
ID	The internal identifier of the database.
Extents	Number of locations the database is using.
Last Table ID	The integer identifier assigned to the last created table.

Field	Description
Config File Version ID	Major (upper 2 bytes) and minor (lower 2 bytes) versions of the configuration file.
Database Version ID	The version of DMF that created the database. Note that this is <i>not</i> related to the Ingres version of the database.
Status	<p>Displays status information for the database. The following status abbreviations may appear:</p> <p>CFG_BACKUP—automatic backup of the configuration file is enabled.</p> <p>CKP—indicates that you must perform a rollforward +c (back to saveset) <i>before</i> you can do a rollforward -c +j.</p> <p>DUMP—the database has undergone dump processing (that is, a dump file was created in the dump location) via some online checkpoint.</p> <p>JOURNAL—the database is journaled.</p> <p>JOURNAL_DISABLED—journaling has been disabled.</p> <p>NOLOGGING—the database has been opened by a set nologging session. Note that if this session encounters an error, the database will be marked inconsistent.</p> <p>ROLL_FORWARD—indicates that rollforward is available on the database and has not been run to completion since the last checkpoint was taken.</p> <p>SMINC—indicates the system catalogs are in an inconsistent state.</p> <p>VALID—the database is consistent and available for use. If this does not appear, the database is marked inconsistent.</p>
At (2)	This section displays comments regarding the status of the database. Important state information is highlighted here.

Field	Description
The Database is Inconsistent. Cause of Inconsistency: <...>	<p>Shown if the database is inconsistent. The cause of inconsistency will be one of the following:</p> <p>NOLOGGING_ERROR— a transaction failed while the database was in the nologging state.</p> <p>NOLOGGING_OPENDB— the database was opened for the first time, but was in the nologging state. This means a session exited abnormally.</p> <p>OPEN_COUNT— the database was opened for the first time, but the database open count in the configuration file was <i>not</i> zero. This means the configuration file could not be read during a recovery attempt.</p> <p>REC_OPEN_FAILURE— the RCP could not recover a database because the database could not be opened.</p> <p>RECOVER_ERROR— the RCP failed to recover a database due to an unexpected logging system or recovery protocol problem.</p> <p>REDO_ERROR— the RCP failed to recover a database due to an error in REDO processing.</p> <p>RFP_FAIL— the rollforward of the database level checkpoint failed.</p> <p>UNDO_ERROR— the RCP failed to recover a database due to an error in UNDO processing.</p> <p>WILL_COMMIT_ERR— the RCP was unable to restore a transaction to the willing commit state.</p>
The Database has been Checkpointed.	Shown if the database has been checkpointed.
The Database is Journalled. or The Database is not Journalled.	Shows the journaling status.
Journaling has been disabled on this database by alterdb. Run 'ckpdb +j' to re-enable journaling.	Shown if journaling has been disabled.

Field	Description
Database is being accessed with Set Nologging, allowing transactions to run while bypassing the logging system.	Shown if a set nologging session is active on the database.
Journals are valid from checkpoint sequence: <i>checkpoint sequence number</i>	Shows the earliest checkpoint from which rollforward is allowed.
Journals are not valid from any checkpoint.	Shown if rollforward is not valid from any checkpoint, or there are no checkpoints.

The following example shows the output from the journal information section:

```

---Journal information---
Checkpoint sequence :    3   Journal sequence :           3
Current journal block :    2   Journal block size :       16384
Initial journal size :    4   Target journal size :         512
Last Log Address Journalled : <760829464:19666:188>

```

This table describes the fields in the journal information output:

Field	Description
Checkpoint sequence	The current checkpoint sequence number. Incremented when a checkpoint operation is performed.
Journal sequence	The current journal file sequence number.
Current journal block	The current journal file block sequence number. This is the logical end-of-file of the current journal file.
Journal block size	The block size of the current journal file, in bytes.
Initial journal size	The number of blocks initialized in the “first journal file” when it is created. The first journal file is the journal file created during the checkpoint +j operation. Subsequent journal files created before the next checkpoint is done will <i>not</i> be initialized.
Target journal size	The size, in blocks, to which the current journal file may grow before a new journal file should be created. A new journal file will be created at the start of the next archive cycle after the current journal file reaches this size.

Field	Description
Last log address journaled	The log address (<i>log sequence number, log page number, log word offset</i>) of the last log record written to a journal file.

The following example shows the output from the dump information section:

```

---Dump information---
Checkpoint sequence :      3   Dump sequence :      1
Current dump block :      1   Dump block size :    16384
Initial dump size :      4   Target dump size :     512
Last Log Address Dumped : <760829464::21100:100>

```

This table describes the fields in the dump information output:

Field	Description
Checkpoint sequence	The current checkpoint sequence number. Incremented when a checkpoint operation is performed.
Dump sequence	The current dump file sequence number.
Current dump block	The current dump file block sequence number. This is the logical end-of-file of the current dump file.
Dump block size	The block size of the current dump file, in bytes.
Initial dump size	The initial allocation of the current dump file, in blocks. The number of blocks initialized when a dump file is created.
Target dump size	The size, in blocks, to which the current dump file may grow before a new dump file should be created. A new dump file will be created at the start of the next archive cycle after the current dump file reaches this size.
Last log address dumped	The log address (<i>log sequence number, log page number, log word offset</i>) of the last log record written to a dump file.

The following example shows the output from the checkpoint history information section:

```

---Checkpoint History for Journal---
Date           Ckp_sequence  First_jnl  Last_jnl  valid  mode
11-NOV-2001 13:23:43.57      1          1          1      1 OFFLINE
11-NOV-2001 13:24:50.40      2          2          2      1 ONLINE
11-NOV-2001 13:58:52.65      3          3          3      1 ONLINE,
                                TABLE

```

The following table describes the fields in the checkpoint history information output:

Field	Description
Date	The date and time the checkpoint operation was done.
Ckp_sequence	The sequence number of the checkpoint.
First_jnl	The journal sequence number of the first (or oldest) journal file corresponding to the checkpoint.
Last_jnl	The journal sequence number of the last (or youngest) journal file corresponding to the checkpoint.
Valid	Indicates whether or not the checkpoint is valid (1 implies valid, 0 implies invalid).
Mode	Indicates whether the checkpoint operation was online or offline. Also indicates TABLE if the checkpoint was a table checkpoint.

To recover the entire database, you need to specify #c2, for example, in the rollforwarddb command to roll forward from the database checkpoint. Checkpoint 3 was taken on selected table(s).

The following example shows the output from the dump checkpoint history information section:

Checkpoint History for Dump						
Date	Ckp_sequence	First_dmp	Last_dmp	valid	mode	
11-NOV-2001 13:58:52.65	2	1	1	1	ONLINE	
11-NOV-2001 13:58:52.65	3	1	1	1	ONLINE, TABLE	

The following table describes the fields in the database's online checkpoints output:

Field	Description
Date	The date and time the checkpoint operation was done.
Ckp_sequence	The sequence number of the checkpoint.
First_dmp	The dump sequence number of the first (or oldest) dump file corresponding to the checkpoint.
Last_dmp	The dump sequence number of the last (or youngest) dump file corresponding to the checkpoint.

Field	Description
Valid	Indicates whether or not the checkpoint is valid (1 implies valid, 0 implies invalid).
Mode	Indicates whether the checkpoint operation was online or offline (should always be online). Also indicates TABLE if the checkpoint was a table checkpoint.

The following example shows the output from the cluster journal history section:

Cluster Journal History			
Node ID	Current Journal	Current Block	Last Log Address
0	1	1	<760829462:4731:1592>
1	3	10	<760829464:35000:188>

The following table describes the fields in the cluster journal history output:

Field	Description
Node ID	The integer identifier of the node.
Current journal	The node's current journal file sequence number.
Current block	The node's current journal file block sequence number. This is the logical end-of-file of the node's current journal file.
Last log address	The log address (<i>log sequence number, log page number, log word offset</i>) of the last log record written to a journal file on this node.

The following example shows the output from the extent directory section:

Extent directory		
Location	Flags	Physical_path
ii_database	ROOT, DATA	c:\IngresII\Ingres\data\default\iiddbdb
ii_journal	JOURNAL	c:\IngresII\Ingres\jnl\default\iiddbdb
ii_checkpoint	CHECKPOINT	c:\IngresII\Ingres\ckp\default\iiddbdb
ii_dump	DUMP	c:\IngresII\Ingres\dmp\default\iiddbdb
ii_work	WORK	c:\IngresII\Ingres\work\default\iiddbdb

The following table describes the fields in the extent directory output, which shows all locations being used by the database:

Field	Description
Location	The logical name of the location.
Flags	Indicates the types of database files being stored in

Field	Description
	the location. The possibilities are: ALIAS—this is a location alias. This means at least one other location points to the same area as this location. This flag is used only for checkpoint and rollforward operations so that locations are neither checkpointed nor rolled forward more than once. AWORK—this is an auxiliary work file location. Work files (that is, for sorts and temporary tables) are stored in the location. CHECKPOINT—checkpoint files are stored in the location. DATA—user data (such as tables, indexes) is stored in the location. DUMP—dump files are stored in the location. JOURNAL—journal files are stored in the location. ROOT—system data (that is, system catalogs) is stored in the location. WORK—work files (for sorts and temporary tables) are stored in the location.
Physical_path	The physical path of the location.

ingmenu

Starts Ingres Menu.



Syntax

```
ingmenu dbname [vnode::dbname]/server_class
          [-e] [-uusername] [-Ggroupid]
```

Description

The `ingmenu` command invokes Ingres Menu, a forms-based interface for accessing the Ingres tools. See the *Using Character-Based Querying and Reporting Tools* guide for a complete description of Ingres Menu.

The following table lists valid command flags and parameters:

Parameter	Description
<i>dbname</i>	The name of the database. The <i>vnode</i> and <i>server_class</i> are specified if required; for details, see the Standard Command Line Flags and Parameters section.
-e	Invokes Ingres Menu in empty mode. This flag is passed to the QBF, RBF, TABLES, and VIFRED options of Ingres Menu. In essence, it causes any catalog of applications, Join Definitions, tables, reports, or other objects to be initially displayed empty, so that the user can enter specific names of such objects.
-uusername	Specifies the effective user for the session. For details, see the Standard Command Line Flags and Parameters section.
-Ggroupid	Specifies a group identifier. For details, see the Standard Command Line Flags and Parameters section.
	You must enclose this parameter in double quotation marks ("-Ggroupid"). 

ingprenv

Displays the Ingres installation environment variables.

Syntax

ingprenv [*variable_name*]

Description

The `ingprenv` command displays values for the Ingres variables defined at installation time. These include:

```
ING_ABFDIR
ING_EDIT
II_CHARSETxx
II_CHECKPOINT
II_COLLATION
II_CONFIG
II_DATABASE
II_DUMP
II_INSTALLATION
II_JOURNAL
II_LANGUAGE
II_MSGDIR
II_TEMPORARY
II_TIMEZONE_NAME
II_WORK
TERM_INGRES
```

Windows

This command reads the `symbol.tbl` file in `%II_SYSTEM%\ingres\files`. To add or remove symbols, use the `ingsetenv`, `ingunset`, and `ingprenv` commands. ■

UNIX

This command reads the `symbol.tbl` file either in `$II_SYSTEM/ingres/files` or, for client installations, in `$II_SYSTEM/ingres/admin/node_name`. ■

To add or remove symbols for:

- Local installation—use `ingsetenv`, `ingunset`, and `ingprenv`
- Global operations (server and client installations)—use `ingsetall`, `ingunsetall`, and `ingprall`

Caution! *Never edit the `symbol.tbl` directly.*

Windows

Use `ingprenv` with the optional *variable_name* to check the value of a particular symbol or variable. For example, to examine the value of `II_DATABASE`, enter:

UNIX

ingprenv II_DATABASE

If you do not specify *variable_name*, all of the installation variables are displayed. If the variable is not defined, no output is returned. ■

ingsetenv

Sets an Ingres installation environment variable.

Syntax

ingsetenv *variable_name value*

Description

The **ingsetenv** command allows you to set or change an Ingres environment variable.

The *variable_name* identifies the installation variable that you want to set or change.

The *value* specifies the value to which you want to set the installation variable.


For example, the following command line sets the ABF directory variable:

Windows

ingsetenv ING_ABFDIR \proj\abf 

UNIX

ingsetenv ING_ABFDIR /proj/abf

You cannot use this command to set Windows or UNIX environment variables. This command affects only the Ingres environment variables stored in the Ingres symbol table. These installation variables can be viewed with the **ingprenv** command. 

ingunset

Windows

Unsets an Ingres installation environment variable from the Ingres symbol table.

Syntax

ingunset *variable_name*

Description

The `ingunset` command deletes the specified Ingres environment variable from the Ingres symbol table.

The *variable_name* identifies the installation variable that you want to unset.

Note: You cannot use this command to unset Windows or UNIX environment variables. This command affects only the Ingres installation environment variables stored in the Ingres symbol table. These installation variables can be viewed with the `ingprenv` command. ■

ipm

Combines the functions of the `lockstat`, `logstat`, `iimonitor`, and `iinamu` utilities in a single forms-based tool.

Syntax

ipm *options*

Description

The Ingres/Interactive Performance Monitor (IPM) is used to view different aspects of a running installation. It can be used to view a running server, examine the logging and locking system, and perform some actions on active servers. It can:

- Monitor the locking system:
 - Determine which tables/pages are locked
 - Find who is holding locks
 - Aid in debugging concurrency problems
- Monitor the logging system:
 - View logging statistics (efficiency and tuning information)
 - Display which processes can be logged
 - Display which databases have transactions
 - Display which transactions are active
 - Terminate servers or sessions

The IPM startup options are summarized in the following list:

-d <i>dbname</i>	Report only on resources for database <i>dbname</i> .
-s	Also display locklists that contain no locks.
-l	Report on all resource types (page, table, database, etc.).
-l <i>restype</i>	Report only on the specified resource type.
-n	Print resources granted in NL (null) mode.
-t	Report on a particular table. With this parameter you must also specify a database.
-rseconds	Set refresh time for various screens.

For complete details on the use of the IPM command, see Visual DBA online help.

isql

Invokes the interactive forms-based Ingres/Terminal Monitor for Interactive SQL (ISQL).

Syntax

isql [*SQL option flags*] *dbname* | *vnode::dbname*[/*server_class*]

Description

The **isql** command invokes the forms-based Terminal Monitor for use with SQL query language. See *Using Character-Based Querying and Reporting Tools* for a complete description of the forms-based Terminal Monitor.

The following table lists valid command flags and parameters:

Parameter	Description
<i>SQL option flags</i>	The flags that are passed when the ISQL Terminal Monitor is invoked. These include the standard flags -u <i>username</i> , -G <i>groupid</i> , and -R <i>roleid</i> as wells as formatting and DBMS control flags. For details see the sql command.

Parameter	Description
<i>dbname</i>	The name of the database. The <i>vnode</i> and <i>server_class</i> are specified if required; for details, see the Standard Command Line Flags and Parameters section.

Example

To invoke interactive forms-based SQL on the employee database, type the following command:

```
isql employee
```

modifyfe

Modifies the storage structure of catalogs for Ingres querying and reporting tools.

Syntax

```
modifyfe dbname [vnode::dbname[/server_class]]  
[-uusername] [+w | -w] {product}
```

Description

The modifyfe command modifies the storage structure of catalogs for Ingres tools such as OpenROAD, Ingres/Vision, or Ingres/ Applications-By-Forms (ABF).

The following table lists valid command parameters and flags:

Parameter	Description
<i>dbname</i>	The name of the database containing the catalogs to be modified. The <i>vnode</i> and <i>server_class</i> are specified if required; for details, see the Standard Command Line Flags and Parameters section.

Parameter	Description
-u <i>username</i>	Specifies the effective user. If you want to modify a database you do not own, you must use this flag to specify the DBA's user name. For details, see the Standard Command Line Flags and Parameters section.
+w -w	Wait/don't wait for the database to be free (not in use). The default is -w. If you specify -w or if you do not specify this flag, then modifyfe aborts immediately if the database is not free (another session has an exclusive connection). If you specify +w then modifyfe waits for anyone with an exclusive connection to disconnect, then proceeds.
<i>product</i>	Specifies the products for which you want to modify catalogs. Allowable <i>product</i> names are ingres, ingres/dbd, vision, and windows_4gl. For details, see the Standard Command Line Flags and Parameters section. If you omit this parameter, all user interface catalogs are modified.

Like upgradedb, modifyfe takes an exclusive lock on the database.

The modifyfe command is normally not called directly. It is called by upgradedb to perform the required catalog updates.

netutil

Forms-based command for configuring Ingres/Net. The Net Management Utility allows you to store and manage the information—connection data and remote user authorizations—needed by the communications and Bridge servers to connect to remote installations.

Syntax

netutil [-u*user*] [-v*node vnode*] -f*ile=filename{,filename}*

The parameters are described in the following table:

Parameter	Description
<i>user</i>	Specifies the effective user name for the session. When creating private connection information, the information will be stored for the specified user.

Parameter	Description
<i>vnode</i>	The name of the remote node on which the connection information is to be stored. This vnode name must have been configured through netutil previously.
<i>file</i>	When the -file switch is specified, netutil operates non-interactively, and all statements in the specified control file are executed. Note: When the input file name is specified as - (a single dash character), input is taken from the standard input channel. This allows the user to enter commands directly from the keyboard.

Description

Netutil is a forms-based program for configuring Ingres/Net. This command allows any number of Ingres sites to be connected together in a single network.

Examples

To edit private connection information for the user emma, type the following command:

netutil -uemma

To edit connection information for the previously defined node new_york, type the following command:

netutil -vnew_york

To run netutil in interactive mode, taking input from the keyboard, for the user emma on the remote node new_york, type the following command:

netutil -uemma -vnew_york -file-

optimizedb

Generates statistics for use by the Ingres Query Optimizer.

Syntax

```
optimizedb [SQL option flags][-i filename]  
[-o filename] [-z flags]  
dbname [vnode::dbname/server_class]  
{-rtablename {-acolumnname}} | {-xrtablename} [-help]
```

Description

The **optimizedb** command generates statistics on the specified columns. These statistics are stored in system catalogs (iistats and iihistograms) and used by the Ingres Query Optimizer to select an efficient query processing strategy.

Complete and accurate statistics in the system catalogs generally result in more efficient query execution strategies and faster system performance. The process of generating complete and accurate statistics requires some time, but a balance between accurate statistics and the time to generate them can be achieved by specifying the **-zx** or **-zs** flag, described below. Statistics need to be refreshed only when a significant change in the distribution of a column's values has occurred.

The statistics generated by the **optimizedb** command for any column consist of two elements: (1) the number of unique values in a column, and (2) a histogram with a variable number of variable-width cells. The accuracy of the histograms can be controlled by the **-zu#** and **-zr#** flags described below. Increasing the number of cells in the histograms increases the amount of space required for the **iihistograms** table and thus increases somewhat the amount of space and time used by the optimizer. However, the increased accuracy of the statistics will generally result in more efficient query execution strategies.

It is recommended that you generate the statistics for all columns that appear in the qualification (where clause) of a query statement. If statistics are missing or incorrect, the query will still execute, although the speed of query processing can be affected.

After running **optimizedb**, it is prudent to run **sysmod**. This is especially true the first time **optimizedb** is run on a database.

Note: Although **optimizedb** does not lock the database or individual tables while it is retrieving values and generating statistics, once the statistics have been collected and stored in the appropriate catalogs, **optimizedb** does take an exclusive lock on the database or individual tables in order to complete its task.

The following table lists valid command flags and parameters:

Parameter	Description
<i>SQL option flags</i>	<p>SQL option flags on the optimizedb command line are automatically passed. The optimizedb command accepts the following SQL option flags:</p> <p>[+ -]U -u -cN -tN -ikN -fkxM.N [+ -]w -xk</p> <p>For a complete description of these flags, see the sql command.</p>
-z flags	Specify options to optimizedb. These flags are listed in the following table.
-i filename	<p>If this flag is specified, optimizedb reads statistics from <i>filename</i> instead of operating directly on the database. <i>Filename</i> must be the name of a file that has been generated by the statdump command using the -o flag. This file is in ASCII format and can be edited. However, only two types of changes are acceptable: a) you can modify values and b) you can add rows describing cells.</p> <p>Do not change the format of the file, that is, do not change the order in which data appears or add an incomplete new row.</p> <p>When the -r and -a flags are used in conjunction with this flag, they act as filters. Optimizedb will only read in from the file those statistics that belong to the specified table or column.</p> <p>Optimizedb does not use the row and page count values in the file unless the -zp flag is also specified.</p> <p>Note: These values are vital for correct operation of the DBMS. If you use the -zp flag, be sure to put new values for row and page counts in iitables.</p>
-o filename	Write the optimizedb output to the specified file instead of to the system catalogs.

