

Advantage™ Ingres®

Migration 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.

© 2001 Computer Associates International, Inc.

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

Contents

Chapter 1: Introduction

Ingres Versions	1-2
Hardware Issues—Planning, Testing and Implementation	1-3
Application Issues	1-4
Upgrade Types	1-4

Chapter 2: Getting Started

Application Preparation	2-2
New Reserved Words	2-2
Report Writer Syntax Change	2-3
Loading the Ingres Development Installation	2-3
Moving Databases	2-4
Upgrading Databases	2-5
Additional Application Preparation	2-6
UPDATE . . . FROM Semantics Change	2-7
Binary Level Support	2-7
Decimal Constant Semantics Change	2-7
Greater Sensitivity to BYREF Errors	2-8
Journaling ON by Default	2-8
Greater Sensitivity to Arithmetic Errors	2-8
4GL TABLE_KEY Type Conversions	2-9
User-Defined Data Type Changes	2-9
Report Writer Runtime Parameter Errors	2-9
System Administration Preparation	2-9
Shared Library Search Path	2-10
UNIX Kernel Parameters	2-10
Ingres Startup and Shutdown	2-10
ingprenv Replaces ingprenv1	2-11
System Monitoring Shellscripts	2-11

Checkpoint Template Changes	2-11
Archiver Exit Shellscript	2-12
Transaction Log Size	2-12
Backup and Restore	2-12
Testing and Practicing	2-12
OpenROAD 4.0 Image File Formats	2-13

Chapter 3: Unload/Reload Upgrade

Unload/Reload Upgrade Procedure	3-1
Step 1 [Each DB] Create Unload Directories	3-2
Step 2 [Each DB] Run Unloaddb	3-2
Step 3 [Each DB] Bogus User Check	3-2
Step 4 [Each DB Including iidbdb] Optional Checkpoint	3-2
Step 5 Disable User Access	3-3
Step 6 Ingres Shutdown and System Backup	3-3
Step 7 [Each DB] Unload	3-3
Step 8 [Each DB] Optional Statdump	3-4
Step 9 [Each DB] Record infodb	3-4
Step 10 [Each DB] Destroy the Database	3-4
Step 11 Clean iidbdb	3-4
Step 12 Record Ingres Configuration	3-5
Step 13 Ingres Shutdown	3-5
Step 14 Disable Ingres Startup	3-5
Step 15 Preserve Site Modifications	3-5
Step 16 Login Fixups	3-6
Step 17 Record Default Locations	3-6
Step 18 Clean Up Ingres	3-6
Step 19 Create Work Location	3-7
Step 20 Install Ingres	3-7
Step 21 Restore Site Modifications	3-8
Step 22 Configure Ingres	3-8
Step 23 Ingres Net Setup	3-9
Step 24 Start Ingres	3-9
Step 25 [Each DB] Recreate Databases	3-9
Step 26 [Each DB] Extend Databases	3-9
Step 27 [Each DB] Fix FE Reload Script	3-10
Step 28 [Each DB] Reload	3-10
Step 29 [Each DB] Front-End Upgrade	3-11
Step 30 [Each DB] Reapply Optimizer Statistics	3-11

Step 31 [Each DB including iidbdb] Checkpoint	3-11
Step 32 Application Upgrade	3-12

Chapter 4: Upgrade Using Upgradedb

Upgradedb Upgrade Procedure	4-1
Step 1 [Each DB Including Star DDBs] Create Unload Directories	4-2
Step 2 [Each DB Including Star DDBs] Run Unloaddb	4-2
Step 3 [Each DB Including Star DDBs] User Check	4-2
Step 4 [Each DB] Editing the Unloaddb Output	4-3
Step 5 [Each DB Including iidbdb] Checkpoint	4-3
Step 6 Disable User Access	4-3
Step 7 Ingres Shutdown and System Backup	4-4
Step 8 [Each DB] Optional Statdump	4-4
Step 9 [Each DB] Object Cleaning	4-4
Step 10 [Each DB] Record infodb	4-6
Step 11 Clean iidbdb	4-6
Step 12 [Each DB Including iidbdb] Checkpoint -j	4-6
Step 13 Record Ingres Configuration	4-7
Step 14 Ingres Shutdown	4-7
Step 15 Disable Ingres Startup	4-7
Step 16 Preserve Site Modifications	4-7
Step 17 Login Fixups	4-8
Step 18 Record Ingres Settings	4-8
Step 19 Clean Up Ingres	4-8
Step 20 Create Work Location	4-9
Step 21 Install Ingres	4-9
Step 22 Restore Site Modifications	4-10
Step 23 Start Ingres	4-10
Step 24 Upgradedb	4-10
Step 25 Configure Ingres	4-11
Step 26 Ingres Net Setup	4-11
Step 27 [Each DB] Recreate Objects	4-11
Step 28 [Each DB] Reapply Storage Structures	4-12
Step 29 [Each DB] Reapply Optimizer Statistics	4-12
Step 30 [Each DB including iidbdb] Checkpoint	4-12
Step 31 Application Upgrade	4-12