Parameter	Description
<i>dbname</i>	The name of the database. The <i>vnode</i> and <i>server_class</i> are specified if required; for details, see the Standard Command Line Flags and Parameters section.
-rtablename	Specifies <i>tablename</i> s to be processed by the optimizer. If no table name is specified, then all columns for all tables in the database are processed. The table name may be qualified with a valid schema name in the format <i>schema.tablename</i> . See the Using Schemas for Owner Qualification section. If <i>tablename</i> specifies a secondary index name, optimizedb creates a composite histogram on the key columns comprising the index.
-xrtablename	Specifies <i>tablename</i> s to be excluded from processing by the optimizer. Except for these, all columns for all tables in the database are processed.
-acolumnname	If the -rtablename flag is specified, then (and only then) individual columns can be specified with the -a flag for the generation of statistics. When table(s) and column(s) are specified, then statistics processing is limited to the specified columns (plus any included through the -zk flag, described in the following table).
-help	Displays syntax online.

Note: The combination of **-rtablename** and **-xrtablename** parameters is not permitted in a single optimizedb request, nor is the combination of **-xrtablename** and **-acolumnname**.

The **-z** flags that are available with the optimizedb command are described in the following table:

Parameter	Description
-zf <i>filename</i>	Directs optimizedb to read <i>filename</i> for all other command line flags, database names, and any other command line arguments. This file must contain only one flag per line (see the examples below). If this flag is specified, no other flags or arguments can appear on the command line; they must, instead, appear in the specified file.

Parameter	Description
-zc	Directs optimizedb to optimize the system catalogs in addition to the base tables. If you want to optimize selected system catalogs, rather than all of them, use this flag and specify the individual tables with the -r flag. This flag is valid only if the user issuing the command is the DBA for the specified database.
-zcpk	Requests a composite histogram on primary key structure.
-zdn	Directs optimizedb to use its algorithm to estimate the number of distinct values and repetition factor for a column whose histogram is built with sampling (see the -zs# option).
-zh	Prints the histogram that was generated for each column. This flag also implies the -zv flag.
-zk	Generates statistics for columns that are keys on the table or are indexed, in addition to columns specified on the command line.
-zlr	Reuse existing repetition factor if there is one.
-zn#	Directs optimizedb to read floating point numbers using the precision level specified by #. Use this flag in conjunction with the -i <i>filename</i> flag.
-zp	Directs optimizedb to read the row and page count values in the file specified with the -i flag and to store those values in the appropriate system catalog (they can be viewed in itables).
-zr#	Specifies the maximum number of cells that the histogram can contain if optimizedb creates an inexact histogram. In an inexact histogram, each cell represents a range of values. The allowable range is 1<#<500 (that is, the minimum is 2 and the maximum is 499). The default number of cells is 15.
-zs[s]#	Creates statistics based on sample data. The percentage of table rows sampled is determined by the value of #. This number must be a floating point number in the range of 0 to 100. Specifying the optional s (-zss) will cause the tuple identifiers (TIDs), which are used to retrieve the sample rows, to be sorted before the rows are retrieved. This decreases retrieval time but increases the amount of memory used by optimizedb.

Parameter	Description
-zu#	<p>Specifies the maximum number of cells an exact histogram can contain. In an exact histogram, each cell represents a single, unique value.</p> <p>The allowable range is $0 < \# < 250$ (that is, the minimum is 1 and the maximum is 249).</p> <p>The default number of cells is 100.</p>
-zv	Prints information about each column as it is being processed.
-zw	Sets the complete flag, which indicates whether a column contains all possible values. The range of values in a column affects query optimization. By default, columns are assumed to be not complete.
-zx	Directs optimizedb to determine only the minimum and maximum values for each column rather than full statistics. Because minimum and maximum values for columns from the same table can be determined by a single scan through the table, this flag provides a quick way to generate a minimal set of statistics. Minimal statistics cannot be created on columns holding only null values.

Examples

To generate full statistics for all columns in all tables in the empdata database, type the following command:

optimizedb empdata

To generate statistics for key or indexed columns in the employee and dept tables and, additionally, generate statistics for the dno column in the dept table, type the following command:

optimizedb -zk empdata -remployee -rdept -adno

To do the same as the above example, but from a file, type the following command:

optimizedb -zf flagfile

where flagfile contains:

```
-zk  
empdata  
-remployee  
-rdept  
-adno
```

The command that follows allows you to:

- Generate statistics for all key or indexed columns in employee, dept, and salhist.
- Process the eno column in employee, whether or not eno is a key or indexed column.
- Generate statistics with only minimum and maximum values from the columns.
- Print status information as each column is processed. To proceed, type the following:

```
optimizedb  
-zk  
-zv  
-zx  
empdata  
-remployee  
-aeno  
-rdept  
-rsalhist;
```

To allow up to 100 unique values from each column in the employee table before merging adjacent values into the same histogram cell, type the following command:

```
optimizedb  
-zu100  
empdata  
-remployee;
```

qbf

Invokes Ingres/Query-By-Forms (QBF).

Syntax

```
qbf dbname [vnode::dbname]/server_class
      [-mmode] [[-t] [-f] [-j] [-l querytarget] [-e] [-s]
      [-uusername] [-Ggroupid]
```


Description

The qbf command invokes QBF, a forms-based interface for manipulating data in a database.

If you specify a query target on the command line, you must own all the tables that underlie the query target or have the proper permissions to access them. If you specify a JoinDef for the query target, you or the database administrator must own it.

The following table lists the valid command parameters and flags:

Parameter	Description
<i>dbname</i>	The name of the database. The <i>vnode</i> and <i>server_class</i> are specified if required; for details, see the Standard Command Line Flags and Parameters section.
-m <i>mode</i>	Bypasses the Join Definition phase of QBF, putting you directly into the <i>mode</i> function for Query Execution, where <i>mode</i> is retrieve, append, update or all. If you use the -m flag, you must also specify a <i>querytarget</i> .
-t	Indicates that <i>querytarget</i> is a table. A table field format will be used to query the table. This is the default type.
-f	Indicates that <i>querytarget</i> is a QB FName. This invokes QBF with a Visual-Forms-Editor (VIFRED) form.
-j	Indicates that the <i>querytarget</i> is a JoinDef.
-l	Allows QBF to locate the <i>querytarget</i> type. QBF looks first for a QB FName, then a JoinDef, and finally a table until it finds the <i>querytarget</i> specified.

Parameter	Description
<i>querytarget</i>	<p>Identifies the table, view, synonym, QBFName, or JoinDef you want to access in your query. Specifying <i>querytarget</i> will bring you directly into the Query Execution phase. If you specify it without also specifying -m<i>mode</i>, you will have the option of switching to the Join Definition phase.</p> <p>The table, view, or synonym name may be qualified with a valid schema name in the format <i>schema.name</i>. See the Using Schemas for Owner Qualification section.</p> <p>You can specify the type of <i>querytarget</i> to QBF by using the -t, -f, -j, or -l flag. If no flag is specified for <i>querytarget</i>, QBF assumes that the type is table and generates an error if it cannot find a table with that name.</p>
-e	Flag that invokes the command in expert mode, causing the catalogs to be displayed empty initially. This allows you to enter the name of a specific object directly, rather than select it from a list.
-s	Suppresses status messages.
-u <i>username</i>	Specifies the effective user for the session. For details, see the Standard Command Line Flags and Parameters section.
-G <i>groupid</i>	Specifies a group identifier. For details, see the Standard Command Line Flags and Parameters section.
	You must enclose this parameter in double quotation marks (" -G <i>groupid</i> ").

See the *Using Character-Based Querying and Reporting Tools* for a complete description of QBF.

query

Invokes the Query Execution phase of Ingres/Query-By-Forms (QBF).

Syntax

```
query dbname | vnode::dbname[/server_class]
      [-mmode] [-t|-f|-j] querytarget [-e]
      [-uusername] [-Ggroupid]
```

Description

The query command invokes the Query Execution phase of QBF, a forms-based interface for manipulating data in a database. Through the Query Execution phase you can append, retrieve or modify data. See *Using Character-Based Querying and Reporting Tools* for a complete description of QBF Query Execution.


The flags and parameter names have the same meaning as those for the qbf command, except that here *querytarget* is required. If you do not include *querytarget*, you are prompted for it. If the *querytarget* type is not specified with the -t, -f or -j flag, query uses the same order for looking up *querytarget* as the flag -l in the qbf command: first QBFName, JoinDef, and then table.


reconcil

Assists in recovering lost data as a means of disaster recovery.

Syntax

Windows**VMS****UNIX**

```
reconcil [vnodename::] dbname target_db_number  
cdds_no | "(x,y,z,...)" | all "start_time" [-udba_name] 
```

```
reconcil [vnodename::] dbname target_db_number  
cdds_no | '(x,y,z,...)' | all 'start_time' [-udba_name] 
```

The following table lists valid command parameters and flags:

Parameter	Description
<i>target_db_number</i>	Specifies the Ingres/Replicator database number of the failed database. It must represent a valid database name and Ingres/Net virtual node name.
<i>dba_name</i>	Name of the DBA who owns the replicated database specified in <i>dbname</i> .

Parameter	Description
<i>cdds_no</i> "(x,y,z,...)" all	<p>Specifies the CDDSs that are to be transmitted to the failed database to bring it back in synch with its replica.</p> <p>To specify the CDDS, use one of the following formats:</p> <ul style="list-style-type: none"> • <i>cdds_no</i> to send a single CDDS number • '(x,y,z,...)' to send a set of CDDS numbers • all to send all CDDS numbers <p>Note: If you specify more than one CDDS number, provide single quotes for UNIX and double quotes for OpenVMS and Windows.</p>
"start_time"	<p>Specifies the start time in Ingres date and time format used for recovering the lost data. Provide single quotes around the date and time for UNIX and double quotes for OpenVMS and Windows NT.</p> <p>To ensure that the start time covers the duration of the information gap, be sure to specify a start time prior to the database failure. It is better to have overlapping data that can be reconciled than risk an information gap in the target database.</p>
[vnode::]dbname	<p>Name of the replicated database that is to provide the lost data to the failed database.</p> <p>This replicated database must have a Replicator server configured to transmit the lost data to the failed database.</p>

Before you access the reconcil command, be sure you have:

- Quieted all Replicator servers in the environment by excluding users from replicated databases.
- Ensured that all the entries in the input queue have been moved to the distribution queue.

Description

The reconcil command is used along with standard DBMS recovery methods to recover lost data from journals, the Transaction Log File, or dump areas. The cause of lost data is usually irrecoverable disk failure.

The reconcil command works only on a replicated database that is restored to a consistent state using one of the standard recovery methods, such as checkpointing, journaling, and operating system backup.

Caution! *Do not use the reconcil command as your only means of disaster recovery. It is intended for use only with other standard disaster recovery tools and only if the conventional methods, alone, are unable to recover the data.*

In the event of a system failure, it is standard procedure to restore the affected database from checkpoints; however, if journals, log files, or dump areas are lost, a gap of missing data will exist on the failed database. This gap of data may still exist on one of the database replicas. The reconcil command can recover this gap using the shadow and archive tables on the affected database, provided those records still exist on these tables and reflect the data that was lost for the duration of the gap.

Note: Since the arcclean command purges records from the archive and shadow tables, you *cannot* use the reconcil command to recover lost data if you have executed arcclean on all of the replicated databases for the time of the information gap.

The reconcil command looks at each shadow table in the replicated database from a user-specified start time. If a record belongs to a CDDS that is common to the failed database, the command places an operation (insert, update, delete) for that record in the distribution queue, provided the operation does not already exist in the queue. Set the collision mode of the CDDS to BenignResolution and start the necessary servers to allow the lost data to be retransmitted to the failed database.

Example

The steps in the following scenario provide an example of how you can perform disaster recovery with the reconcil command:

1. A system failure occurring between 10:25 a.m. and 10:30 a.m. on September 20 destroys a disk on database lon::europe. A disk containing the transaction log file is also destroyed.

As a result, there is an estimated five-minute gap in committed transactions that were in the log file after the journals were re-run.
2. The DBA recovers the database from checkpoint which brings the database lon::europe to consistency as of 10:25 a.m.


3. In order to recover lost data in the database from the transaction log file, the DBA selects two databases to use with the `reconcil` command, `nyc::hq` and `hkg::asia`, both of which are full-peer replicas of the original `lon::europe` database.

The database `lon::europe` shares CDDS numbers 0 and 1 with `nyc::hq` and CDDS number 2 with `hkg::asia`.

4. The DBA removes databases `nyc::hq` and `hkg::asia` from user service and quiets their Replicator servers.
5. In both databases, the DBA ensures that all the entries in the input queue have been moved to the distribution queue.
6. The DBA invokes the `reconcil` command on the `nyc::hq` and `hkg::asia` databases. For example, the DBA issues the following command respectively on the `nyc::hq` and `hkg::asia` UNIX machines:


 UNIX

```
reconcil nyc::hq 20 '(0,1)' '16-nov-98 10:20'
```

```
reconcil hkg::asia -uwong 20 2 '16-nov-98 10:20' 
```

The target database number for both these commands is 20 (the number for `lon::europe`). On the `nyc::hq` machine, the CDDS set specified is `'(0,1)'`, while on the `hkg::asia` machine, only CDDS number 2 is specified.

Note: Since data was lost between 10:25 and 10:30, the DBA starts the `reconcil` command at 10:20, providing an overlap of at least five minutes to ensure the gap of missing data is recovered.

7. The DBA configures CDDSs 0, 1, and 2 with collision mode `BenignResolution`.
8. The DBA starts the Replicator servers to bring the database back in synch.

relocatedb

Moves the journal, dump, checkpoint, or default work location for a database to another location. Relocates or makes a copy of an entire database.

Syntax

```
relocatedb dbname[/server_class]  
  -new_ckpt_location=locationname |  
  -new_dump_location=locationname |  
  -new_jnl_location=locationname |  
  -new_work_location=locationname |  
  -new_database=newdbname  
  [-location=locationname{, locationname}  
  -new_location=locationname {, locationname}]
```

Description

The `relocatedb` command performs relocation operations. It can move the journal, dump, checkpoint, or default work locations. This is done, for example, when a disk fills or is swapped out.

The `relocatedb` command can also make a copy of an entire database. Any location in the original database can be moved to a new location in the new database.

The following table lists valid command parameters and flags:

Parameter	Description
<i>dbname</i>	The name of the database whose file(s) are to be moved. The <i>server_class</i> is specified if required; for details, see the Standard Command Line Flags and Parameters section.
-new_ckpt_location=	Flag preceding the name of the new checkpoint location. The location must be defined with checkpoint usage. Relocating a checkpoint location requires that checkpoints be disabled.
-new_dump_location=	Flag preceding the name of the new dump location. The location must be defined with dump usage. Relocating a dump location requires that checkpoints be disabled.

Parameter	Description
-new_jnl_location=	Flag preceding the name of the new journal location. The location must be defined with journal usage. Relocating a journal location requires that journaling and checkpoints be disabled.
-new_work_location=	Flag preceding the name of the new work location. The database must have been previously extended to this location for work usage. An exclusive lock is required while the relocation is done.
-new_database=	Flag preceding the name of the new database to be created. An exclusive lock on the original database is required while the original database is copied to the new database.
<i>newdbname</i>	The name of the new database to be created. The new database name must not exist.
-location=	<p>Flag preceding a list of data locations. All locations in the location list that follows this flag must be valid locations for the database. When -new_location is also specified, the data in each area in the location list is moved to the corresponding area in the new location list.</p> <p>This option is valid only for database relocation (-new_database= option).</p>
-new_location=	<p>Flag preceding a list of new data locations. All locations in the new location list that follows this flag must be defined for the installation, and the usage must be compatible with the usage of the corresponding area in the location list. Each location in the location list must be mapped to a distinct location in the new location list.</p> <p>This option is valid only for database relocation (-new_database= option).</p>
<i>locationname</i>	The location name for the location involved in the relocation.

Examples

To relocate the checkpoint location for the empdata database to newckp, type the following command:

```
relocatedb empdata -new_ckp_location=newckp
```

The foregoing command does the following:

- Performs an update to the iidbdb: update iidatabase.ckpdev='newckp' where name='empdata' (This can be verified by connecting to the iidbdb database and doing a select from iidatabase where name='empdata'.)
- Updates the checkpoint location in the configuration file. (This can be verified by examining the output of the infodb empdata command.)
- Copies checkpoint files from the old checkpoint location to the new checkpoint location.
- Deletes checkpoint files from the old checkpoint location.

To relocate the journal location for the empdata database to newjnl, type the following command:

relocatedb empdata -new_jnl_location=newjnl

The command above does the following:

- Performs an update to the iidbdb: update iidatabase.jnldev='newjnl' where name='empdata'. (This can be verified by connecting to the iidbdb database and doing a select from iidatabase where name='empdata'.)
- Updates the journal location in the configuration file. (This can be verified by examining the output of the infodb empdata command.)
- Copies journal files from the old journal location to the new journal location.
- Deletes journal files from the old journal location.

To relocate the dump location for the empdata database to newdump, type the following command:

relocatedb empdata -new_dump_location=newdump

The command above does the following:

- Performs an update to the iidbdb: update iidatabase.dmpdev='newdump' where name='empdata' (This can be verified by connecting to the iidbdb database and doing a select from iidatabase where name='empdata'.)
- Updates the dump location in the configuration file. (This can be verified by examining the output of the infodb empdata command.)
- Copies dump files from the old dump location to the new dump location.
- Deletes dump files from the old dump location.

To relocate the work location for the empdata database to newwork, type the following command:

relocatedb empdata -new_work_location=newwork

The command above does the following:

- Performs an update to the iidbdb: update iidatabase.sortdev='newwork' where name='empdata' (This can be verified by connecting to the iidbdb database and doing a select from iidatabase where name='empdata'.)
- Updates the default work location in the configuration file. (This can be verified by examining the output of the infodb empdata command.)
- No files are copied for work locations.

To copy the empdata database to a new database called empdev, type the following command:

relocatedb empdata -new_database=empdev

The command above does the following:

- Inserts a record in the iidatabase table for this database.
- Inserts a record in the iiextend table for all locations to which the new database is extended.
- Creates all the directories and copies all the files from the old database to the new database.
- Builds a database configuration file for the new database.

After the new database is created, you should be able to connect to it and access all data.

repcat

Creates and loads Ingres/Replicator catalog tables.

Syntax

repcat [+w | -w] [-udba_name] [vnode::]dbname

Parameter	Description
+w -w	Wait or don't wait for an exclusive lock on the database (-w is the default).
-udba_name	Specifies the effective user for the session. You must run repcat as the owner of the database. This flag is optional.

Parameter	Description
<i>dbname</i>	The name of the database. The <i>vnode</i> is specified if required. For details, see the Standard Command Line Flags and Parameters section.

Description

The `repcat` command creates and populates the Ingres/Replicator catalogs and creates Ingres/Replicator database events. You must run `repcat` before starting the Replicator Manager for the first time.

The `repcat` command must be run on every Replicator database containing Full Peer or Protected Read-only CDDs. You do not need to run `repcat` on databases with only Unprotected Read-only CDDs. The `repcat` command must be run before you can move the Ingres/Replicator configuration from a configuration database to the other participating databases.

Note: You can run `repcat` from a single location by specifying the virtual node name (*vnode*).

Examples

```
repcat europe
```

```
repcat -urep_dba nyc::hq
```

repcfg

Allows configuration of Replicator installations from the command line as an alternative to using Replicator Manager or Visual DBA.

Syntax

```
repcfg dbname cdds activate [-username] cdds_no [cdds_no...]
```

```
repcfg dbname cdds deactivate [-username] cdds_no [cdds_no...]
```

```
repcfg dbname table createkeys [-username] [-q] table_no [table_no...]
```


The following table lists valid command parameters:

Parameter	Description
<i>dbname</i>	The name of a database.
<i>obj_type</i>	Can be cdds or table. These arguments can be abbreviated to the initial character and are case-sensitive.
<i>action</i>	Can be activate, deactivate, or createkeys. These arguments can be abbreviated to the initial character and are case-sensitive.
<i>object</i>	Can be a CDDS number or table number.

Examples

To activate CDDS 0 in the repdb database:

repcfg repdb cdds activate 0

To create replication keys for tables 3 and 4 in the europe database and populate the input queue:

repcfg europe table createkeys -q 3 4

repdbcfg

Configures multiple Ingres mobile databases simultaneously.

Syntax

repdbcfg [*vnode::dbname filename* [- *udba_name*]

where *vnode::dbname* is the name of the database to be configured and *filename* is the name of the input file.