Chapter 5: Upgradedb Problems

Problem 1:	5-1
Problem 2:	5-1
Problem 3:	5-1
Problem 4: Hang with.....	5-2
Problem 5: File Extend Conversion Loop.....	5-2
Problem 6:	5-2
Problem 7: iifile_info View Not Recreated.....	5-3
Problem 8: All to Public Grants Not Created	5-3

Appendix A: UNIX-Specific Shellscript

Appendix B: Features Introduced in Ingres 2.6

User-Visible Language Enhancements	B-1
Row Producing Procedures	B-1
Increase Maximum Size of Character Data Types	B-2
User-Visible DBA Enhancements	B-2
Usermod Utility	B-2
Auditdb Utility	B-2
Copydb Utility	B-3
Raw Location Support	B-3
GatherWrite Threads	B-3
XML Import/Export Utility	B-3
Ingres Journal Analyzer	B-3
Ingres Import Assistant	B-4
Automated Creation of Location Directories	B-4
RMCMDB Enhancements	B-5
Microsoft Transaction Server Support	B-5
Concurrent Rollback	B-5
Internal Performance Enhancements	B-5
Aggregate Sort Nodes	B-5
Composite Histograms	B-6
Optimizer Support for Hash Joins	B-6
Locking System Performance Improvements	B-6
Preallocated RSB/LKBs	B-6
Miscellaneous Locking System Improvements	B-6
Logging System Performance Improvements	B-7
Buffer Manager Performance Improvements	B-7

Operating System Integration	B-7
64-Bit Operating Systems	B-7
Operating System Thread Implementation on Linux	B-8
Ingres/ICE Enhancements	B-8
ICE Development Environment	B-8
JDBC Enhancements	B-8
Support for Unicode	B-9
New Character Sets for Euro Currency Symbol Support	B-9

Appendix C: Features Introduced in Ingres 2.5

Sort Enhancements	C-1
QEF Sort Enhancements	C-2
DMF Sort Enhancements	C-2
Parallel Sort Techniques	C-3
ANSI/ISO Constraint Enhancements	C-3
Large Cache Support	C-4
Dynamic Write Behind Threads	C-4
Partitioned Transaction Log File	C-5
Optimizedb Enhancements	C-5
Miscellaneous Enhancements	C-6
Read-only Database Support	C-6
New SQL Functionality	C-7
Bit-wise Operator Support	C-7
Miscellaneous Functions	C-7
Extended Date Support	C-8
64 Bit File Support	C-8
Large Catalogs	C-8
Row Locking for System Catalogs	C-8
Update Mode Locking	C-9
Value Locking for Serializable Transaction with Equal Predicate	C-9
Case Expression Support	C-9
Expression in Order By	C-9
Expression in Group By	C-10
Query Optimization/Execution Enhancements	C-10
Ingres Star Enhancements	C-10
Ingres 2.5 Net Features	C-10
Ingres/ICE 2.5 Features	C-11
Security	C-11
Session Management	C-12

Storage Management	C-12
Macro Language Extensions	C-12
Visual DBA Features	C-12
Ingres 2.5 Replicator Enhancements	C-13
Generic Replicator Server	C-13
Increased Replicator Concurrency	C-13

Appendix D: Features Introduced in Ingres 2.0

Variable Page Size	D-1
New Page Format for Larger Page Size	D-2
Larger Tuple Support	D-2
Distributed Multi-Cache Management	D-2
Enhanced Performance of Locking/Logging and Buffer Manager	D-3
Interval Based Deadlock Detection	D-3
Row Level Locking and Transaction Isolation Levels	D-3
Alter Table Support	D-4
Async I/O Support	D-5
Parallel Backup and Restore	D-5
Parallel Checkpointing to Disk	D-5
Parallel Checkpointing to Tape	D-6
Parallel Rollforwarddb from Disk	D-6
Parallel Rollforwarddb from Tape	D-6
Fast Load Support	D-6
R-tree Support: A Spatial Index for Ingres 2.0	D-7
Spatial Data Types and Operators	D-7
Statement Level Rules	D-7
Temporary Tables as Procedure Parameters	D-8
Other Optimizer Enhancements	D-8
Operating System Thread Support	D-9
Table Cache Priorities	D-9
Transaction Access Mode	D-10
Soundex Function	D-10
Ingres 2.0 Star Features	D-11
Ingres 2.0 Net Features	D-11
Protocol Bridge Support	D-11
Data-Stream Compression Support	D-12
SNA Duplex Support	D-12
DECNet/OSI Support	D-12
Ingres 2.0 OpenAPI	D-13
Multi-Threaded OpenAPI	D-13

Introduction

Take advantage of the enhanced features and functions of Ingres. By going to the next level of Ingres, you can obtain superior support services and reap the benefits associated with running the latest version of this Computer Associates product.