The input file should have the following format:

user_name [*db_no*] [*dbname*] [*cdds_no*] [*target_type*]

The parameters of the input file are described in the following table:

Parameter	Description
<i>user_name</i>	The user name of the owner of the mobile database. The <i>user_name</i> must be unique and 32 characters or less.
[<i>db_no</i>]	The number of the mobile database. The optional <i>db_no</i> must be a number in the range of 1-32,767 that has not already been used in your Replicator configuration (it must not exist in the <i>dd_databases</i> table). If <i>db_no</i> is not provided on the first line in the file, repdbcfg uses 101 as a default or the next higher number that does not exist in <i>dd_databases</i> . If <i>db_no</i> is not specified on subsequent lines, the previous value incremented by one is used.
[<i>dbname</i>]	The name of the mobile database. The optional <i>dbname</i> should be a unique database name up to 32 characters long. If it is not provided, the <i>user_name</i> is used as the default database name. The node name for all mobile databases is "mobile."
[<i>cdds_no</i>]	The number of the CDDS for the mobile database. The optional <i>cdds_no</i> should be a number in the range of 1-32,767. It should already be defined in the <i>dd_cdds</i> table through the CDDS Detail screen. If it is not provided on the first line, repdbcfg uses 50 as the default. If it is not provided on subsequent lines, the previous value is used.
[<i>target_type</i>]	The CDDS target type of the mobile database. The possible values for <i>target_type</i> are: FP – Full Peer PR – Protected Read-only UR – Unprotected Read-only If <i>target_type</i> is not provided on the first line, the default is FP. If it is not provided on subsequent lines, the default is the previous value. For each <i>target_type</i> , repdbcfg creates a data propagation path, under the given or default CDDS, from the local database to the mobile database. If the <i>target_type</i> is FP, repdbcfg creates an additional path from the mobile database to the local database.

Note: In Full Peer situations, the paths created by repdbcfg will not be sufficient to permit all changes from each mobile database to be replicated to all other mobile databases.

Description

The `repdbcfg` command allows you to configure multiple Ingres mobile databases simultaneously from the operating system prompt of the host machine to which Ingres/Replicator on mobile Replicator will connect.

Examples

In the first example, `repdbcfg` is invoked against an input file containing 26 user names:

```
albert
barbara
charlie
...
zoe
```

The `repdbcfg` utility defines 26 new full peer mobile databases, numbered 101 through 126. Connection names are in the form “mobile::charlie.” Data propagation paths are added for CDDS 50 to and from the local database and each new mobile database. If the local database number is 5, the first four paths are 5-5-101, 101-101-5, 5-5-102, and 102-102-5.

In the second example, the `repdbcfg` uses the input file:

```
albert 201 albert 3 FP
barbara
charlie 301 charlie 4 PR
daniel
```

The `repdbcfg` utility defines four new mobile databases numbered 201, 202, 301, and 302. The first two are full peer and each has two data propagation paths for CDDS 3. The other two databases are protected read-only and each has one propagation path for CDDS 4.

repinst

A black rounded rectangle with the word "Windows" in white text.

Command used to create or remove one or more Ingres/Replicator services.

Syntax

repinst *num_servers*

repinst **remove**

Parameter	Description
<i>num_servers</i>	Number of Replicator servers to install.

Description

The `repinst` command creates or removes Replicator servers as Windows Services. The first format of the command creates *num_servers* services. Servers are numbered sequentially starting from 1. If some services have already been created, `repinst` only creates new services beyond the existing ones, but up to *num_servers*. The second format of the command removes all services.

Examples

To create three services, type the following command:

repinst 3

To create two more services, type the following command:

repinst 5

To remove all five services, type the following command:

repinst remove 

repmgr

Invokes the Replicator Manager used to configure Ingres/Replicator.

Syntax

repmgr [-*udba_name*] [*vnode::*]*dbname*

The following table describes the `repmgr` parameters:

Parameter	Description
-<i>udba_name</i>	Specifies the effective user for the session. You must run <code>repmgr</code> as the owner of the database. This flag is optional.

Parameter	Description
<code>[vnode::]dbname</code>	Specifies the database to connect to. By specifying the nodename of a remote database in Replicator Manager, the database administrator can administer the entire Ingres/Replicator network from the local machine.

Note: To run Replicator Manager, you must have the correct system privileges and use the correct DBA name. An Ingres user with security privilege can impersonate the DBA and modify Replicator Manager information.

Description

The Replicator Manager command lets the distributed DBA for Ingres/Replicator:

- Connect to a single local database for table registration, a task that is required to set up a table or database for replication
- Configure the replication scheme
- Move replication configuration information between databases
- Check for replication configuration errors
- Issue database events to determine server action
- Monitor the distributed replication in a local database
- Run Ingres/Replicator integrity reports

Example

Assume the following command is executed from a remote San Francisco computer:

```
repmgr -unyc_dba nyc::hq
```

This command allows Replicator Manager to be run in client-server mode from the local San Francisco computer to the hq database on the nyc node, and assumes that the San Francisco DBA is impersonating the New York DBA (nyc_dba).

repmod

Modifies the Ingres/Replicator system tables to predetermined storage structures.

Syntax

repmod [*vnodename*] *dbname* [- *udba_name*]

where [*vnodename*] *dbname* is the name of the database whose system tables are to be modified.

Description

The repmod command is used to modify Ingres/Replicator system tables in a replicated database to the most appropriate storage structure for accelerating query processing. You must run repmod on the whole database.

report

Runs a default report or a report created with the rbf or sreport command.

Syntax

report *dbname* [*vnodename*:/*server_class*]
[-**r** | -**m** [*style*]] *report_target* [(*variable=value* {*variable=value*})]
[-**f***outfile*] [-**o***printer*] [-**n***copies*]
[-**5**] [-**6**] [-**+b** | -**b**] [-**d**] [-**h**] [-**l***pagewidth*] [-**qmxquer**]
[-**+t** | -**t**] [-**v***pagelength*] [-**wmxwrap**]
[-**i***filename*] [-**s**] [-**u***username*] [-**G***groupid*]

Description


The report command creates a report set up by the rbf or sreport command, or creates a default report for a table in the database.

The following table lists valid command parameters and flags:

Parameter	Description
<i>dbname</i>	The name of the database containing the report data. Specify the <i>vnodename</i> and <i>server_class</i> if required; for details, see the Standard Command Line Flags and Parameters section.

Parameter	Description
-r	Indicates that a report is specified as the <i>report_target</i> . If the specified report is not found, an error message is returned.
-m[style]	<p>Indicates that a table is specified as the <i>report_target</i>. This instructs report to format a default report for the specified table.</p> <p>The optional <i>style</i> specifies the style of your report. Accepted values are wrap, tabular (same as column), block, labels, and indented. If you do not specify a style, report selects either tabular or block, depending on the width of your report. Tabular is used if all of the columns fit on one page; otherwise block is selected. The default report width is 132 characters.</p>
<i>report_target</i>	<p>The name of the object on which you wish to run the report. The <i>report_target</i> may be:</p> <p>An existing report, created using RBF or sreport.</p> <p>A table, view, or synonym in your database on which you want a default report formatted.</p> <p>The table, view, or synonym name may be qualified with a valid schema name in the format <i>schema.name</i>. See the Using Schemas for Owner Qualification section.</p> <p>You can specify the type of <i>report_target</i> to report by using the -r or -m flag. If neither flag is specified, report looks first for a report having the specified name. If a report is not found, but a table with the same name exists, report sets up a default report for that table.</p>
<i>variable=value</i>	The <i>variable</i> specifies the name of a parameter used in the report. The <i>value</i> specifies the value that is replaced for every occurrence of the corresponding variable name in the report specifications. If you want to specify a string or a date, <i>value</i> must be quoted. <i>Variable/value</i> combinations on the command line can be separated by blanks, tabs, or commas.
-foutputfile	Directs the formatted report to the <i>outputfile</i> . If this option is not specified, the report is written to the standard output file (normally your terminal), or, in the case of a report specified by the Report-Writer, to the file designated by the .output command in the report specification file.

Parameter	Description
-o <i>printer</i>	Sends the report to the specified printer. To set a default printer, define <code>ING_PRINT</code> . If you require special print options, specify the options in <code>ING_PRINT</code> , and specify the <code>-o</code> flag with no argument.
-n <i>copies</i>	Specifies the number of copies of the report to print.
-5	Forces version 5 compatibility mode, as follows: The <code>+t</code> option is the default for aggregates. All arithmetic is floating point, unless all values in the computation are integers. By default, the month portion of the <code>current_date()</code> function is displayed in capital letters.
-6	(SQL reports only) Eliminates duplicate rows from reports whose specification contains <code>.data</code> , <code>.table</code> , <code>.view</code> , or <code>.sort</code> statements.
-b +b	<code>+b</code> forces form feeds at the end of each page. <code>-b</code> suppresses form feeds for the end of each page. The flag overrides any <code>.formfeed</code> or <code>.noformfeed</code> commands in the report specification file.
-d	Directs report to continue running the specified report if the <code>.setup</code> or <code>.cleanup</code> statements generate DBMS errors.
-h	Provides a null set of data for a report that retrieves no rows. All <code>.header</code> and <code>.footer</code> sections are executed. The detail section is suppressed. This feature allows you to include the following <code>.if</code> statement in the report footer to indicate that no rows were found: <pre> if count(column) = 0 .then .print "No data matched the query specifications." .endif </pre>
-l <i>pagewidth</i>	Sets the maximum output line size to <i>pagewidth</i> characters. By default, if output is to a file, the maximum output line size is 132 characters; otherwise, the default maximum line size is the width of the terminal.
-q <i>mxquery</i>	Sets the maximum length of the query after all substitutions for runtime parameters have been made to <i>mxquery</i> characters. By default, the maximum query size is 2048 characters. This option is needed only for particularly long queries.

Parameter	Description
-t +t	<p>If enabled (+t), causes aggregates and breaks to occur over rounded values for any floating point column whose format has been specified in a .format command as numeric F or template. Each value in the column is rounded to the precision given by its format. Additionally, breaks for date columns that use a date template occur over the actual value appearing for the dates.</p> <p>If disabled (-t), aggregates and breaks use the underlying values, <i>not</i> the rounded values. -t is the default.</p>
-vpagelength	<p>Sets the number of lines for each page of output. <i>pagelength</i> must be a positive integer. This flag overrides any .pagelength command in the report specification file. The default is 61 lines per page if the report is written to a file, and 23 lines per page if written to a terminal.</p>
-wmxwrap	<p>Sets <i>mxwrap</i> as the maximum number of lines to wrap with one of the column C formats, or the maximum number of lines that can be used within any block. By default, the maximum value is 300 lines. This maximum is provided as a protection against misspecified columns, and is rarely needed.</p>
-ifilename	<p>Reads a report specification from the specified file outside of the database, and runs the report. This eliminates the need to use the sreport command to place the report source file within the database for processing.</p> <p>You must omit <i>report_target</i> and the -r -m flag if you use the -I flag to specify a command file.</p>
-s	Suppresses status messages.
-uusername	Specifies the effective user for the session. For details, see the Standard Command Line Flags and Parameters section.
-Ggroupid	<p>Specifies a group identifier. For details, see the Standard Command Line Flags and Parameters section.</p> <p>You must enclose this parameter in double quotation marks ("-Ggroupid").</p>
	

See *Using Character-Based Querying and Reporting Tools* for a complete description of Ingres reporting tools and use of the report command.

repstat

Provides statistics about Ingres/Replicator transactions.

Syntax

repstat

Description

The repstat command provides statistics about Ingres/Replicator transactions, which include the Mutex address, queue size in entries, the start and end of queue at entry number, and the number of entries currently in the queue.

rollforwarddb

Recovers a database or table from the last checkpoint and the current journal and dump files. Can be used to relocate as well as recover tables.

Syntax

rollforwarddb *dbname[/server_class]*
[+c | -c] [+j | -j] [#m[n]] [-mdevice {, device}]
[-bdd-*mmm-yyyy:hh:mm:ss*] [-edd-*mmm-yyyy:hh:mm:ss*]
[#c[n]] [+w | -w] [-v] [-#f] [-uusername] [-help] [-statistics]
[-table=*tablename* {, *tablename*}]
[-nosecondary_index] [-on_error_continue]
[-relocate -location=*locationname* {, *locationname*}]
[-new_location=*locationname* {, *locationname*}]]
[-dmf_cache_size= *x*]
[-dmf_cache_size_4k | 8k | 16k | 32k | 64k= *x*]

Description

The rollforwarddb command recovers the specified databases. If a table list is given, only those tables will be recovered. When executing table level recovery, you may optionally move the table to a new location with the -relocate option.

If the target checkpoint was performed online (while the database was in use), then rollforwarddb does the following:

1. Restores the database from the checkpoint location to the database location.

2. Applies the log records in the dump location to the database, which returns the database to its state when the checkpoint began.
3. Applies the journal records to the database.

If the target checkpoint was executed offline, then the second step is omitted.



To execute the rollforwarddb command you must be:

- The DBA for the named database, or
 - An Ingres system administrator running rollforwarddb with the `-u` flag
- If you are using this command against a database in a group level installation, you must have the VMS CMKRNL privilege to run the command. ■

VMS

The following table lists valid command parameters and flags:

Parameter	Description
<i>dbname</i>	The database to be recovered. One database name must be specified. The <i>server_class</i> is specified if required; for details, see the Standard Command Line Flags and Parameters section.
+c -c	Recover/do not recover the database from the checkpoint file. The default is +c .
+j -j	Recover/do not recover the database from the journal. The default is +j .
-mdevice {, <i>device</i> }	Recovers the checkpoint from the specified tape device. If a list of tape devices is supplied, parallel recovery will be used for a multi-location database.
-bdd-<i>mmm-yyyy</i> [<i>:hh:mm:ss</i>]	Recovers only transactions that were completed after the specified date and time. <i>Before using this option, read Caution notice below.</i>
-edd-<i>mmm-yyyy</i> [<i>:hh:mm:ss</i>]	Recovers only transactions that were completed before the specified date and time. <i>Before using this option, read Caution notice below.</i>
#c [<i>n</i>]	Recovers from an older checkpoint. The checkpoint number <i>n</i> must be a valid checkpoint number (as shown by the infodb command). This flag can be used to recover the database when the current checkpoint is unfinished. If <i>n</i> is omitted, the most recent usable finished checkpoint is used for the recovery.

Parameter	Description
+w -w	Wait/don't wait for the database to be free (not in use). The default is -w. This flag is described further below.
 VMS	This flag can be used only in interactive sessions and not in batch mode. 
#m[n]	For a multi-location database, recover <i>n</i> locations at a time from disk.
-v	Recovers the database from the journal in verbose mode, which provides diagnostic information about all operations executed during the recovery process.
-#f	If rollforwarddb with journaling is attempted on a database that has had journaling disabled, it will fail unless the force flag #f is specified.
-uusername	Specifies the effective user for the session. For details, see the Standard Command Line Flags and Parameters section.
-help	Displays syntax online.
-statistics	Flag to print statistics about the rollforward.
-table=tablename {, tablename}	<p>A list of tables to be recovered from the target checkpoint. If multiple tables are specified, no space is allowed between the tables listed. Table recovery is not allowed for views, system catalogs, or Enterprise Access tables.</p> <p>If recovering a base table:</p> <p>Blob columns (long byte and long varchar columns) will also be recovered.</p> <p>Secondary indexes will also be recovered, unless -nosecondary_index is specified.</p>
-nosecondary_index	<p>Flag to inhibit automatic recovery of secondary indexes.</p> <p>Note: All secondary indexes will be marked inconsistent. The base table cannot be accessed until the secondary indexes are rebuilt or dropped.</p> <p>This option is invalid for database level recovery.</p>

Parameter	Description
-on_error_continue	<p>Flag to continue processing if possible even if an error occurs. If any error occurs processing a table, the table is removed from the table list and processing continues. If rollforwarddb processing has already started for the table when the error occurs, the table is marked inconsistent and all further operations on this table are ignored.</p> <p>If this option is not specified and an error is encountered, all tables being recovered are marked inconsistent and rollforwarddb terminates.</p> <p>Note that this option will not force continuation of an invalid rollforwarddb command. Rollforwarddb is terminated immediately if an invalid table is specified in the table list, for example, if a view, system catalog, or Enterprise Access table, a nonexistent table; or a table for which recovery is disallowed is specified.</p> <p>This option is invalid for database level recovery.</p>
-relocate	<p>Flag to indicate that a table is to be relocated to a new location during recovery. When -relocate is specified -location and -new_location must also be specified.</p> <p>This option is invalid for database level recovery.</p>
-location= <i>locationname</i> {, <i>locationname</i> }	<p>Flag preceding a list of data locations. When -relocate and -new_location are also specified, the data in each area in the location list is moved to the corresponding area in the new location list. Only tables being recovered are relocated.</p> <p>This option is invalid for database level recovery.</p>
-new_location= <i>locationname</i> {, <i>locationname</i> }	<p>Flag preceding a list of new data locations. When -relocate and -location are also specified, the data in each area in the location list is moved to the corresponding area in the new location list. Only tables being recovered are relocated.</p> <p>When this option is specified, -relocate and -location must also be specified, and the number of locations in the location list must equal the number of locations in the new location list. (The number of location names associated with a table cannot be changed using rollforwarddb.)</p> <p>This option is invalid for database level recovery.</p>
<i>locationname</i>	The location name for the location involved in the relocation.

Parameter	Description
<code>[-dmf_cache_size= x]</code> <code>[-dmf_cache_size_4k 8k 16k 32k 64k= x]</code>	Specifies the size of the local cache that rollforwarddb allocates. Default values are: 256 for <code>-dmf_cache_size= x</code> which specifies number of 2 KB buffers. 200 for 4 KB, 8 KB, 16 KB, 32 KB, and 64 KB which specifies the number of respective buffers. For example: <code>-dmf_cache_size_64k=200</code> would indicate the 200 64 KB buffers. However, if you specify 0 for the 4 KB, 8 KB, 16 KB, 32 KB, or 64 KB buffers, 256 buffers will be allocated.

If the database was checkpointed to a tape, you can use the `-m` flag to restore the database from the tape.

Before you execute rollforward from a tape device, the tape must be inserted into the tape drive. 📌

VMS

Caution! The `-b` and `-e` options of rollforwarddb should be used with caution. Recovery enhancements in the Ingres release do not guarantee attempts to skip recovery of a segment of the journal file with these parameters, and using `-b/-e` in this manner is not supported. Table level recovery using the `-e` option will result in the table being logically inconsistent.

The `+w` | `-w` flag directs rollforwarddb to wait (`+w`) or not wait (`-w`) for the database to be free before recovering the database. Since rollforwarddb requires the database to be locked, this flag allows you to decide whether or not to wait for the database to be free if it is in use. If you specify `+w`, rollforwarddb will wait as long as necessary for the database to become free for locking and recovery. If you specify `-w`, an error is returned if the database is busy. The default is `-w`.

By default, rollforwarddb will sequentially restore data locations one at a time. A database with more than one data location can be restored in parallel.

For detailed procedures on performing backup and recovery of the database, see the *Database Administrator's Guide*.

Examples

The following command allows you to recover the empdata database from the target checkpoint and journal. This assumes that both the journal and the checkpoint are currently online. If not, they should be placed online before executing these commands. To proceed, type the following command:

rollforwarddb empdata -v

To recover tables emp and emphist from the empdata database, type the following command:

rollforwarddb empdata -table=emp,emphist

To recover tables emp and emphist from the empdata database without recovering the indexes, type the following command:

**rollforwarddb empdata -table=emp,emphist
-nosecondary_index**

Note: The indexes on tables emp and emphist will have to be rebuilt or dropped before the tables can be accessed.

To recover table emp in the empdata database and also relocate it from location emploc to the new location newemploc, type the following command:

**rollforwarddb empdata -table=emp,emphist
-relocate -location=emploc -new_location=newemploc
rollforwarddb empdata +c +j -m/dev/rmt0** ■

UNIX

VMS

rollforwarddb empdata +c +j -mMTA0: ■

rpserver

Command used to start up each Replicator server individually from the operating system prompt.

Syntax

rpserver *n*

where *n* is the number of the server you want to start.

Description

The `rpserver` command is used to start a Replicator server from the command line. It takes only a single parameter, the server number. All other parameters are read in from the `runrepl.opt` file, which should be present in the corresponding server directory.

rsstatd

Provides Ingres/Replicator server statistics.

Syntax

rsstatd

sql

Invokes the line mode Ingres/Terminal Monitor.

Syntax



sql [*SQL option flags*] [*line-mode flags*]
 dbname [*vnode::dbname/server_class*]
 [*<altin*] [*>altout*]

Description

This command invokes the line mode Ingres/Terminal Monitor. For procedures on using this Terminal Monitor, see the *SQL Reference Guide*.

The following table lists valid command parameters and flags:

Parameter	Description
<i>SQL option flags</i>	Flags that can be used with the line mode Terminal Monitor and other commands where noted. These flags are described below.
<i>line-mode flags</i>	Flags that can be used only with the line mode Terminal Monitor. These flags are described below.

Parameter	Description
<i>dbname</i>	The name of the database. The <i>vnode</i> and <i>server_class</i> are specified if required; for details, see the Standard Command Line Flags and Parameters section.
< <i>altin</i>	Specifies a file from which the Terminal Monitor reads commands. The file must contain all the terminal monitor commands needed to run the session.
> <i>altout</i>	Directs output from the Terminal Monitor to the specified file. If you specify this parameter, you will not see any output.
	No space must come between the angle brackets (< or >) and the filename. 

Line-mode Flags

The following table lists the *line-mode flags* that can be used only with the `sql` command:







Flag	Description
+a -a	Set/clear the autoclear option in the terminal monitor. The default is +a.
+d -d	Print/don't print the dayfile. The default is +d.
+s -s	Print/don't print any of the monitor messages, including prompts. The default is +s; if you specify -s, the dayfile is not displayed.
-vX	Set the column separator to the character specified by X. The default is vertical bar ().



SQL Option Flags

SQL option flags are accepted by the `sql` command and other commands where noted as the parameter [*SQL option flags*]. You can specify a maximum of 12 SQL option flags.

The SQL option flags are presented in the table below. The first four flags (-c, -f, -i, and -t) specify the format of output. The remaining SQL option flags affect the behavior of the DBMS.

Flag	Description
-cN	Set the minimum field width for printing character columns to <i>N</i> . The default is 6.

Flag	Description
-f <i>kxM.N</i>	<p>Set floating point output column width to <i>M</i> characters (total), including <i>N</i> decimal places, and (if warranted) <i>e</i>+<i>-xx</i> and the decimal indicator character itself. <i>k</i> may be 4 or 8 to apply to f4's or f8's respectively. <i>x</i> may be E, F, G or N (uppercase or lowercase) to specify an output format. E indicates exponential format. F or N indicates the floating point format. G indicates the floating point format and guarantees decimal alignment.</p> <p>If you specify F, N, or G and the number is too large for the format indicated by the flag, it is displayed in exponential format. To prevent this format overflow, <i>M</i> should be greater than or equal to <i>N</i> + 7.</p> <p>The default display format for both f4 and f8 is n10.3, unless your computer supports the IEEE standard for floating point numbers, in which case the display format for f4 and f8 is n11.3.</p>
-i <i>kN</i>	Set integer output column width to <i>N</i> . <i>k</i> may be 1, 2, or 4 for i1's, i2's, or i4's, respectively. The default for <i>N</i> is 6 for i1 and i2 fields, and 13 for i4 fields.
-t <i>N</i>	Set the minimum field width for printing text columns to <i>N</i> . The default is 6.
+U -U	<p>Enables/disables user updating of the system catalogs and secondary indexes. This flag takes an exclusive lock on the database.</p> <p>To update system catalogs you must have the update system tables privilege obtained through accessdb.</p>
	You must enclose this flag in double quotation marks ("U" or "-U"). 
+Y -Y	Same as "+/-U" flag, except an exclusive lock on the database.
	You must enclose this flag in double quotation marks ("Y" or "-Y"). 
-u <i>username</i>	Specifies the effective user for the session. For details, see the Standard Command Line Flags and Parameters section.
-G <i>groupid</i>	Specifies a group identifier. For details, see the Standard Command Line Flags and Parameters section. You must enclose this parameter in double quotation marks ("-G <i>groupid</i> "). 
	

Flag	Description
-Rroleid	Specifies a role identifier for the session. For details, see the Standard Command Line Flags and Parameters section.
	You must enclose this parameter in double quotation marks ("-Rroleid").
-l	Locks the database for your exclusive use. When you specify this flag, no one else can open the database while you are in it. If you attempt to take an exclusive lock on a database that is in use, the system informs you that the database is temporarily unavailable.
-nM	Sets modify mode on the index command to M. M must be one of the following storage structures: ISAM, CISAM, B-tree, CB-tree, Hash, or CHash. The default is ISAM.
+w -w	Specifies wait/don't wait for the database. The default is -w. If you specify +w, there is a wait, provided that certain processes are running (sql -l, sql -U, verifydb, rollforwarddb or sysmod) on the given database. Upon completion of those processes, the operation proceeds. If you specify -w and the database is not available, a message is returned and execution is stopped. If you omit the w flag and the database is unavailable, then an error message is returned if running in foreground (more precisely, if the standard input is from a terminal). Otherwise the wait option is invoked.
	This flag is not valid in batch mode.
-numeric_ overflow = fail ignore warn	<p>Sets error handling mode for numeric overflow, underflow and division by zero.</p> <p>The fail setting causes an error message to be issued and the statement is aborted. This is the default setting. To obtain ANSI-compliant behavior, use this setting (or omit for the default).</p> <p>The ignore setting causes no error message to be issued.</p> <p>The warn setting causes a warning message to be issued.</p>
-string_ truncation = fail ignore	<p>Sets error handling mode for string truncation errors. This error will occur if you attempt to insert a string into a table column that is too short to contain the value.</p> <p>The fail setting causes an error message to be issued and the statement is aborted.</p> <p>The ignore setting causes no error message to be issued. The string is truncated and inserted. This is the default setting.</p>

Examples

Open the empdata database:

```
sql empdata
```

Open empdata, suppressing the dayfile message:

```
sql empdata -d
```

Open empdata, suppressing the dayfile message and the terminal monitor prompts and messages; read into the workspace the contents of the batchfile:

```
sql empdata -s <batchfile
```

Open empdata, display f4 columns in G format with two decimal places and i1 columns with three spaces:

```
sql empdata -f4g12.2 -I13
```

starview

Invokes the forms-based program, StarView, which is used for managing distributed database(s) using Ingres/Star. StarView allows access to multiple databases simultaneously.

Syntax

```
starview [vnode::][distdbname]/star
```

You may also include a remote vnode name to run StarView against a distributed database on a remote node.

Note: If you invoke StarView *with* a database name, the Node Status and LDB Types frame is displayed. If you invoke StarView *without* specifying a database name, the opening StarView main frame is displayed.

Description

StarView is a simple forms-based program that helps you manage your distributed databases.

Using StarView you can:

- Display all your distributed databases

- Display the nodes, databases, and tables registered in a distributed database
- Display the local database objects that make up a distributed database
- Test the network connections to each node in a distributed database
- Register local tables and views in a distributed database without using the SQL register as link statement
- Remove database objects that you had previously registered in your distributed database

In addition, you can query databases from within StarView with direct access to:

- Ingres Interactive SQL (ISQL)
- The Tables command

The StarView program operates in exactly the same way as forms-based Ingres tools. See *Using Character-Based Querying and Reporting Tools* for a complete explanation of using forms.

Example

To manage distributed databases on the remote node `new_york`, type the following command:

```
starview new_york::mystar
```

statdump

Prints statistics contained in the `iistats` and `iihistograms` catalogs of the Standard Catalog Interface.

Syntax

```
statdump [-zf filename]
or
statdump [-zq] [-zdl] [-zn#]
[-zc] [-o filename]
[ingres flags] dbname
{-rtablename {-acolumnname}} | {-xtablename}
or
statdump [-help]
```

Description

The statdump command allows you to inspect the iistats and iihistograms catalogs in the Standard Catalog Interface. These views contain statistical information about columns used by the Ingres Query Optimizer as it selects an efficient query processing strategy. The statistical information is usually generated by issuing the optimizedb command.

The following table lists valid command parameters and flags:

Parameter	Description
-zf <i>filename</i>	Directs statdump to read <i>filename</i> for all other command line flags, database names, and any other command line arguments. This file must contain only one flag per line. (See the examples below.) If this flag is specified, no other flags or arguments can appear on the command line; they must, instead, appear in the specified file.
-zq	Displays only the information contained in the iistats catalog and not the histogram information contained in iihistograms (Quiet mode).
-zdl	Deletes statistics from the system catalogs. When this flag is included, the statistics for the specified tables and columns (if any are specified) are deleted rather than displayed.
-zn <i>#</i>	Displays floating point values in scientific notation (for example, 9.9999+e9) and sets the precision to the level specified by <i>#</i> . The total width of the displayed number will be equal to the value of the precision level + 7.
-zc	Displays statistics on the system catalogs as well as the base tables. If you want statistics for selected system catalogs, use this flag and specify the individual tables with the -r flag. You must be the DBA of the specified database to use this flag.
-o <i>filename</i>	Directs the output of statdump to the file specified by <i>filename</i> . The resulting file is an ASCII file whose content is identical to the information normally sent to the terminal screen. The resulting file can be used as input to optimizedb. See the -i flag description for the Optimizedb command.

Parameter	Description
<i>ingres flags</i>	<p>Pass any of these flags. Statdump accepts the following SQL option flags:</p> <p>[+ -]U -u -cN -tN -ikN -fkxM.N [+ -]w -xk</p> <p>For a complete description of these flags, see the <code>sql</code> command.</p>
<i>dbname</i>	<p>The name of the database.</p> <p>The <i>vnod</i>e and <i>server_class</i> are specified if required; for details, see the Standard Command Line Flags and Parameters section.</p>
-rtablename	<p>Produces statistics for the specified tables only. If omitted, then statistics for all tables are produced.</p> <p>The table name may be qualified with a valid schema name in the format <i>schema.tablename</i>. See the Using Schemas for Owner Qualification section.</p>
-rxtablename	<p>Specifies <i>tablename</i>s to be excluded from processing by statdump. Except for these, statistics for all columns in all tables in the database are produced.</p>
-acolumnname	<p>Produces statistics for the specified column(s) only. To specify individual columns you must specify the table name with the -r flag. If column names are omitted, then all columns of the specified table are processed.</p>
-help	<p>Displays syntax online.</p>

Note: If a specified table or column cannot be found, then a warning message is printed and processing continues.

Note: The combination of **-rtablename** and **-rxtablename** parameters is not permitted in a single statdump request, nor is the combination of **-rxtablename** and **-acolumnname**.

Examples

Print the statistical information for all columns in the employee table in the empdata database:

statdump empdata -remployee

For all columns in all tables of the empdata database, print out only the information in the iistats system table:

statdump -zq empdata

Delete statistics for all columns in the employee table:

statdump -zdl empdata -remployee

sysmod

Modifies the system tables to predetermined storage structures.

Syntax

```
sysmod dbname [/server_class]
        {tablename} [-f product {product}] [-page_size=n][+w |-w]
```



Description

The sysmod command modifies a database's system tables (catalogs) to the most appropriate storage structure for accelerating query processing. You can run sysmod on the whole database or on specified tables.

To use this command, you must be the DBA for the specified database or the Ingres system administrator.

The following table lists valid command parameters and flags:

Parameter	Description
<i>dbname</i>	The name of the database. The <i>server_class</i> is specified if required; for details, see the Standard Command Line Flags and Parameters section. Do not specify the <i>server_class</i> as /star if the database is a Star distributed database.

Parameter	Description
<i>tablename</i>	Specifies individual tables to be modified by sysmod. May be Star standard catalogs or Star-specific system catalogs. If omitted, all tables in the database are processed. The table name may be qualified with a valid schema name in the format <i>schema.tablename</i> . See the Using Schemas for Owner Qualification section.
-f <i>product</i>	Specifies the user interface product(s) for which you want to modify system tables. Allowable <i>product</i> names are ingres, ingres/dbd, vision, and windows_4gl. If you omit this parameter, all user interfaces are processed. For details, see the Standard Command Line Flags and Parameters section. Note: You cannot specify individual user interface catalogs; when you specify the <i>product</i> parameter, all catalogs are processed for that user interface product.
-page_size	Permits modifying the existing system catalogs with a different page size: Specify <i>page_size=n</i> , where <i>n</i> is one of 2048, 4096, 8192, 16384, 32768, 65536. Example: SYSMOD test -page_size=4096
+w -w	Directs sysmod to wait or not wait until the database is free before executing. Sysmod requires exclusive access to the database.
	This flag is not valid in batch mode. 

tables

This command starts the tables program.

Syntax


```
tables dbname | vnode::dbname[/server_class][-e] [-uusername]  
[-Ggroupid]
```

Description

The tables program invokes Tables, a forms-based interface for creating, destroying and examining tables.

See *Using Character-Based Querying and Reporting Tools* for a complete description of tables.

The following table lists valid command flags and parameters:

Parameter	Description
<i>dbname</i>	The name of the database. Note: The <i>vnode</i> and <i>server_class</i> are specified if required. For details, see the Standard Command Line Flags and Parameters section.
-e	Invokes Tables in empty mode. In essence, this flag causes any catalog of tables to be initially displayed empty, so that the user can enter specific names of such objects.
-uusername	Specifies the effective user for the session. For details, see the Standard Command Line Flags and Parameters section.
-Ggroupid	Specifies a group identifier. For details, see the Standard Command Line Flags and Parameters section.
	You must enclose this parameter in double quotation marks (" <i>-Ggroupid</i> ").

Example

Invoke the tables utility with an empty initial catalog of the emp database for the effective user emma:

```
tables emp -e -uemma
```

unloaddb

Creates command files for complete unloading and reloading of a database.

Syntax

```
unloaddb dbname [vnode::dbname/server_class]  
[-c] [-ddirname] [-source=dirname] [-dest=dirname]  
[-uusername] [-Ggroupid]
```

Description

The unloaddb command creates two command files that the DBA uses to unload the data from a database and reload the data into a new, empty database. The name of the two command files depend upon the operating system:

Windows

unload.bat—contains commands to read sequentially through the database, copying every user table into its own file in the named directory.

unload.ing—contains commands to read sequentially through the database, copying every user table into its own file in the named directory. ■

UNIX

reload.bat—contains commands to load a new, empty database with the information contained in the files created by unload.bat.



reload.ing—contains commands to load a new, empty database with the information contained in the files created by unload.ing. ■

The unloaddb command does not do the unloading or reloading of the database; the command files created by unloaddb must be executed by the DBA to accomplish these tasks. It is important that the database be recreated with reload.ing before doing any work (for example, creating tables, forms, and reports) in the new database.

The following table lists valid command parameters and flags:

Parameter	Description
<i>dbname</i>	The name of the database. The <i>vnode</i> and <i>server_class</i> are specified if required. For details, see the Standard Command Line Flags and Parameters section.
-c	Directs unloaddb to create printable data files. This is useful for transporting databases between computer systems whose internal representations of non-ASCII data differ. Unloaddb cannot create printable files if (1) binary data is stored in varchar columns, or (2) tables contain user-maintained logical keys.

Parameter	Description
-ddirname	<p>Stores the unload.ing and reload.ing in the location specified by dirname instead of the default current directory. The specification can be either a full or relative directory specification.</p> <p>The dirname must not be the actual database directory, because the files created by unloaddb may have the same names as the tables in the database. The actual database directory is:</p> <p>[II_DATABASE]\ingres\data\default\dbname</p> <div> <div>Windows</div> <div>UNIX</div> <div>VMS</div> </div> <p>Note: [II_DATABASE] must be replaced by the value returned by ingprenvII_DATABASE. ■</p> <p>\$II_DATABASE/ingres/data/default/dbname ■</p> <p>II_DATABASE:[INGRES.DATA.DBNAME] ■</p>
-source=dirname	<p>The source directory from which the database will be reloaded. An empty <i>dirname</i> specification ("") denotes the current directory. The -source specification overrides a -d specification for the reload file.</p> <p>If a source is specified without a destination (no -d or -dest) then the default unload directory is used.</p> <p>The source directory specification is not checked for validity or existence. This allows the scripts to be moved to another machine for reloading.</p>
-dest=dirname	<p>The destination directory into which the database will be unloaded. An empty <i>dirname</i> specification ("") denotes the current directory. The -dest specification overrides a -d specification for the unload file.</p> <p>If a destination is specified without a source (no -source) then the default reload directory is used.</p> <p>The destination directory specification is not checked for validity or existence. This allows the scripts to be moved to another machine for unloading.</p>
-username	<p>Specifies the effective user for the session. For details, see the Standard Command Line Flags and Parameters section.</p>

Parameter	Description
-Ggroupid	Specifies a group identifier. For details, see the Standard Command Line Flags and Parameters section.
	You must enclose this parameter in double quotation marks (" -Ggroupid "). 

The unloaddb command unloads all objects in the database. These include tables, views, integrity constraints, permissions, forms, graphs, and report definitions. Use unloaddb when a database must be totally rebuilt, or for checkpointing the database.


In order to optimize performance, run the sysmod and optimizedb commands after recreating the database.

The unloaddb command uses a version of the copydb command to generate the copy commands in the unload.ing and reload.ing files. Consequently, all limitations of the copydb command apply to the unloaddb command.


Examples

To unload and reload the empdata database:


Windows

```
cd\mydir\backup
unloaddb empdata
unload
destroydb empdata
createdb empdata
reload
sysmod empdata 
```

UNIX

```
cd /mydir/backup
unloaddb empdata
unload.ing
destroydb empdata
createdb empdata
reload.ing
sysmod empdata 
```

VMS

```
set default [mydir.backup]
unloaddb empdata
@unload.ing
destroydb empdata
createdb empdata
@reload.ing
sysmod empdata 
```

To unload the empdata database with separate source and destination directory specifications:

```
unloaddb empdata -source="misc/loaddir/"  
-dest="misc/dumpdir"
```

Copy statements in the reload script would have the form:

```
copy emps () from 'misc/loaddir/emp.bob'
```

Copy statements in the unload script would have the form:

```
copy emps () into 'misc/dumpdir/emp.bob'
```

Unload the empdata database from the \$HOME directory with source and destination directory specifications with no path:

```
unloaddb empdata -source="" -dest=""
```

Copy statements in the reload script would have the form:

```
copy emps () from 'emp.bob'
```

Copy statements in the unload script would have the form:

```
copy emps () into 'emp.bob'
```

upgradedb

Installs and upgrades databases.

Syntax

```
upgradedb dbname [vnode::dbname[/server_class]] [-all  
[-f product {product}]] [-help]
```

Description

Upgradedb is used to install and upgrade one or all databases in the Ingres installation.

Note: If databases are not upgraded at install time, rmcmd must be shut down (that is, set to zero in CBF) before running upgradedb. Upgradedb takes a lock on iidbdb, and it cannot get the lock if rmcmd is running.

The following table lists valid command parameters and flags:

Parameter	Description
<i>dbname</i>	<p>The name of a specific database to be upgraded.</p> <p>There can be only one database specified at a time.</p> <p>The <i>vnode</i> and <i>server_class</i> are specified if required; for details, see the Standard Command Line Flags and Parameters section.</p>
-all	<p>Causes upgradedb to operate on all databases in the installation that have not already been upgraded to the new release level. When you specify the -all flag, upgradedb skips any databases already at the current release level.</p> <p>You can specify either <i>dbname</i> or all, but not both.</p>
-f product	<p>Specifies the user interface product(s) for which you want to upgrade the database. Allowable <i>product</i> names are <i>ingres</i>, <i>ingres/dbd</i>, <i>vision</i>, <i>windows_4gl</i> and <i>nofeclients</i>. If you omit this parameter, all Ingres tools for the database are processed. For more details, see the Standard Command Line Flags and Parameters section.</p>
-help	Displays syntax online.

Note: It is usually advisable to use the same **-f** command to upgrade the database that was used to create it initially. If you are upgrading a pre-6.3 database, the **-fingres** command upgrades your existing user interface catalogs without creating the catalogs for the Ingres tool products that were new with 6.3 (Vision and Windows 4GL).

Upgradedb triggers *upgradefe*, which is described in more detail in the *upgradefe* command description.

If upgradedb cannot upgrade the user interface catalogs for a database, it prints a warning and marks the database operative. You can then either run *upgradefe* directly on the database, or rerun *upgradedb*, specifying the database individually with the *dbname* parameter.

upgradefe

Installs and upgrades Ingres tool catalog definitions.

Syntax

```
upgradefe dbname [vnode::dbname/server_class]  
             {product} [-b] [-vversion] [-s] [-uusername]
```

Description

The upgradefe command installs and upgrades the catalogs required by Ingres tools. You must execute the upgradefe command after installing a new version of any Ingres tool, if the new version requires changes to the catalog definitions.

The following table lists valid command parameters and flags:

Parameter	Description
<i>dbname</i>	The name of the database. The <i>vnode</i> and <i>server_class</i> are specified if required; for details, see the Standard Command Line Flags and Parameters section.
<i>product</i>	Specifies the Ingres tool products for which you want to upgrade the database. Allowable <i>product</i> names are ingres, ingres/dbd, vision, and windows_4gl. If you omit this parameter, all user interfaces for the database are processed. For more details, see the Standard Command Line Flags and Parameters section.
-b	Installs the modules required to support the product Ingres. (Specifying the -b flag is equivalent to specifying the product ingres.)
-v <i>version</i>	Specifies the version number of the product to be installed; if the -v flag is not specified, the highest known version is installed.
-s	Suppresses messages from upgradefe.