This guide, together with the documentation set included on the Ingres CD, will assist in the planning and execution of a successful upgrade. The most important part of any upgrade is to prepare a detailed plan. With detailed planning, potential problems will become evident. Allowing time to determine their cause can help you avoid problems. Problem prevention is the key to any upgrade.

Items as simple as how long to complete a backup and how to verify that the data is complete and secure need to be in the plan. Testing the plan, preferably with a copy of the production system data, will focus attention on any areas that might cause problems during the actual upgrade of the production system. Implementation of the plan should not start until preliminary testing is complete.

The best strategy for doing an upgrade is to implement any compatibility fixes in the current environment first. Once the databases and applications are Ingres ready, test them in that environment, practice the upgrade, and then do the real thing. Do not use any of the new features until after the upgrade is successful. This reduces the number of variables at each step to a minimum and thus maximizes the chances of success.

Ingres Versions

The following chart shows a matrix of Ingres versions. On the left side is the current version. Along the top is the target version. The intersections contain the available upgrades. For example, upgrading from OpenIngres 1.2/01 to Ingres 2.6 can be done by using the unload/reload method or by using the upgradedb method as defined in this document.

To From	Ingres 2.0	Ingres 2.5	Ingres 2.6
Pre Ingres 6.4	Unload/Reload	Unload/Reload	Unload/Reload
Ingres 6.4	Unload/Reload Upgradedb	Unload/Reload Upgradedb	Unload/Reload Upgradedb
OpenIngres 1.x	Unload/Reload Upgradedb	Unload/Reload Upgradedb	Unload/Reload Upgradedb
OpenIngres 2.0	Unload/Reload Upgradedb	Upgradedb	Upgradedb
Ingres 2.0		Unload/Reload Upgradedb	Unload/Reload Upgradedb
Ingres 2.5			Upgradedb

Note: You can use unload/reload in moving from Ingres 2.0 to a higher version, but it is not necessary to do so.

Included as appendixes in this document are Ingres 2.6, Ingres 2.5, and Ingres 2.0 new features. The information is provided so that *after* the upgrade has been completed and has been running successfully for a number of days, you can consider making use of the new features within the release.

Hardware Issues—Planning, Testing and Implementation

For a safe and orderly upgrade, at least four Ingres installations are needed:

- Production
- Development
- Installation for testing the upgrade
- Ingres development installation for preparing and testing applications

These four installations do not need to be on four separate machines. Four machines can be used, separating the Ingres 6.4 and Ingres development machines. However, this arrangement could be counter-productive as it could lead to considerable back-and-forth between the Ingres 6.4 and Ingres installations.

If possible, keep the Ingres installations away from the production machine. If there is no hardware available for the above installations, consideration must be given to temporarily getting additional hardware.

The recommended minimum hardware setup is three machines:

- Development
- Test
- Production

It is possible—but not recommended—to use two machines:

- Development
- Production

It is possible to do everything on less hardware, but it is more difficult and carries greater risks.

Tip: Expect to do no Ingres development while practicing the Ingres upgrades.

Note: There is no remote installation procedure for Ingres. If the machine does not have local media support (CD-ROM or tape), such support will need to be arranged. Otherwise, it will be necessary to copy the distribution files from wherever the CD-ROM or tape drive is situated.

Tip: Back up all data before starting.

Application Issues

Take an application and database inventory before starting, and make sure that every application can be rebuilt. If an application is found that cannot be rebuilt, test it under Ingres as soon as possible. It may be possible to run the application against an Ingres installation and database, but this only applies if that application has no upward compatibility issues, for example, reserved words. You may find yourself completely recreating that application, or doing without it.

If active application development is underway, you need to plan how to coordinate new development with Ingres compatibility. Generally, try to synchronize the test and live Ingres upgrades with an appropriate time in the application life cycle. For example, one site addressed this issue by synchronizing Ingres compatibility with a code release. Then, development was converted to Ingres, while an Ingres 6.4 “bug fix” installation was maintained on a different machine.