Parameter	Description
-u <i>username</i>	Specifies the effective user for the session. If you want to upgrade a database you do not own, you must use the -u flag to specify the user name of the DBA. For details, see the Standard Command Line Flags and Parameters section.

Examples

The following example installs catalogs for OpenROAD:

```
upgradefe mydb windows_4gl
```

The following example installs catalogs for the base tools and Vision:

```
upgradefe mydb ingres vision
```

usermod

Modifies the user-defined tables to predetermined storage structures.

Syntax

```
usermod dbname [vnode::dbname[/server_class]] [-uusername] [tables] [-noint]
```

Description

The usermod command modifies a database's user-defined tables to the most appropriate storage structure for accelerating query processing. Just like sysmod, which does a modify on system catalogs, this is a useful utility for maintaining tables on a regular basis. Running this utility on a regular basis or when the table has excess overflow pages, improves the performance of user applications.

You can run sysmod on the whole database or on specified tables. If no specific tables are specified, all the tables belonging to the user will be modified.

The following table lists valid command parameters and flags:

Parameter	Description
-----------	-------------

Parameter	Description
<i>dbname</i>	The name of the database. The <i>vnode</i> and <i>server_class</i> are specified if required; for details, see the Standard Command Line Flags and Parameters section. Do not specify the <i>server_class</i> as <i>/star</i> if the database is a Star distributed database.
-u <i>username</i>	Specifies the effective user for the session. For details, see the Standard Command Line Flags and Parameters section.
<i>tables</i>	Specifies individual tables to be modified by usermod, the table names should be separated by spaces. May be Star standard catalogs or Star-specific system catalogs. If omitted, all tables in the database are processed. The table name may be qualified with a valid schema name in the format <i>schema.tablename</i> . See the Using Schemas for Owner Qualification section.
-noint	The “-noint” flag specifies uninterrupted run, that is, usermod should be run for all the specified tables, even if there were errors.

verifydb

Destroys or lists any unrequired disk files, expired tables, or temporary tables in the specified databases, or removes references to a specified table from the DBMS system catalogs.

Syntax

```
verifydb -mmode -sscope -ooperation [-n | -lflogfilefilename]  
[-v] [-uusername]
```


Description

The verifydb command performs clean up operations on one or more databases in an installation. Using this command, you can delete all unrequired disk files in a database directory, delete temporary and/or expired tables, or remove all references to a specified table from the DBMS system catalogs.


This command requires exclusive access to databases. Verify that there are no active sessions in the DBMS before continuing. If users are connected to the database, a runtime error is displayed. Processes that maintain database connections, rmcmd and icesvr, should be shut down.

Verifydb logs all of its actions to the terminal screen. It also logs to a verify log file, unless the `-n` (nologging) flag is used. The default log file is `iivdb.log` and is used unless another name is specified with the `-lf` option. Note that verifydb always outputs the log file to the `II_CONFIG` location. If `II_CONFIG` is not defined, it outputs to location:

Windows

`%II_SYSTEM%\ingres\files` 

UNIX

`II_SYSTEM/ingres/files` 

VMS

`II_SYSTEM:[INGRES.FILES]` 

If the log file does not exist when you execute verifydb, it is created. If it does exist, verifydb appends to it. Since this file grows each time you execute verifydb with this log file, be sure to delete it occasionally in order to save disk space.

The following table lists valid command parameters and flags:

Parameter	Description
<code>-mmode</code>	<p>This mandatory flag specifies the mode in which verifydb executes. <i>Mode</i> can be any of the following:</p> <p><code>report</code>— directs verifydb to log its findings. Use the report mode if you want verifydb to only log, rather than actually delete, the tables and/or files that it finds.</p> <p><code>run</code>— directs verifydb to perform the specified operation and log all actions that it performs.</p> <p><code>runinteractive</code>— directs verifydb to prompt the user for confirmation before each action is taken. If the user responds negatively to a prompt, verifydb skips that action and goes on to the next.</p> <p><code>runsilent</code>— tells verifydb to perform the specified operations but turns off the logging to the terminal. (Logging to the log file continues.)</p>

Parameter	Description
-sscope	<p>This mandatory flag specifies the scope of the verifydb command. <i>Scope</i> can be any of the following:</p> <p>dbname "<i>dbname</i> [<i>vnode::dbname</i> [/server_class] {<i>dbname</i> [<i>vnode::dbname</i> [/server_class]]}"</p> <p>Tells verifydb to perform the operation only on the databases specified. All databases specified by this qualifier must have the same owner. You can specify up to 10 databases.</p> <p>dba—directs verifydb to operate on all databases for which the user is the DBA or for all databases owned by the DBA specified by the -u flag.</p> <p>installation—requires verifydb to perform the operation on all operative databases. You must be a privileged user to use this qualifier.</p>
-ooperation	<p>This mandatory flag specifies the operation to be performed. If the report mode is specified, the files or tables found are not actually deleted, but only logged. The options for <i>operation</i> are:</p> <p>accesscheck—checks each database specified by the scope and returns a message that says whether the server can connect to the database and, if not, provides a short message indicating why not. When you use this option, you must also specify report mode (-mreport).</p> <p>You must be either a DBA or a privileged user to use this option. If you are a DBA and use the dbname option for scope, you must be the DBA of all the listed databases. If you use the dba option, verifydb will check all the databases for which you are the DBA. To use the scope's installation option, you must be a privileged user. In that case, accesscheck will check all the databases in the installation.</p> <p>Additionally, if you are a privileged user, you can use the -u flag to run this option as another user.</p> <p>purge—directs verifydb to delete all disk files in the database directory that are no longer required. This operation is a combination of temp_purge and expired_purge.</p>

Parameter	Description
-ooperation (cont.)	<p>temp_purge – tells verifydb to search for and delete all temporary tables from the database.</p> <p>expired_purge – directs verifydb to search for and delete all expired tables from the database.</p> <p>drop_table “tablename” – tells verifydb to remove all references to a specified table from the DBMS system catalogs. If you specify this option, you must use the dbname option for the -s flag.</p> <p>table “tablename” – checks the specified tables and reports any inconsistencies found, making recommendations to repair those inconsistencies. The table operation may not be used on core system catalogs. Secondary indexes may be checked but cannot be repaired. A table lock is taken during verifydb table operations, but a database lock is not taken. Use this option <i>only</i> when you are using the report mode (-mreport).</p> <p>This operation also verifies referential integrity between the internal pointers for long data types stored in base table records and the extension table records they point to. Any inconsistencies are reported.</p> <p>Caution! <i>Using this option when you are in any run mode is not supported unless you are receiving assistance from Technical Support and are advised to do so; it can have severe, unexpected results.</i></p> <p>xtable “tablename” – functions like the table option, however xtable uses a stricter patch algorithm, which guarantees data integrity – with the risk that some valid data may be discarded. Use this option <i>only</i> when you are using the report mode (-mreport).</p>

Parameter	Description
-ooperation (cont.)	<p>Caution! Using this option when you are in any run mode is not supported unless you are receiving assistance from Technical Support and are advised to do so; it can have severe, unexpected results.</p> <p>dbms_catalogs—checks the dbms catalogs and reports any inconsistencies found, making recommendations to repair those inconsistencies. Use this option <i>only</i> when you are using the report mode (-mreport).</p> <p>Caution! Using this option when you are in runinteractive mode is not supported unless you are receiving assistance from Technical Support and are advised to do so; it can have severe, unexpected results. This operation is not supported in run modes other than runinteractive</p> <p>force_consistent—used to permit entry into a database that is inconsistent. This does not fix the problem with the database; it merely allows you to force the database to act as if it were in a consistent state. This can be very dangerous if used against a production database. Hidden data damage may render one or more tables in the database unrecoverable at some time in the future. Use this option <i>only</i> when you are using the report mode (-mreport).</p> <p>Caution! Using this option when you are in any run mode is not supported unless you are receiving assistance from Technical Support and are advised to do so; it can have severe, unexpected results.</p> <p>refresh_ldbs—directs verifydb to assure that a distributed database correctly reflects the release level of all remote databases that contain objects registered to the distributed database. It is recommended that you run this operation on a distributed database after you run upgradedb on any of the remote databases accessed by the distributed database.</p> <p>The distributed databases are specified by the -sscope parameter. For the refresh_ldbs option only, verifydb skips all non-distributed databases and processes only distributed databases. (In all other cases, verifydb processes only non-distributed databases.)</p>
-n	Nolog mode. Turns off the logging to the log file; logging to the terminal continues.
-lfilename	<p>Specifies an alternate log file (in the II_CONFIG location) to which verifydb is to log activity.</p> <p>The -n flag cannot appear if an alternate log is specified.</p>

Parameter	Description
-v	Verbose mode. Provides additional dialog messages when performing the verifydb operation. This flag applies only for table operations.
-uusername	Specifies the effective user for the session. For details, see the Standard Command Line Flags and Parameters section.

Examples

Execute verifydb against all databases for which you are the DBA, removing all unrequired disk files and logging all the operations that are performed:

verifydb -mrun -sdba -opurge

Run verifydb against the database teach_examp in report mode, looking for expired tables:

verifydb -mreport -sdbname "teach_examp" -oexpired_purge

Execute verifydb as the user fredk against all the databases for which fredk is the DBA, deleting temporary and expired tables:

verifydb -mrun -sdba -opurge -ufredk

Drop references to the table new_benefits in the database new_employee:

verifydb -mrun -sdbname "new_employee" -odrop_table "new_benefits"

Run consistency checks on the DBMS catalogs for the iidbdb database. Note that only the report mode is used for this catalog verify activity:

verifydb -mreport -sdbname "iidbdb" -odbms_catalogs

Run consistency checks on the DBMS catalogs for all databases that you own, with output going to the alternate log file checkdbs.log. Note again that only the report mode is used for this catalog verify activity:

verifydb -mreport -sdba -odbms_catalogs -lfcheckdbs.log

xmlimport

Imports the xml data file into Ingres.

Syntax

```
xmlimport dbname [uuser] [P] [-GgroupID] [-debug] xmlfile
```

Description

The `xmlimport` utility imports into the Ingres database any xml data file containing Ingres table data that conforms to Ingres dtd. You may have several table definitions and index definitions in the xml file. When `xmlimport` is run, the tables and indexes specified in the xml file are created in the provided database, and the data is uploaded.

Internally, `xmlimport` parses the xml file to generate an SQL script from the metadata information in the xml file and for the data files for each table's data. The `xmlimport` utility then runs the SQL script to create tables and upload the data from the data files. Files are created in the temp directory and deleted when the script has been run, except when the `-debug` flag is specified.

Parameter	Description
<i>dbname</i>	The name of the database being exported. The <i>vnode</i> and <i>server_class</i> are specified if required. For details, see the Standard Command Line Flags and Parameters section.
-u <i>user</i>	Specifies the effective user for the session. For details, see the Standard Command Line Flags and Parameters section.
-P	Password if the session requires a password.
-G <i>groupID</i>	Specifies a group identifier. For details, see the Standard Command Line Flags and Parameters section.
-debug	Leaves the generated xml file and the data files in the temp location. By default the files in this location are deleted.
<i>xmlfile</i>	The name of the xmlfile that needs to be imported into the database.

The xmlimport utility validates the xmlfile against the generic Ingres dtd. If the ingres dtd is External, it should be by default at the same location as the xml file. If Ingres is referred to the ingres.dtd in the \$II_SYSTEM/ingres/files area, the dtd should be present in the \$II_SYSTEM/ingres/files directory.

If the tables and indexes are already present in the database, an error displays and the data is appended to the existing table.

This utility is useful to import an xml file generated by genxml into an Ingres database. (See the genxml section for more information about exporting Ingres data in xml format).

Example

To import an xmlfile 'xmlout.xml' into an Ingres database testdb, run:

xmlimport testdb xmlout.xml.

The xmlimport utility parses xmlout.xml and then creates the tables and indexes defined in the xmlout.xml file in testdb database.

Ingres Utilities

This appendix lists Ingres utilities that are executed at the command line. These utilities require special privileges to invoke or are special purpose programs.

cacheutil

Returns information about shared memory buffer caches installed in the Ingres installation.

Syntax

cacheutil

Description

The cacheutil utility can be used to show brief or detailed information about existing caches that are no longer being used. It can also be used to destroy shared memory segments used for buffer caches that are no longer being used. Cacheutil will not destroy a buffer cache that is currently being used.

VMS

You must have OpenVMS privileges to use the cacheutil utility. 

When you see the CACHEUTIL> prompt, choose one of the following options:

- **list**
List the installation's existing shared buffer caches. The list gives the size of the cache and the number of connected DBMS servers.
- **show *cache_name***
Display detailed statistics on the specified shared buffer cache.
- **destroy *cache_name***
Destroy the shared memory segment associated with the specified cache name.

This is needed on systems where shared segments are allocated in a manner such that they are not automatically released when all DBMS servers connected to them are brought down. On these systems, if a DBMS with a shared memory cache fails or is brought down in an unsupported manner, the shared memory segment cannot be automatically cleaned up by the recovery system.

In systems where shared segments are automatically released, when the failed server is restarted, it will automatically clean up the old shared segment. In this case cacheutil is not necessary to release the shared memory.

However, if no server will be restarted that specifies the same cache name as the orphaned cache, then the shared segment must be cleaned up through the destroy option of cacheutil.

- **help**
Display help on the cacheutil utility.
- **exit**
Exit from the cacheutil utility.

catalogdb

Lists databases that you own.

Syntax

catalogdb [-uusername] [-vnode=vnode]

Description

The catalogdb utility is a forms-based interface that enables you to list your databases, the databases that you can access, the location names known to the system, the extensions made to your databases, and your user capabilities. See the accessdb command in the appendix, "Ingres Commands," for information on how to modify these attributes.

The following table lists valid command flags and parameters:

Parameter	Description
-uusername	Specifies the effective user for the session. For details, see the Standard Command Line Flags and Parameters section in the appendix, "Ingres

Commands".	
-vnode = <i>vnode</i>	Specifies a vnode name as described in the Standard Command Line Flags and Parameters section in the appendix, "Ingres Commands".

Catalogdb requires you to define the type of terminal you are using. When you invoke the catalogdb command, the following main menu appears:

Databases User Locations Help Quit :

These menu options are described in the following table:

Menu Option	Function
Databases	Detailed information on a selected database
User	Summary information about your user ID
Locations	Information about system location names
Help	Provides help
Quit	Exits the catalogdb program

Each command invokes a form to browse. Each form includes its own command menu, including a Help option – which provides a help message – and an End command, which returns you to the main menu.

Examples

To browse through data on your own account and databases, type the following:

catalogdb

As system administrator to browse the data for another user, type the following:

catalogdb -uPeter

cbf

Starts the Ingres/Configuration-By-Forms (CBF) utility.

Syntax

cbf [-host=*name*]

Description

The CBF utility displays current values of the server parameters and provides menu and screen selections for changing them. With CBF you can:

- Configure various components of the installation:
 - Name, DBMS, Bridge, Net, or Star server
 - Locking or logging segment
 - Security
 - Internet communications
 - Visual DBA Remote Command Server
 - Primary and secondary transaction log files
- Select which databases can be accessed by a DBMS server
- Reformat transaction log file(s) and enable/disable dual logging
- Reconfigure protocol accesses for the Net server
- Set a new value of any configuration parameter, or restore the factory default
- Automatically calculate configuration parameters derived from other parameters
- Protect any derived parameter from further change
- Run a system check for sufficient resources on a new configuration

With an optional -host parameter specified, the CBF utility is capable of configuring remote NFS client installations. The only difference in the operation of the utility in this mode is that system resource checking must be disabled.

cscleanup

A black rectangular button with rounded corners and a slight 3D effect. The word "UNIX" is written in white, uppercase, sans-serif font in the center.

Deallocates low-level Ingres shared memory and semaphore resources.

Syntax

cscleanup

Description

The `cscleanup` utility deallocates the UNIX shared memory and semaphore resources that were allocated by `csinstall` for use by Ingres. This utility is called by `ingstop`, so you should ordinarily have no need to run this utility.

Use `cscleanup` when the Ingres servers have aborted or you are forced to stop one of those servers by using the UNIX `kill` command. Use `csreport` and the UNIX command `ipcs` to verify the cleanup. If this utility fails for some reason, you can remove Ingres shared memory and semaphores with the UNIX command `ipcrm`.

The `cscleanup` utility does *not* deallocate shared memory buffer caches. For information about destroying shared caches, see the `cacheutil` section. ■

csreport



Display shared memory and semaphore information.

Syntax

csreport

Description

The `csreport` utility displays shared memory and semaphore information for your installation. Verify that `ingres` system administrator is the owner of any shared memory segments or semaphores used by your Ingres installation. Running `csinstall` or `ingbuild` as root or some other user can cause segments and semaphores to be allocated that are *not* owned by user `ingres`. `Cscleanup` will *not* remove these, so they will have to be specifically removed under the user login of the user who created them. Use the UNIX utility `ipcrm` in the format:

ipcrm -m*mid* | **-s***sid*

where *mid* is the ID number of the shared memory segment and *sid* is the semaphore identifier.

Examples

Here is an example of csreport output:

```
!Installation version 610008
!Max number of servers 16
!Description of shared memory for control system:
!Key 0x493D183C: size 16384 attach 00000000
!Description of shared memory for logging & locking system:
!Key 0x493D184E: size 229376 attach C0005000
!Semaphore information for installation:
! sysV semid 38, 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 0 0 0 0 0 0 0
!Event system: used space 13268, length space 16384
```

Csreport output can be tailored to obtain the server connect ids for iimonitor input. The following command shows the processes that are currently running:

csreport | grep "inuse 1"

Here is a sample output:

```
!inuse 1, pid 3375, connect id 2217, id_number 0, semid 3900
!inuse 1, pid 3384, connect id 2220, id_number 1, semid 4101
!inuse 1, pid 3389, connect id , id_number 2, semid 0
```

The recovery server is listed first and can be connected to iimonitor with the connect ID 2217. The other server in this listing has the connect ID 2220. Processes (as opposed to servers) do not have a connect ID. ■

deregdocs

Deregister specified files from an Ingres/ICE business unit.

Syntax

deregdocs *options filename(s)*

Description

The deregdocs utility allows a system administrator to deregister files from Ingres/ICE in bulk without having to use the Visual DBA utility. With this utility, you can build scripts to aid in the maintenance of an Ingres/ICE web site.

The *filename(s)* parameter specifies one or more files that you want to deregister. Alternatively, you can use the **-i***listfile* flag to specify the names of multiple files in the specified *listfile*, saving you typing, and input errors while using the command line.

The following table lists valid options for this utility:


Options	Description
-u <i>unitname</i>	The name of the business unit to which the files belong.
-l <i>location</i>	The name of the location of the files within the business unit. (A business unit can have more than one location associated with it.)
-i <i>listfile</i>	The name of the file that contains the list of files (pages and facets) to be removed from the business unit.
-help	Displays syntax online.

ICETranslate


Reads an Ingres/ICE XHTML template file and convert it into the equivalent Ingres/ICE macro template file. This utility must be used when creating web sites with Ingres/ICE XHTML template files.

Syntax

UNIX

ICETranslate *input_file.xml* [> *output_file.html*] 

Windows

icetranslate *input_file.xml* [> *output_file.html*] 

Description

On UNIX platforms, the ICETranslate utility must be entered with the first 4 letters in uppercase. On Windows, this is not a requirement. ICETranslate reads an Ingres/ICE XHTML template file and converts it into the equivalent Ingres/ICE macro template file. This utility writes to its standard output so you must redirect the output to the file name of your choice. Please see the example below for how to do this. Once the file has been registered with the Ingres/ICE server, it is available for use.

Example

```
icetranlate my_query.xml > my_query.html
```

iigenres

Used by the installation process to generate a default CONFIG.DAT file.

Syntax

```
iigenres [-v] [host] [rule_map]
```

The parameters are described in the following table:

Parameter	Description
-v	The -v (for verbose) flag, displays system commentary to the standard output device as the iigenres operation continues.
<i>host</i>	The <i>host</i> parameter if provided, specifies the host for which the configuration should be generated.
<i>rule_map</i>	Specifies the rule map file to use. The rule map file, contains a list of the rule system files (CRS extension files) to use when generating the default configuration.

Description

The iigenres utility generates a default configuration for an Ingres installation. Use this utility to regenerate the Ingres configuration file, should it be accidentally deleted or corrupted.

Example

```
iigenres usilgpqo default.rfm
```

iigetres

Used by CBF and the installation process to get the value of a named resource.

Syntax

iigetres *name*

where *name* is the name of the configuration parameter, as it appears in the default configuration file.

Description

The iigetres utility looks up a value in the default configuration file (CONFIG.DAT) and prints the value to the standard output device.

iimklog

Invoked by CBF and the Configuration Manager to generate an Ingres transaction log file.

Syntax

iimklog

Description

The current transaction log must be deleted before creating a new one.

iiremres

Removes a specified configuration parameter from CONFIG.DAT and recalculates any derived resources.

Syntax

iiremres [-v] *name*

The parameters are described in the following table:

Parameter	Description
-v	Displays system commentary to the standard output device as the operation continues
<i>name</i>	The parameter name as it appears in the default configuration file

Example

iiremres -v ii.lusilgpqo.gcn.local_vnode

iisetres

Sets a configuration resource in CONFIG.DAT and recalculates derived resources.

Syntax

iisetres [-v] [-p] *name value*

The parameters are described in the following table:

Parameter	Description
-v	Displays system commentary to the standard output device as the operation continues
-p	Protects the parameter from further automatic adjustments
<i>name</i>	Name of the configuration parameter to set
<i>value</i>	The value of the configuration parameter

Example

iisetres ii.usilgpqo.dbms.*.default_page_size 4096

iivalres

Validates a configuration resource for rule system constraint violations.

Syntax

iivalres [-v] *name value* [rule_map]

The parameters are described in the following table:

Parameters	Description
-v	Displays system commentary to the standard output device as the operation continues
<i>name</i>	Name of the configuration parameter to set
<i>value</i>	The value of the configuration parameter
<i>rule_map</i>	Specifies the rule map file to use. The rule map file contains a list of the rule system files (CRS extension files) to use when generating the default configuration.

Example

iivalres -v ii.usilgpqo.dbms.*.default_page_size 2048

iimonitor

Monitors and administers DBMS and recovery servers.

Syntax

iimonitor *server_id*

Description

The iimonitor utility is an operating system level utility that allows an Ingres system administrator or other privileged user to perform a number of session and server connection functions. The *server_id* refers to the server's GCF address. You can obtain this address using the iinamu or (UNIX) csreport utility.

You can use the iimonitor utility to examine the status of a DBMS server or to shut down a server. In addition, you can use iimonitor to monitor or shut down a particular server session and perform other server control functions.

If you are using iimonitor to terminate a session that has an active transaction, the server first rolls back the transaction. The session is not completely removed until the rollback is complete.

At the IIMONITOR > prompt, the following commands are available:

Command	Description
help	Lists the available commands.
show server [listen shutdown]	<p>The show server command 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.</p> <ul style="list-style-type: none">■ The listen option displays the server listen state, either OPEN or CLOSED.■ The shutdown option displays the server shutdown state, either OPEN or CLOSED.
show [user] system all sessions[formatted]	<p>The show sessions command displays a list of active sessions and their current state:</p> <ul style="list-style-type: none">■ The user option, or the default if no option is specified, gives the information on user sessions.■ The system option provides the information on system sessions.■ The all option provides the information on both user and system sessions.■ If formatted is specified, additional information is shown for each session in a block format.

Command	Description
show [<i>user</i>] system all sessions [formatted] (<i>cont.</i>)	<p>Following is an explanation of session states:</p> <p>CS_EVENT_WAIT: Session is waiting for an event. The event type is shown in parentheses. The session is waiting for:</p> <ul style="list-style-type: none"> ■ (LOCK)—a lock to be granted ■ (DIO)—a disk i/o to complete ■ (LOG-IO) - the completion of i/o to the transaction log ■ (BIO)—a message to be received from or sent to its associated user interface ■ (GWFIO)—completion of a request it has made through a gateway to a non-Ingres database <p>CS_MUTEX: Awaiting a semaphore (access to a system data structure)</p> <p>CS_COMPUTABLE: Runnable and waiting for a chance to run</p> <p>CS_INTERRUPT: The current wait state may be interrupted if desired.</p>
set server shut closed open	<p>Affects the server running state, as follows. This command may only be run by a privileged user:</p> <ul style="list-style-type: none"> ■ The shut option disallows additional connections and shuts the server down when currently connected sessions finish. ■ The closed option is the same as shut except that the server is not terminated on idle. ■ The open option re-allows connections and cancels a pending set server shut.
stop server	<p>Stops the server. This command may be run only by a privileged user.</p> <p>This commands stops the server immediately. Only use this if absolutely necessary, for example, if an Ingres tool program is hanging.</p>

The following commands 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:

Command	Description
format <i>session_id</i> all	Gives a synopsis of the information about a session. The all option gives the information on all active sessions.
remove <i>session_id</i>	Disconnects a particular user session. This command may be run only by a privileged user. This command cannot be used to drop system threads.
suspend <i>session_id</i>	Suspends a compute-bound session to allow a trace of the problem.
resume <i>session_id</i>	Resumes a suspended session.
quit	Terminates the iimonitor session.

The system sessions shown by iimonitor include server threads. The server threads are described in the following table:

Thread	Description
Admin thread	Assists in administrative chores. This thread cannot be seen with iimonitor.
Idle thread	Assists in administrative chores.
Event thread	Handles event processing.
Write behind thread	Performs write behind processing.
Consistency point thread	Previously called the Fast Commit thread; now all servers use this thread to perform consistency points (even non-Fast Commit servers).
Dead process thread	Checks for abnormal process termination.
Force abort thread	Performs force abort processing.
Group commit thread	Performs group commit processing.
Lock callback thread	Performs all lock callback actions.
Log writer thread	Performs transaction logfile writes.
License thread	Checks the validity of the Ingres license.
Security audit thread	In C2 enabled servers only, performs security auditing.

The iimonitor utility can also be used to connect to the recovery process (DMFRCP). Formatting the recovery thread in the recovery process displays the current state of online recovery operations, if any are taking place. The recovery process is multi-threaded, and has the following threads that can be viewed with iimonitor:

Thread	Description
Recovery thread	Performs online recoveries
Consistency point timer thread	Performs timed consistency points

iinamu

Monitors and administers the name server.

Syntax

iinamu

Description

You can use the Name Server Maintenance Utility (iinamu) to display DBMS server information and administer the Name server. Only a privileged user can execute the administrative options, such as adding or deleting entries or stopping the name server process.

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

Command	Description
show [<i>svr_type</i>]	Show the list of currently registered servers. <i>svr_type</i> can be: INGRES: Ingres – DBMS server type and the default COMSVR – GCC Net server process type. IINMSVR – Name server process STAR – Star server process

Command	Description
<div>Windows</div> <div>UNIX</div> <div>VMS</div>	<p>The following is an example of SHOW command output:</p> <pre> II\INGRES\6 II\INGRES\aa INGRES * 1201 INGRES * 1243 INGRES * II_DBMS_389 INGRES * II_DBMS_4E6 </pre> <p>The first column is the server type, the second is a list of databases registered to be served by this DBMS server, and the third is the GCF_ADDRESS.</p> <p>The database name entry * means that the server has registered to service requests for any database.</p> <p>The GCF_ADDRESS column contains the GCF specific address for access to this server. This can be used with the iimonitor command.</p>
add <i>svr_type obj_name gcf_address [flag]</i>	<p>Manually add to the list of registered servers. Only a privileged user may run this command. See also the information below on non-registered servers. The optional <i>flag</i> can be one or more of the following:</p> <p>sole – the server being added is a sole server.</p> <p>merge – existing entries for the server at GCF_ADDRESS should not be deleted when the new entry is added. This flag may be used to add a new object to be serviced by an existing server.</p>
delete <i>svr_type obj_name gcf_address</i>	Manually delete a server from the list of registered servers. Only a privileged user may run this command.
stop	Stop the GCF name server. This is the correct way to stop the name server. If the name server is stopped while DBMS servers are running, no users can connect to those servers. Connected users will function undisturbed until they disconnect their sessions. Only a privileged user may run this command.
help	Displays command information.
quit	Quit iinamu.

It is sometimes desirable to start DBMS servers that are not publicly registered with the name server. This can be done with the `nonames` option as shown below. If this option is included, then DBMS servers will not register with the name server upon start up and will therefore be invisible to iinamu. You can still find the GCF address, however, by examining `II_DBMS_SERVER`, if it is defined.

The default setting is `names`, thus automatically registering with the name server on DBMS server start up.

Examples

UNIX output formats are shown:

- Show all DBMS servers:

IINAMU> show ingres

Here is a sample output:

```
INGRES * 3105
INGRES * 4204
```

- The first column is the server type.
- The second column is a list of databases registered to be served by this DBMS server. The database name entry `*` means that the server has registered to service requests for any database.
- The third column is the `GCF_ADDRESS`. This column contains the GCF address for access to this server. The example shows that there are two DBMS servers, running at GCF addresses 3105 and 4204. This can be used with the `iimonitor` command.

- Show the communications server registrations:

IINAMU> show comsvr

Here is a sample output that indicates there are two communications servers running:

```
COMSVR * 3197
COMSVR * 3321
```

- Add a DBMS server with GCF address 1093 to the name server registry. Any database may use this server:


IINAMU> add ingres * 1093

Add a DBMS server with GCF address 2180 to the name server registry. Only the *salesdb* database may use this server:

IINAMU> add ingres salesdb 2180

- Delete a DBMS server with GCF address 1093 from the GCN registry so it is no longer visible from the name server:

IINAMU> delete ingres * 1093

Windows

- Stop the name server:

```
IINAMU> stop
```

- Show all DBMS servers:

```
IINAMU> show ingres
```

Here is a sample output:

```
INGRES * II¥INGRES¥aa
INGRES * II¥INGRES¥ca
```

- The first column is the server type
- The second column is a list of databases registered to be served by this DBMS server. The database name entry * means that the server has registered to service requests from any database.
- The third column is the server identifier. This can be used with the `iimonitor` command.

- Show the communication server registrations:

```
IINAMU> show comsvr
```

Here is a sample output that indicates there are two communications servers running:

```
COMSVR * II¥COMSVR¥b3
COMSVR * II¥COMSVR¥a2
```

- Add a DBMS server with a process ID of af to the name server registry. Any database may use this server:

```
IINAMU> add ingres * II\INGRES\af
```

Add a DBMS server with a process ID of ab to the name server registry. Only the *salesdb* may use this server:

```
IINAMU> add ingres salesdb II\INGRES\ab
```

- Delete a DBMS server with a process ID of af from the name server registry so it is no longer visible from the name server:

```
IINAMU> delete ingres * II\INGRES\af
```

- Stop the name server:

```
IINAMU> stop
```

iishowres

A black rectangular button with the word "UNIX" in white capital letters.

Displays the amount of shared memory used by the locking and logging system.

Syntax

iishowres [-d] [-help]

Description

The iishowres utility displays the amount of shared memory the locking and logging system uses to manage logging and concurrency in an installation. The size of the logging and locking memory segment depends on several parameters such as the number of concurrent users, the number of open databases, and the number of lock lists.

If you do not use the -d flag, iishowres returns the total amount of shared memory needed by the logging and locking memory segment (in bytes).

If you use the -d flag, iishowres returns the amount of memory each component of the locking and logging system uses.

The -help option displays the iishowres syntax online. 

iizic

Customizes time zone table files.

Syntax

iizic [-doutput_directory] timezone_rule_file

Description

The iizic utility customizes the time zone table files that are provided, allowing the user to tailor time zone information when needed for special cases not covered by the supplied time zone selections.

This utility works similarly to the UNIX `zic` utility.

The default *timezone_rule_file* is indicated by `IL_TIMEZONE_NAME`. When `IL_TIMEZONE_NAME` is set to the name of a time zone table, utilities such as `date` and C functions such as `localtime()` and `gettimeofday()` make GMT adjustments using the time zone table.

The Olsen time zone table for making Greenwich Mean Time (GMT) adjustments is supported. This method builds time zone tables based on the given rule file. The time zone tables consist of start and end GMT times for each time period. In addition, the corresponding GMT offset value and the abbreviation of the time zone name for each period are kept in the table.

When retrieving internal dates, which are stored in GMT, Ingres searches for the correct time period and then applies the corresponding GMT offset value as well as the abbreviated time zone name to determine the local time.

iizck

Displays the time zone table files.

Syntax

iizck [-name=*timezone_name*] [-f*pathname/filename*]

Description

The `iizck` utility displays the time zone table files. The time zone table currently in effect can be checked, or you can check a newly created one with `iizck` by specifying the `-f` parameter.

The display shows the Greenwich Mean Time (GMT) offset for the time zone. For those timelines with Daylight Savings Time (DST) adjustment, it gives each date that the GMT offset changed. If no parameters are specified, `iizck` displays the `IL_TIMEZONE_NAME` table. For example:

```
timezone name: australia-nsw
timezone file: c:\ingres\%ingres%\files\%zoneinfo%\astrl\%nsw
```

Period Begin (YYYY_MM_DD HH:MM)	GMT offset (Minute)	
1971_10_31 02:00	660	EST
1972_02_27 03:00	600	EST
1972_10_29 02:00	660	EST

The dates range from 1971 to 2037.

ingbuild (UNIX) or vmsinstal (OpenVMS)

Runs the Ingres installation procedure.


Syntax

UNIX


Interactive mode:

ingbuild

Command line mode:

ingbuild [*flags*] [*distribution_medium*] [-help] 

VMS

@vmsinstal *distribution_medium*. 

Description

The ingbuild (UNIX) or vmsinstal (OpenVMS) utility performs the initial install of your Ingres release. Use this utility when installing for the first time and when updating to a new version.

During install, you will unload the Ingres software and set the non-default configuration parameters. If you are updating an existing installation, it must be shut down before you run the install program. Shut it down using ingstop.

You must be logged into the ingress system administrator account to use the install utility.

ingnet

Permits you to view and define Ingres/Net node definitions.

Syntax

ingnet

Description

The Ingres Network Utility (ingnet) is a stand-alone tool that permits you to view and define Ingres/Net node definitions, which then allows you to connect to remote Ingres installation through Ingres/Net. In addition, ingnet allows you to launch the stand-alone Database Object Manager, Monitor, and SQL/Test windows for such installations.

ingstart

Starts up an Ingres installation.

Syntax

```
ingstart [-iigcn | -dmfrcp | -dmfacp | -rmcmd | -icesvr |  
         [-iidxbms | -iigcc | -iigcb | -iijdbc | -iistar [= config_name]]] [-help]
```

Description

In the start process, ingstart checks that you have sufficient operating system resources to run Ingres and have initialized the log file. The utility then starts up the name server, recovery and archiver processes, and DBMS server(s). If you are authorized to run them, ingstart also starts up the Net Communications server (iigcc), the Visual DBA Remote Command server (rmcmd), the Ice server (icesvr), and the Star server (iistar). The ingstart utility starts up all the elements of your installation in the correct sequence.

Note: If a server or process is specified, only that component is started up.

Important! You must be logged into the ingres system administrator account to use this command.

The following table lists valid command flags and parameters:

Parameter	Description
-iigcn	Starts the Name server.
-dmfrcp	Starts the recovery process.
-dmfacp	Starts the archiver process.
-rmcmd	Starts the Remote Command server required by Visual DBA.
-icesvr	Starts the Ice server.

Parameter	Description
-iidxbms	Starts the DBMS server. You can optionally specify a <i>config_name</i> as the name of that DBMS server.
-iigcc	Starts the Net communications server. You can optionally specify a <i>config_name</i> as the name of that communications server.
-iigcb	Starts the Bridge server. You can optionally specify a <i>config_name</i> as the name of that communications server.
-iidxbc	Starts the JDBC server. You can optionally specify a <i>config_name</i> as the name of that communications server.
-iistar	Starts the Star server. You can optionally specify a <i>config_name</i> as the name of that Star server.
<i>config_name</i>	If specified, this is the name of the server being started up. This name appears in the Config Name field of the Ingres Configuration Manager main window.
-help	Displays syntax online.

Examples

To start an additional default DBMS server, type the following command:

```
ingstart -iidxbms
```

To start the “speedy” Net server, type the following command:

```
ingstart -iigcc=speedy
```

UNIX Examples

Start up your installation interactively using the configuration option, where \$II_SYSTEM is set to /install/r6:

```
setenv II_SYSTEM /install/r6  
ingstart
```

Start up your installation automatically by including ingstart in the /etc/rc or other boot script.

```
su ingres -c /install/r6/ingres/utility/ingstart \  
> /dev/console
```


ingstop

Shuts down an Ingres installation.

Syntax

```
ingstop [-f] [-timeout=minutes] [-kill] [show]  
        [-force | -immediate] [-help]
```

Description

The **ingstop** command shuts down the servers in an Ingres installation in an orderly fashion for reconfiguration or system shutdown. It automatically brings down all or selected server-related processes in the installation. It can shut down servers, the archiver and recovery processes, and deallocate shared memory.

The **ingstop** command provides a graceful shutdown: the program waits for all traffic to terminate and for all users to exit from Ingres before shutting down the Ingres processes.

You can optionally specify a forced shutdown.

Because it is important that processes be brought down in the correct sequence, you should use **ingstop** whenever you shut down the entire installation. You may also use **ingstop** to shut down the locking and logging system.

You must be logged into the **ingress** system administrator account to use this command.

The following table lists valid command flags and parameters:

Parameter	Description
-f	Forces immediate shutdown.
-timeout=minutes	Waits the specified number of minutes for active sessions to terminate before shutting down the installation.
-kill	Shuts down the installation without waiting for currently executing transactions to complete. Transaction recovery will be required when the installation is restarted. Any Ingres processes that cannot be shut down by conventional means are terminated.

Parameter	Description
-force	Forces the shut down of active servers in the installation without waiting for users to disconnect.
-immediate	Shuts down the installation immediately. It does not wait for currently executing transactions to complete. Transaction recovery will be required when the installation is restarted.
-show	Displays a list of currently running Ingres processes that would have been stopped by ingstop without actually shutting any of them down.
-help	Displays syntax online.

ivm

Manages Ingres components and provides access to other utilities.

Syntax

ivm

Description

The Ingres Visual Manager (IVM) is an integrated, GUI tool that provides a global view into your Ingres installation. It serves as a system console from which you can manage Ingres components and access other utilities. This utility captures events that are occurring in the system and allows them to be filtered for emphasis, according to the system administrator's preferences.

Use IVM to do the following:

- Start and stop the Ingres installation
- Monitor the status of the installation or individual servers
- View and configure system, user, and other types of parameters
- View log files and event statistics for the installation or individual servers
- View and define error message alerts

lockstat

Displays locking status.

Note: The forms-based ipm utility incorporates the lockstat functions.

Syntax

```
lockstat [-help | -summary | -statistics | -lists | -user_lists  
         | -special_lists | -resources]
```

Description

The lockstat utility displays locking status information for your Ingres installation. It allows you to examine the state of the Lock Database by providing a summary listing and a snapshot of the installation's locking activity.

The lockstat options are described in the following table. Type **lockstat**, the default, to display all reports as follows:

Locking System Quotas
Locking System Summary
Locks by Lock List (User and Special)
Locks by Resource

lockstat Options	Description
-help	Displays the lockstat command options online.
-summary	Locking System Quotas
-statistics	Locking System Quotas Locking System Summary
-lists	Locks by Lock List (User and Special)
-user_lists	Locks by Lock List (User)
-special_lists	Locks by Lock List (special, that is, "NONPROTECT")
-resources	Locks by Resource

This tool is useful for finding lock contention and concurrency problems. It will help you identify locking bottlenecks so that you can correct the problem by setting lockmode appropriately or by remodeling the application if it is a fundamental design or program flow problem.

Example

A compressed example of the output from the lockstat utility follows. An explanation of each part of the output appears in the What to Look for in a Lockstat Output section:

```

=====Fri Apr 25 13:34:06   Locking System Quotas=====
Total Locks      65000   Total Resources   65000
Locks per transaction      500
Lock hash table   12983   Locks in use      129
Resource hash table 12983   Resources in use   127
Total lock lists   1480   Lock lists in use   57

=====Fri Nov 13 13:34:06   Locking System Summary=====
Create lock list  49395   Release lock list  49327
Request lock      586295   Re-request lock    457810
Convert lock      68157   Release lock       410846
Escalate          10     Lock wait          27489
Convert wait       2     Convert Deadlock    1
Deadlock Wakeups  2218   Max dlk queue len   10
Deadlock Search   1947   Deadlock            134
Cancel            135   Convert Search       2
Allocate CB       1161478   Deallocate CB       1160861

LBK Highwater     3     LLB Highwater       87
SBK Highwater     5     LKB Highwater       428
RBK Highwater     5     RSB Highwater       420
Max Local dlk srch 3     Dlk locks examined  2361
Max rsrc chain len 5     Max lock chain len   5
Callback Wakeups   0     Callbacks Invoked     0
Callbacks Ignored  0

-----Locks by lock list-----
Id: 00000001 Tran_id: 0000000000000001A R_llb: 00000000 R_cnt: 0
Wait: 00000000 Locks: (0, 0/250) Status: NONPROTECT,NOINTERRUPT PID: 10781 SID:0000000E
Id: 00000002 Tran_id: 00000000000000019 R_llb: 00000000 R_cnt: 0
Wait: 00000000 Locks: (0, 0/250) Status: NONPROTECT,NOINTERRUPT PID: 10781 SID:00000009
Id: 00000003 Tran_id: 00000000000000018 R_llb: 00000000 R_cnt: 0
Wait: 00000000 Locks: (0, 0/250) Status: NONPROTECT,NOINTERRUPT PID:
10781 SID:0000000C
Id: 00000004 Tran_id: 0000334C3362D62B R_llb: 00000000 R_cnt: 0
Wait: 00000000 Locks: (0, 0/250) Status: NONPROTECT PID:
10781 SID:00000007
Id: 00000005 Tran_id: 00000000000000016 R_llb: 00000000 R_cnt: 0
Wait: 00000000 Locks: (0, 0/250) Status: NONPROTECT,NOINTERRUPT PID:
10781 SID:00000007
Id: 00000006 Tran_id: 00000000000000015 R_llb: 00000000 R_cnt: 0
Wait: 00000000 Locks: (0, 0/250) Status: NONPROTECT,NOINTERRUPT PID:
10781 SID:0000000B
Id: 00000007 Tran_id: 00000000000000014 R_llb: 00000000 R_cnt: 0
Wait: 00000000 Locks: (0, 0/250) Status: NONPROTECT,NOINTERRUPT PID:
10781 SID:00000005
Id: 00000008 Tran_id: 00000000000000013 R_llb: 00000000 R_cnt: 0
Wait: 00000000 Locks: (0, 0/250) Status: NONPROTECT,NOINTERRUPT PID: 10781 SID:00000006
Id: 00000009 Tran_id: 00000000000000012 R_llb: 00000000 R_cnt: 0
Wait: 00000000 Locks: (1, 0/250) Status: NONPROTECT,NOINTERRUPT PID:
10781 SID:00000001
Id: 00000026 Rsb: 0000003C Gr: N Req: N State: GR PHYS(1)
KEY(AUDIT, LABEL_CACHE)
Id: 0000000A Tran_id: 00000000000000011 R_llb: 00000000 R_cnt: 0
Wait: 00000000 Locks: (0, 0/0) Status: NONPROTECT, PSHARED PID:10781
SID:00000001

Id: 0000000B Tran_id: 00000000000000010 R_llb: 00000000 R_cnt: 0

```

Wait: 00000000 Locks: (95,0/250) Status: NONPROTECT,NOINTERRUPT,
SHARED PID:10781 SID:00000001

```
Id: 0000001F Rsb: 00000034 Gr: IS Req: IS State: GR PHYS(1)
-----Locks by resource-----
Id: 00000020 Gr: IS Conv: IS Cbacks 0 Value: <0000000000000002>
KEY(SV_PAGE, DB=00000001, TABLE=[1, 0], PAGE=3)
Id: 0000000A Lib: 0000000B Gr: IS Req: IS State: GR PHYS(1)
Id: 00000021 Gr: IS Conv: IS Cbacks 0 Value: <0000000000000000>
KEY(SV_PAGE, DB=00000001, TABLE=[1, 0], PAGE=30)
Id: 0000000B Lib: 0000000B Gr: IS Req: IS State: GR PHYS(1)
Id: 00000022 Gr: IS Conv: IS Cbacks 0 Value: <0000000000000000>
KEY(SV_PAGE, DB=00000001, TABLE=[1, 0], PAGE=2)
Id: 0000000C Lib: 0000000B Gr: IS Req: IS State: GR PHYS(1)
Id: 00000023 Gr: IS Conv: IS Cbacks 0 Value: <0000000000000000>
KEY(SV_PAGE, DB=00000001, TABLE=[1, 0], PAGE=23)
Id: 0000000D Lib: 0000000B Gr: IS Req: IS State: GR PHYS(1)
Id: 00000024 Gr: IS Conv: IS Cbacks 0 Value: <0000000000000000>
KEY(SV_PAGE, DB=00000001, TABLE=[1, 0], PAGE=1)
Id: 0000000E Lib: 0000000B Gr: IS Req: IS State: GR PHYS(1)
Id: 00000025 Gr: IS Conv: IS Cbacks 0 Value: <0000000000000000>
KEY(SV_PAGE, DB=00000001, TABLE=[1, 0], PAGE=33)

Id: 0000000F Lib: 0000000B Gr: IS Req: IS State: GR PHYS(1)
```

What to Look for in a Lockstat Output

Keep in mind that lockstat gives detailed statistics on *all* locking activity in the installation. If there is much activity, there will be a considerable quantity of lockstat output.

The key items to examine in lockstat output are the waiting statistics. Look especially at:

- Lock wait in the Locking System Summary
- Deadlock in the Locking System Summary
- Wait in the Locks by Lock List
- Status WAIT in the Locks by Lock List

Interpreting Locking System Quotas

The first portion of the output shown above is a summary listing of locking quotas for this installation. All values are cumulative from the time ingstart was run for this iteration of the system. The meaning of each entry is described in the following table:

Field	Description
Total locks	Maximum number of locks in the installation
Total Resources	Maximum number of lockable resources in the installation
Locks per transaction	Maximum number of locks that may be acquired by a transaction

Field	Description
Lock hash table	Number of hash buckets in the locking system hash table
Locks in use	Number of locks currently in use in the installation
Resource hash table	Number of hash buckets in the resource hash table
Resources in use	Total number of countable resources in use
Total lock lists	Maximum number of lock lists available
Lock lists in use	Number of lock lists currently in use

Interpreting the
Locking System
Summary

The next portion of the output shown above is a summary listing of locking activity for this installation. Again all values are cumulative from the time `ingstart` was run for this iteration of the system. The meaning of each entry is described in the following table:

Field	Description
Create lock list	Number of times a lock list was created for server, session, or transaction
Release lock list	Number of times a release of a lock list occurred for a server, session, or transaction
Request lock	Number of new lock requests that the locking system processed
Re-request lock	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	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	Number of times a specific logical lock is released, as opposed to a full, partial, or physical lock release
Escalate	Number of times a partial release occurred to allow lock escalation from page to table level

Field	Description
Lock wait	Number of times a new lock request had to wait to be granted
Convert wait	Number of times an existing lock waited for conversion to a different lock mode
Convert deadlock	Number of times a request for conversion turned into a deadlock
Deadlock Wakeups	Number of times the interval-based deadlock detection thread was awakened
Max dlk queue len	Maximum number of waiting lock lists examined by the deadlock detection thread
Deadlock search	Number of times a deadlock search was initiated
Deadlock	Number of times that deadlock existed
Cancel	Number of times a lock request was canceled due to a time-out or interrupt
Convert search	Number of times a convert deadlock search was initiated. The searches are performed when converting a lock from one mode to another.
Allocate CB	Number of locking control block allocations
Deallocate CB	Number of locking control block deallocations
LBK Highwater	Maximum number of lock list blocks allocated
LLB Highwater	Maximum number of lock lists allocated
SBK Highwater	Maximum number of lock blocks allocated
LKB Highwater	Maximum number of locks allocated
RBK Highwater	Maximum number of resource blocks allocated
RSB Highwater	Maximum number of resources allocated
Max Local dlk srch	Maximum number of locks examined to resolve
Dlk locks examined	Number of locks examined by the deadlock detection thread
Max rsrc chain len	Maximum length of a resource hash chain
Max lock chain len	Maximum length of a lock hash chain

The remaining fields are relevant only when the installation has been configured to run with the Distributed Multi-Cache Management (DMCM) protocol:

Field	Description
Callback Wakeups	Number of times the DMCM callback thread was awakened
Callbacks Invoked	Number of times callback functions were invoked to resolve a blocking cache lock
Callbacks Ignored	Number of blocking cache locks which had already been released by the time the callback function was invoked

Interpreting the Locks by Lock List Portion

The next portion of the lockstat utility prints out the lock information sorted by lock list. The first line item reports the lock list identifier. Any locks associated with the specified lock list are listed following the lock list description and indented to set them off.

Most lock lists represent transaction units and hold the locks owned by their transactions. Some lock lists are used to hold special server or cache locks required for processing; these lock lists are owned and managed by the DBMS server or recovery process rather than by user transactions.

Locks by Lock List fields are described in the following table:

Field	Description
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	Made up of three values: total number of locks currently on the list, number of logical locks on the list currently, and total number of locks allowed to be on this list

Field	Description
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 recovery (system lock lists) ORPHAN – lock list remaining without transaction EWAIT – waiting for 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 for is done NOINTERRUPT – lock requests on this list are non-interruptible
PID	Process ID of the lock list owner
SID	Session ID of the lock list owner

The values indented under individual lock lists are lock block values. These are described in the following table:

Lock Block Values	Description
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. <ul style="list-style-type: none">■ When checking contention on data pages, the key will contain PAGE, the database ID, the table reltid and reltidx, and the page number.■ ROW is a special type of lock used to reserve space for deleted rows in four core catalogs only: iirelation, iirel_idx, iiattribute and iidevices.

Interpreting the Locks by Resource Portion

The last portion of lockstat groups the individual locks by resource block and shows any contention that can lead to query performance problems.

The Locks by Resource fields are described in the following table:

Field	Description
Id	Internal Resource block identifier
Gr	Granted mode of the resource
Cbacks	This field is relevant only when the installation has been configured to run with the Distributed Multi-Cache Management (DMCM) protocol. It describes the number of resource locks which contain DMCM callback information.
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. These lock blocks are described in the following table:

Lock Block Values	Description
Id	Internal Lock block identifier
Llb	Lock list identifier on which this lock resides
Gr	Granted mode of the lock
Req	Requested lock mode
State	Current state of the lock (GR = granted, WT = waiting)

logstat

Displays logging status.

Note: The forms-based ipm utility incorporates the logstat functions. Also, you can use Visual DBA to monitor log information. See Visual DBA online help.

Syntax

logstat [-help]

The `-help` parameter displays logstat command options online.

Description

Logstat output is composed of the following major sections:

- **Logging System Summary**
Provides an overall view of the logging system. It yields information on how well the logging system is tuned.
- **Current log file header**
Gives quantitative information on the logging system, such as the size of the log file, log buffers, and CP interval.
- **List of active processes**
Provides information on processes currently active in the logging system.
- **List of active databases**
Provides users with statistical information on all the active databases in the logging system.
- **List of active transactions**
Provides statistical information per active transaction.

Each of these major sections is described below, followed by an example with a detailed interpretation.

Logging System Summary

The Logging System Summary on the logstat output provides an overall view of the logging system. It yields information on how well the logging system is tuned.

The fields are described in the following table in the order in which they appear on the logstat output:

Field	Description
Database add	Number of times a database has been added to the logging system. This number is incremented whenever a session is the first session to access the database.
Database removes	Number of times a database has been removed from the logging system. This number is decremented when the last session accessing the database exits.

Field	Description
Transaction begins	Number of times a transaction has been started in the logging system. This number indicates the total number of transactions started in the logging system.
Transaction ends	Number of times a transaction has been completed in the system during normal processing. This number corresponds to all the transactions that were committed or rolled back without incident and properly terminated. This value does <i>not</i> include FORCE-ABORT, LOGFULL or any other rcp actions that terminated a transaction abnormally.
Log read i/o's	Number of times a read was performed on the log file. This is a physical operation. This number corresponds to the number of times the log buffer was read from the transaction log file to perform an abort, archive or purge operation.
Log write i/o's	Number of times a write was performed on the log file. This is a physical operation. This number corresponds to the number of times a log buffer was written to the transaction log file.
Log writes	Number of writes from the database buffer into the log buffers. These are memory-to-memory writes.
Log forces	Number of requests made to the logging system to force the current log buffers to the log file. This is most frequently done to commit a transaction or to guarantee the consistency of the log file before writing an update to the database.
Log waits	Number of times any event wait condition requires a log buffer write to stall. These are events such as LOGFULL, CP writing, RECOVERY, ARCHIVING required, FREE WAIT for log buffers, OPENDB wait, log buffer SPLIT wait, wait for completion of log i/o (that is, from the log buffer to the log file).
Log split waits	The number of times a log split operation is delayed due to no free log buffers on the free queue. Log splits in and of themselves are not to be interpreted as bad events. What is potentially harmful is the inability of the logging system to proceed with the log record split. This situation can be remedied by adding additional log buffers to the system or by increasing the size of the current buffers to minimize the need for splits. Any modification should be examined in conjunction with the effect that it has on the other wait states.

Field	Description
Log free waits	Number of times all the log buffers are either in force mode or unavailable for writing. One log buffer is written to at a time. If log free waits is frequent, then an increase in the amount of log buffers from 4 to 8 may be the solution. Remember that an increase in the number of buffers requires (number_of_log_buffers * log_buffer_size) more memory on the host system.
Log stall waits	Number of times any writes to the log buffers are stalled due to either CP events (consistency point writing) or LOGFULL events occur. All logging system writes are stalled for users until these conditions are cleared. This is most often seen from the user's viewpoint as a "hung" system. Always check logstat for the status in the header block. If this value is LOGFULL then this is a stall condition.
Logfull BCP waits	Number of times a thread was stalled waiting for Begin Consistency Point information to be written to disk. This is a very brief stall performed at the start of a consistency point.
Logfull stall waits	Number of times a thread was stalled waiting for a LOGFULL condition to clear.
Log group commit	Number of times that multiple transactions participate in a log buffer flush to the log file.
Log group count	<p>Number of transactions that are participating in the flush to the log file. This value is the wait count associated with the Log group commit count above. This value is incremented based on the number of waiters at write completion time.</p> <p>The ratio of Log group count to Log group commit gives an indication of how effectively the group commit mechanism is working in the current configuration.</p>
Check commit timer	Number of times the timer associated with the group commits completes. This does <i>not</i> necessarily mean that a write to the log file has to occur, because the log buffer that initiated the timer may have already been written due to being full.
Timer write	Number of times a log file write actually occurs as a result of the timer expiration. As explained above, this will occur only if the buffer has not completely filled before this timer expires.

Field	Description
Inconsistent db	Number of times the logging system has had to mark a database inconsistent due to an inability to recover some portion of work that currently exists in the logging system.
Kbytes written	Number of bytes written to the log file
ii_log_file read	Number of physical reads of the primary log file
ii_dual_log read	Number of physical reads of the dual log file
Write complete	Number of times a write of the primary log file completes successfully
Dual write complete	Number of times a file write of the dual log file completes successfully

Current Log File Header

The Current Log File Header gives quantitative information on the logging system, such as the size of the log file, log buffers, and CP interval. This section has the following fields:

Field	Description
Block size	Size of the log buffer and log file blocks in bytes. The log file is organized as a series of blocks that are laid down in a circular fashion and used for on line backup.
Block count	Size of the log file in blocks
Buffer count	Number of log file buffers. These are shared by all processes connected to the logging system.
CP interval	Number of blocks between consistency points. CPs may also be caused by other events, such as archiver PURGEs and online checkpoints.
Logfull interval	Number of log file blocks used before LOG_FULL is signaled
Abort interval	Number of log file blocks that must be used before a FORCE_ABORT is signaled
Last Transaction Id	ID of the last transaction to write a log record
Begin, CP, End	Log addresses of the beginning of the log file, the last consistency point, and the end of the log file
Percentage of log file in use or reserved	Percentage of the log file that has either been used or is reserved for use by the recovery system

Field	Description
Log file blocks reserved by recovery system	Number of log file blocks reserved for transaction recovery operations. Space reserved by a transaction is freed when the transaction commits normally, or it is used to write compensation log records during transaction abort processing.
Archive Window	The segment of the log file that may be examined by the archiver for journal or dump processing
Status	<p>This field indicates the current logging system status. This field can take one or more of the following values:</p> <p>ACP_SHUTDOWN – the archiver is preparing to shut down. (This indicates that an rcpcnfig command with the shutdown option has been issued.)</p> <p>ARCHIVE – the archiver process is archiving journaled transactions to the journal files.</p> <p>BCPSTALL – the logging system is requesting the recovery process to start writing a begin consistency point.</p> <p>CKP_SBACKUP – the logging system marks the start of online backup. It marks this block as the online backup start block (SB). Ckpdb starts backing up the database.</p> <p>CLOSEDB – the logging system is in the process of closing a database.</p> <p>CPFLUSH – DBMS servers are flushing their modified pages to disk.</p> <p>CPNEEDED – the logging system is about to take a consistency point.</p>

Field	Description
Status (<i>cont.</i>)	<p>CPWAKEUP – the logging system is synchronizing the fast-commit threads.</p> <p>DISABLE_DUAL_LOGGING – the logging system is in the process of disabling dual logging.</p> <p>DUAL_LOGGING – dual logging is enabled. (Note that this does not mean that both primary and dual logs are active. For active logs look at the Active Log(s) field.)</p> <p>ECP – the logging system is requesting that the recovery process start writing an end consistency point.</p> <p>ECPDONE – the logging system has taken an end consistency point. This status flag is present most of the time while the logging system is functioning normally.</p> <p>FORCE_ABORT – the force-abort-limit has been reached; the oldest open transaction will be aborted.</p> <p>IMM_SHUTDOWN – the logging system has been told to shut down immediately. (This is displayed when the user invokes rcpconfig with the imm_shutdown option.) Note that the logging system does not perform any housekeeping as part of the shutdown process. The recovery process then becomes responsible for backing out any uncommitted transactions left in the log file once the logging system has been restarted.</p>

Field	Description
	LOGFULL – the log file is full. The Ingres system administrator should determine the cause of this and increase the log file size. A warning indicator is also displayed.
	MAN_ABORT – the logging system has been requested to manually abort a distributed transaction.
	MAN_COMMIT – the logging system has been requested to manually commit a distributed transaction.
	ONLINE – the logging system is on line. The logging and recovery systems are operating OK.
	OPENDB – the logging system is in the process of opening a database.
	PURGEDB – a database has been closed by the last user who had it open; the archiver is archiving transactions that belong to this database.
	RCP_RECOVER – the recovery process is recovering transactions from a runaway DBMS.
Status (cont.)	RECOVER – the logging system has requested the recovery process to perform recovery.
	START_ARCHIVER – this is an important status that indicates that the archiver has stopped and must be restarted by the DBA. <i>This is not done automatically.</i> If the DBA does <i>not</i> do it, the log file will eventually fill up, reaching the LOG_FILE_FULL limit, and cause the system to stall.
	START_SHUTDOWN – the logging system is shutting down. As part of the shutdown process, the logging system commits to disk all the committed transactions and backs out any uncommitted ones. The archiver also journals all the committed transactions for tables with journaling enabled.
Active Log(s)	Displays which log files are currently active.

List of Active Processes

The List of Active Processes provides information on processes currently active in the logging system. This section has the following fields:

Field	Description
ID	The internal logging system ID for a process

Field	Description
PID	The process ID
TYPE	Indicates the type of the active process. The type field can be: FCT—a DBMS server running Fast Commit SLAVE—a DBMS server not running Fast Commit ARCHIV—the archiver process MASTER—the recovery process
OPEN_DB	Number of times the server opened a database. The recovery process and archiver each have their own database opened at all times, while the server is the process that opens the databases.
WRITE	Number of writes this process has performed in the logging system
FORCE	Number of times this process requested that a log buffer be forced to disk
WAIT	Number of times any transaction in this process needed to wait for a logging system-related reason
BEGIN	Number of transactions started by this process
END	Number of transactions ended by this process

List of Active Databases

The List of active databases provides statistical information on all the active databases in the logging system. This section has the following display fields:

Field	Description
Id	The logging system ID number of an active database
Database	The first element in the row marks the database's name and the second element indicates the DBA's name.
Status	This field indicates the current state of the database.
Tx_cnt	Number of transactions currently active in the database
Begin	Number of transactions started in this database
End	Number of transactions ended in this database
Read	Number of reads that the logging system performed on behalf of this database
Write	Number of writes that the logging system performed on behalf of this database

Field	Description
Force	Number of times that the logging system had to force out the log buffer on behalf of this database
Wait	Number of times that the logging system had to wait for a log buffer on behalf of this database
Location	The physical location of this database in the file system
Journal Window	The active journal window on this database. In the case where there is no journaling active on the database, the window would have boundaries <0,0,0>. .<0,0,0>.
Start Backup Location	The log file end-of-file (EOF) address when a database backup is started. This is used during online backup processing.

List of Active Transactions

The List of active transactions provides statistical information on each active transaction. The fields in this section are described below:

Field	Description
Tx_id	Transaction ID used by the logging system
Tran_id	Uniquely identifies a transaction. It is used by both the logging and locking systems. This ID is useful when you want to follow a transaction from lockstat to logstat output.
Database	Unique ID of the database. This ID is the same as the ID in the List of active databases section of logstat.
Process	Process ID of the process currently working on this transaction. This field corresponds to the internal logging system ID in the List of active processes section of logstat.
Dis_tran_id	Currently not used
Session	The user session ID that owns this transaction. This is the same ID used in iimonitor output. Use this ID to locate the user and the terminal that initiated the transaction.
First	Log file address of the first record associated with this transaction
Last	Log file address of the last record associated with this transaction
Cp	First consistency point address taken that concerns this transaction

Field	Description
Write	Number of log buffer writes because of this transaction
Split	Number of times this transaction had to wait for a log buffer in order to write a log record that spanned multiple buffers
Force	Number of times the log buffer was flushed. The force conditions are commented in more detail under the Log forces field in the Summary section above.
Wait	Number of times this transaction had to wait for a logging system-related event
Reserved	Number of log blocks reserved by this transaction for recovery operations
Status	<p>Present status of this transaction. This field can take the following values:</p> <p>ACTIVE—this transaction has written a number of records to the log file.</p> <p>INACTIVE—this transaction is in the retrieve mode and has not written any records to the log file.</p> <p>PROTECT—this transaction is a user transaction (as opposed to an internal system transaction) and will be recovered in the event of a server or system failure.</p> <p>JOURNAL—this transaction must be journaled. This flag indicates that the transaction should be archived.</p>

Field	Description
Wait Reason	<p>Reason for wait. This field can take the following values with respect to the transaction:</p> <p>(not waiting): The transaction is not waiting.</p> <p>FORCE—waiting for a log force</p> <p>FREE—waiting for a free log buffer</p> <p>SPLIT—waiting for a log split completion</p> <p>HDRIO—waiting for log header I/O completion</p> <p>CKPDB—waiting for a ckpdb completion</p> <p>OPENDB—waiting for an open database completion</p> <p>BCPSTALL—waiting for BCP log write to complete</p> <p>LOGFULL—waiting because of LOGFULL condition</p> <p>FREEBUF—waiting for a free buffer</p> <p>LASTBUF—waiting for the last buffer in the transaction to be written</p> <p>BUFIO—waiting for a log buffer to be freed</p> <p>EVENT—waiting for a log event</p> <p>ABSOLUTE_LOGFULL—waiting at the absolute end of a LOGFULL condition</p>
User	<p>The owner of this transaction. This field can take the following values:</p> <ul style="list-style-type: none">■ logfile_I/O_thread—log file read/write thread■ group_commit_thread—group commit thread■ buffer_manager—the buffer manager■ log_reader_transaction—log file read/write thread■ recovery_thread—the DMFRCP recovery thread■ consistency_pt_thread—the consistency point thread■ consistency_point_timer—the consistency point timer thread■ write_behind—the write behind thread■ security audit thread—in C2 enabled systems only, the security audit thread■ username—user session

Example

The following example is typical partial output from logstat. A discussion of this output follows:

```
=====Wed Nov 11 13:42:30 1998 Logging System Summary=====
Database add          95      Database removes      79
Transaction begins    5675    Transaction ends      5659
Log read i/o's        6857    Log write i/o's      11008
Log writes            24116    Log forces            733
Log waits             5389    Log split waits      1774
Log free waits         0      Log stall waits       26
Log BCP waits         26      Logfull stall waits   0
Log group commit      2189    Log group count       2217
Check commit timer     0      Timer write           0
Inconsistent db        0      Kbytes written        15955
ii_log_file read       717    ii_dual_log read      6140
write complete        5503    dual write complete   5503

---Current log file header---
Block size: 4096 Block count: 2048 Buffer count: 4
CP interval: 102 Logfull interval: 1945 Abort interval: 1536
Last Transaction Id: 00002D5B2D5BF9F7
Begin: <760996814:778:3008> CP: <760996814:1019:1060>
End: <760996814:1299:2656>
Percentage of log file in use or reserved: 32
Log file blocks reserved by recovery system: 140
Archive Window: <760996814, 981, 2800>..<760996814, 1299, 2656>
Previous CP: <760996814, 778, 3008>
Status:      ONLINE, ARCHIVE, CPFLUSH
Active Log(s): LOG_FILE

---List of active processes---
ID      PID  TYPE  OPEN_DB  WRITE  FORCE  WAIT  BEGIN  END
-----
0001000C 7802  MASTER  7      217    62    916    7      0
00010012 7817  FCT     6      23838  552   4129   5656   5650
00010015 7811  ARCHIV  1      0      50    268    2      1
---List of active databases---
Id: FFFF0001 Database: ($recovery,$ingres) Status: NOTDB, ACTIVE
Tx_cnt: 13 Begin: 16 End: 1 Read: 0 Write: 217 Force: 583 Wait: 1707
Location: None

Journal Window: <0, 0, 0>..<0, 0, 0>
Start Backup Location: <0, 0, 0> (0, 0)
Id: 00280005 Database: (testdb, test) Status: JOURNAL, FAST_COMMIT, ACTIVE
Tx_cnt: 1 Begin: 17 End: 16 Read: 0 Write: 4690 Force: 6 Wait: 681
Location: /devsrc/65sun4/install/test/ingres/data/default/testdb
Journal Window: <760996814, 981, 2800>..<760996814, 1300, 3184>
Start Backup Location: <0, 0, 0> (0, 0)

---List of active transactions---

[... transaction information deleted ...]

Tx_id: 295D001D Tran_id: 00002D5B2D5BF9F7 Database: 00280005
Process: 00010012 Dis_tran_id: <0, 0> Session: 0093C000
First: <760996814, 1160, 3100> Last: <760996814, 1300, 3184>
Cp: <760996814, 881, 948>
Write: 463 Split: 107 Force: 0 Wait: 109 Reserved: 140
Status: ACTIVE, PROTECT, JOURNAL
Wait Reason: (not waiting)
User: <test>
```

Determining Proximity to FORCE-ABORT-LIMIT

To determine how close your installation is to the FORCE-ABORT-LIMIT, use the information from the section titled Current log file header in the logstat output. Four figures are relevant: the abort interval, the Block count, the Begin, and the End.

The number appearing after abort interval is the number of blocks in the log file that must be filled before the FORCE-ABORT-LIMIT is reached. The Block count refers to the total number of blocks in the log file. Begin refers to the block marking the log file's Beginning of File (BOF) and End refers to the block marking the log file's End of File (EOF). The numbers following Begin and End are divided into three groups separated by colons. It is the middle group of numbers that are most relevant. For example, in the sample output, block 778 marks the beginning of the log file.

To calculate how close your installation is to the FORCE-ABORT-LIMIT:

1. Calculate the number of blocks in use.

Because the log file is a circular file, the block marking the file's beginning can have a higher number than the block marking the file's end (see the Second Log File Header).

Consequently, there are two ways to determine the number of blocks in use:

- a. If the End of File is larger than the Beginning of File, subtract the BOF from the EOF to obtain the number of blocks in use. For example, in the sample output, the BOF is 778 and the EOF is 1299. The number of blocks in use in this example is:

$$1299 - 778 = 521$$

- b. If the End of File is smaller than the Beginning of File, subtract the BOF from the block count figure and add the result to the EOF to obtain the number of blocks in use. For example, in the second Log File Header, the Beginning of File is 1702, the Block Count is 2048, and the End of File is 107. The number of blocks in use is:

$$(2048 - 1702) + 107 = 453$$

2. Subtract the number of blocks in use from the Abort interval figure to determine how many blocks are available before the FORCE-ABORT-LIMIT is reached.

For example, in the sample output, 521 blocks are in use and the Abort interval is 1536, so the number of blocks still available is:

$$1536 - 521 = 1015$$

```
-----Current log file header-----
Block size: 4096 Block count: 2048 Buffer count: 4
CP interval: 102 Logfull interval: 1945 Abort interval: 1536
Last Transaction Id: 00002D5B2D5BFA03
Begin: <760996814:1702:2304> CP: <760996814:1873:3592> End: <760996815:107:20>
Percentage of log file in use or reserved: 30
Log file blocks reserved by recovery system: 180
Archive Window: <760996814, 1991, 3508>..<760996815, 107, 220>
Previous CP: <760996814, 1702, 2304>
Status:      ONLINE, ARCHIVE, CPFLUSH
Active Log(s): LOG_FILE
```

CP—Consistency Points

Within the Current log file header section is a group of numbers preceded by the label CP. These numbers, like the numbers following Begin and End are divided into three groups. The middle of the three groups refers to the block marking the last consistency point. This consistency point contains a list of all open transactions and open databases at that time.

In the sample output, the block marking the consistency point is 1873. CPs shorten the recovery window after a system goes down. Instead of reading from BOF to EOF, the last CP is read and recovery is begun from there.

Interpreting Status

Within the Current log file header section is an area called Status, which in this case states: "ONLINE,ARCHIVE,CPFLUSH." The status provided here is of the logging and recovery systems. ONLINE indicates that everything is fine. ARCHIVE indicates that archiving is currently taking place. CPFLUSH indicates that a consistency point is taking place.

Using the Lists of Active Databases and Transactions

The bottom half of the logstat output is two sections, the List of active databases and the List of active transactions. You can use the information in these sections to determine which databases are open and active. This is useful if you must shutdown the installation, since it is good policy to make sure that all databases are closed before shutting down an installation. Many times, knowing which databases are open and active allows you to determine whom to notify of the impending shutdown.

To determine which databases are currently active:

1. Look first at the section titled List of active databases in the logstat output. For each database listed, the database's ID number, its name and owner and its status appear on one line. (Note that the first entry listed will always be owned by \$ingres.)
2. Make a note of the ID number of each database listed. These numbers are used in the List of active transactions in the logstat output to identify the database associated with each transaction.
3. Compare the database ID numbers to the entries following the heading Database in the listings of active transactions. If you find a match, it means that the database associated with that ID is currently active.

For example, in the sample output, the second database shown in the output is testdb owned by test. The ID for this database is 00280005. In the List of active transactions, the transaction listed belongs to Database: 00280005. The status of this database (testdb) is ACTIVE,PROTECT,JOURNAL. If an installation shutdown was pending, you could then inform the testdb's owner, test, of the impending shutdown.

mkrawarea


A black rectangular button with rounded corners and a slight 3D effect, containing the word "UNIX" in white, uppercase, sans-serif font.

Sets up a raw area file in a UNIX installation.

Syntax

mkrawarea

Description

The mkrawarea command is run at install time from the root login to create a raw database area and link it to a character special file. 

mkrawlog


A black rectangular button with rounded corners and a slight 3D effect, containing the word "UNIX" in white, uppercase, sans-serif font.

Sets up a raw log file in a UNIX installation.

Syntax

mkrawlog *[-dual]*

Description

The mkrawlog command is run at install time from the root login to create a raw log file for the transaction log. The command is optionally run after installation to set up a raw log file for the dual log. 

rcpconfig

Configure or shut down the logging and locking systems.

Syntax

```
rcpconfig [[-init | -init_log | -init_dual [-node nodename]] |
          -force_init | -force_init_log | -force_init_dual |
          -enable_log | -enable_dual | -disable_log |
          -disable_dual | -shutdown | -imm_shutdown][[-silent]
          [-help]]
```

Description

The rcpconfig utility can be used to control the state of the logging system. It is designed for use in system maintenance for special purpose control functions. You must be an Ingres system administrator to use this utility.

The flags specify the following functions:

Flag	Description
init	Initializes both transaction log files. This may be done only when the installation is offline. For clustered installations only, the node flag may optionally be used.
init_log	Initializes the primary transaction log file. This may be done only when the installation is offline. For clustered installations only, the node flag may optionally be used.
init_dual	Initializes the dual transaction log file. This may be done only when the installation is offline. For clustered installations only, the node flag may optionally be used.
node <i>nodename</i>	Is used to query a specific node. A <i>nodename</i> is valid only in a clustered installation.
force_init	Forcibly initializes both transaction log files. This may be done only when the installation is offline, but after allocating shared memory (csinstall).
force_init_log	Forcibly initializes the primary transaction log file. This may be done only when the installation is offline.

Flag	Description
force_init_dual	Forcibly initializes the dual transaction log file. This may be done only when the installation is offline.
enable_log	Enables the primary transaction log file. This may be done only when the installation is offline.
enable_dual	Enables the dual transaction log file. This may be done only when the installation is offline.
disable_log	Disables the primary transaction log file.
disable_dual	Disables the dual transaction log file.
shutdown	Gracefully shuts down the installation. It waits for any currently executing transactions to finish and cleans up the logging and locking system prior to shutdown.
imm_shutdown	Immediately shuts down the installation. It does not wait for currently executing transactions to complete. Transaction recovery will be required when the installation is restarted.
silent	Sets the program exist status to TRUE or FALSE according to the results of the operation.
help	Displays syntax online.

Note: Each flag on a command line must be preceded by a hyphen -.

Rcpconfig is called by ingstart and ingstop to configure or shut down the archiver and recovery processes for your installation. You will normally not run the rcpconfig utility directly. Instead, use ingstart to reconfigure the locking and logging system and ingstop to shut down the entire Ingres system, including the locking and logging system.

rcpstat

Displays the status of the logging system.

Syntax

```
rcpstat [[[-exist | -format | -enable [-dual]] | -online |  
         -transactions | -sizeok | -csp_online  
         [-node nodename]] | -any_csp_online  
         [-silent] [-help]
```

Description

The rcpstat utility can be used to query the state of the logging system. It is designed for use in system maintenance and is not normally employed by users.

The flags specify the following functions:

Flag	Description
exist	Show whether the primary (default) or dual (dual flag) transaction log file exists.
format	Show whether the primary (default) or dual (dual flag) transaction log file is formatted.
enable	Show whether the primary (default) or dual (dual flag) transaction log file is enabled.
dual	Flag option for the exist, format, or enable options to specify the dual log.
online	Show whether the logging system is online or offline.
transactions	Show if the logging system has recoverable transactions in the transaction log file.
sizeok	Show whether the primary and dual transaction log files are the same size.
csp_online	Show if, in a clustered installation, the Cluster server (CSP) is online on this node.
node <i>nodename</i>	Is used to query a specific node, on any of the preceding flags. A <i>nodename</i> is valid only in a clustered installation.
any_csp_online	Display if, in a clustered installation, the Cluster server (CSP) is online on any node.
silent	Sets the program exist status to TRUE or FALSE according to the results of the operation.
help	Displays syntax online.

Note: Each flag on a command line must be preceded by a hyphen -.

regdocs

Register specified files to an Ingres/ICE business unit.

Syntax

regdocs *options filename(s)*

Description

The regdocs utility allows a system administrator to register files to Ingres/ICE in bulk without having to use the Visual DBA utility. With this utility, you can build scripts to aid in the maintenance of an Ingres/ICE web site.

The *filename(s)* parameter specifies one or more files that you want to register. Alternatively, you can use the *-ilistfile* flag to specify the names of multiple files in the specified *listfile*, saving you typing, and input errors while using the command line.

The following table lists valid options for this utility:

Options	Description
-nnode	The node name of the target server.
-oowner	The Ingres/ICE user who owns the file.
-uunitname	The name of the business unit to which the files will belong.
-llocation	The name of the location of the files within the business unit. (A business unit can have more than one location associated with it.)
-tdoctype	The type of document being registered, either pages or facets. p—the document is a page, which is processed by the Ingres/ICE server. This is the default. f—the document is a facet, which is not processed by the Ingres/ICE server.
-fflags	The flags associated with the file that is being registered. e—the file is located in an external location. This flag causes the flags r, p, f, and s to be ignored. g—the file is accessible by any user. r—the file is located in the repository. This flag causes the p, f, and s flags to be enabled. p—the file is preloaded during server start. f—the file is always loaded. s—the file is loaded on demand.

Options	Description
-help	Displays syntax online.
-i <i>listfile</i>	The name of the file that contains the list of files (pages and facets) to be added to the business unit.

rmcmdgen

Utility used to generate the Visual DBA remote command catalogs.

Description

The rmcmdgen utility can only be started by the ingres system administrator, and it has no command line parameters. It generates the objects in the iidbdb database that are needed by rmcmd.

The rmcmdgen utility is normally invoked by the installer.

rmcmdrmv

Utility used to remove the Visual DBA remote command catalogs.

Description

Generally used when upgrading an installation.

rmcmdrmv can only be executed by the ingres system administrator, and it has no command line parameters. It removes (from the iidbdb database) the objects (tables/view/ procedures/dbevents) that are needed by rmcmd.

rmcmdstp

Utility used to stop rmcmd – the remote command process.

Description

rmcmdstp can only be executed by the ingres system administrator, and has no command line parameters. rmcmdstp stops the rmcmd process. **Note:** Normally, rmcmd is stopped by the ingstop process.

syscheck

UNIX

Displays process and system resources and verifies that there are enough resources to run an Ingres installation as currently configured.

Syntax

syscheck [-v] [-ofilename] [-help]

Description

The syscheck utility checks if there are sufficient resources available in your system to run Ingres. If not, syscheck displays the resources needed. If you have enough resources, syscheck prints a confirming message and continues. If it finds that your system does not have enough resources, it prints an error message and exits with an error status.

If you run this command, be sure to run it after you have configured your system since syscheck reads the locking and logging parameters before it checks for resources.

The following table lists valid command parameters and flags:

Parameter	Description
-v	Verbose mode. The command displays messages on all resources, not just the insufficient ones.
-ofilename	Specifies that syscheck output is to go to the specified <i>filename</i> .
-help	Displays syntax online.

Ingstart calls syscheck after the system is configured and before starting the servers, so you will ordinarily have no need to call this command directly. However, if system resources have changed since installation, you can use this command to see if you are reaching operating system resource limits that might cause the system to fail. ■

vcbf

The vcbf utility starts up the Configuration Manager, a graphical user interface for configuring your Ingres installation.

Syntax

vcbf

Description

The Configuration Manager displays current values of the server parameters and provides menu and screen selections for changing them.

With the Configuration Manager you can:

- Configure various components of the installation:
 - Name, DBMS, ICE, Net, or Star servers
 - Locking or logging systems
 - Archiver or recovery process
 - Transaction log files
- Select which databases can be accessed by a DBMS server
- Reformat transaction log file(s) and enable/disable dual logging
- Reconfigure protocol accesses for the Net server
- Set a new value of any configuration parameter, or restore the factory default
- Automatically calculate configuration parameters derived from other parameters
- Protect any derived parameter from further change
- Run a system check for sufficient resources on a new configuration
- View a log of all configuration changes

vdba

The Visual DBA utility is a graphical user interface through which the database administrator can administer an Ingres installation.

Syntax

To launch the specified environment at Visual DBA startup, use the `vdba environmentname.cfg` syntax. This syntax requires the environment to have been previously saved with Visual DBA.

vdba *environmentname.cfg*

To start Visual DBA with a list of parameters, use the `vdba /c` syntax:

vdba /c [**maxapp**] [**maxwin**] [**nonodeswindow**] [*windowdesc {windowdesc}...*]

Description

The `vdba` utility is a graphical user interface through which the database administrator can manage a local or remote Ingres installation.

The following table describes `vdba` parameters:

Parameter	Description
<code>/c</code>	Provides a means of invoking Visual DBA with a list of parameters.
<code>maxapp</code>	Maximize the Visual DBA application.
<code>maxwin</code>	Maximize the MDI windows within Visual DBA.
<code>nonodeswindow</code>	Do not display the Virtual Nodes toolbar and window.

Parameter	Description
<i>windowdesc</i>	<p>Type of window to open on startup. Valid values are dom, sql, monitor, and dbevent.</p> <p>Syntax for each <i>windowdesc</i>:</p> <p>dom sql monitor dbevent [<i>nodename</i>][/<i>serverclass</i>] [-u<i>username</i>][<i>objecttype objectidentifier</i>]</p> <p>where:</p> <p>dom sql monitor dbevent Specifies the type of window to be opened</p> <p>[<i>nodename</i>]: (optional) Node where the window is to open. No node designates the local node.</p> <p>[/<i>serverclass</i>]: Optional server class (allows work on gateways)</p> <p>[-u<i>username</i>]: Optional user to impersonate through the - u option</p> <p>[<i>vnode</i>]: Specifies the remote node for which specified windows should be opened. For details, see the Standard Command Line Flags and Parameters section in the “Ingres Commands” chapter.</p> <p>[<i>objecttype objectidentifier</i>]: Places the selection on the corresponding object: In dom and monitor, expands appropriate branches and places the selection of the corresponding object. In sql and dbevent, only a database can be specified (it becomes the active database).</p> <p>Valid values for <i>objecttype</i>: database table view procedure user group role location server</p> <p>Valid values for <i>objectidentifier</i>: serverno (for servers) (in monitor windows) objectname (if not a child branch from a database) dbname/objectname (if child a database). Schema prefixes are acceptable.</p> <p>Note: The last <i>windowdesc</i> becomes the topmost window in Visual DBA.</p>

Examples

Invoke vdba without the Virtual Nodes window, maximized on the screen, with *one* dom window opened for the local node:

vdba /c nonodeswindow maxwin dom

Invoke vdba, maximized on the screen, with both the performance monitor window and a dom window opened for the local node:

vdba /c maxapp monitor, dom

Invoke a saved vdba environment:

vdba gateway.cfg