Upgrade Types

There are two options for doing production upgrades:

- The unload-reload method
- The upgradedb utility

The two can be mixed, upgrading some databases while reloading others. Unless a cold Ingres re-install is done, iidbdb and imadb will be upgradedb even if it is decided to unload-reload the user databases.

The upgradedb utility allows for a fast, in place, upgrade path for an Ingres database, with no additional disk space requirements. Preparing for a safe and reliable upgradedb can take some time. Older versions of upgradedb have some specific issues that must be accommodated.

A database unload/reload ensures a clean start with a fresh database. Depending on the kind of tables structures, additional disk space may be needed in order to do the unload and reload; this could be as large as three to five times the space of the database that is to be upgraded. For instance, compressed tables with wide CHAR or VARCHAR columns can expand substantially when unloaded. The unload/reload process takes longer than upgradedb, thus increasing the downtime of the production system. However, an unload/reload ensures a cleaner final installation.

A database that has been running for years, perhaps surviving a number of system crashes and hardware failures, could have suffered hidden damage that might confuse upgradedb. For example, a database that is used by a small department or group of people may not be maintained as well as a production database. To continue this example, such a database may have worktables owned by a user who no longer exists or missing table data files.

The upgrade procedure discussed in this document is designed to detect and resolve as many issues as possible, but it cannot correct for missing data files. If consideration was being given to rebuilding the installation and/or databases, upgrade time might be a good time to do it.

UNIX

The assumption is that the development computer will support both current and proposed Ingres installation: Here are the basic steps of how to installing Ingres on the development machine.

1. Create a new Ingres directory in a location with sufficient disk space. For the purposes of this example the directory will be called `/ing20/ingres`.

2. Execute the following commands:

```
mkdir /ing20/ingres
```

```
chmod 755 /ing20/ingres
```

3. Create two scripts called, for example, “set64” and “set20”; these are to be used to set the environment to Ingres 6.4 and Ingres 2.0 respectively. Here are example scripts for the C Shell. They may need to be adjusted to make them installation specific. For instance, the PATH settings may be different, and LD_LIBRARY_PATH may be named LIBPATH or SHLIB_PATH, depending on the platform.

```
set64:
setenv II_SYSTEM /ing64/ingres
set path=(. /usr/local/bin /bin /usr/ucb /usr/sbin /usr/openwin/bin
$II_SYSTEM/ingres/bin $II_SYSTEM/ingres/utility /usr/ccs/bin)
set inst=`ingprev1 II_INSTALLATION`
setenv LD_LIBRARY_PATH /usr/lib:/usr/openwin/lib
set prompt=`whoami`. `uname -n`[$inst]% "
echo "Switching to Ingres 6.4 [$inst] installation"

set20:
setenv II_SYSTEM /ing20/ingres
set path=(. /usr/local/bin /bin /usr/ucb /usr/sbin /usr/openwin/bin
$II_SYSTEM/ingres/bin $II_SYSTEM/ingres/utility /usr/ccs/bin)
set inst=`ingprev II_INSTALLATION`
setenv LD_LIBRARY_PATH /usr/lib:/usr/openwin/lib:$II_SYSTEM/ingres/lib
set prompt=`whoami`. `uname -n`[$inst]% "
echo "Switching to 2.0 [$inst] installation"
```

If required, define aliases in the C shell or shell functions in the Bourne/ksh to source the set64 and set20 scripts.

For instance:

```
alias set64 source ~/ingres/set64
alias set20 source ~/ingres/set20
```

Use the “set20” alias to set the Ingres environment, and cd to \$II_SYSTEM/ingres. Follow the installation instructions to install Ingres.

Note: Do not use the same data, checkpoint, journal, dump, or log directories as for the Ingres 6.4 installation, though they can be on the same disks. ■

If you are new to Ingres, take the time to investigate the new Ingres management facilities, for example, CBF. Also, note that Ingres looks different in a “ps” listing. On a platform that supports OS threads or asynchronous I/O, there are no I/O slaves. The DMFRCP process, the recovery server, has been replaced by another IIDBMS process.

Regardless of which installation method you use, it is important that the 1.x, 2.0, or 2.5 databases be structurally sound before you begin. The following procedures are recommended:

1. Run **modify** on all your database tables.
2. Run **sysmod** `<database>` on all databases.
3. Run **verifydb** `-mreport -sdbname <database> -odbms_catalogs`
4. Run **infodb** `<database>` to verify the status is VALID.
5. Take an offline checkpoint of each database.

Application Preparation

Some applications created under Ingres 6.4 will run unchanged under Ingres, but this should not be relied upon. There are changes in Ingres 2.0 that may require application changes; in general, none of these changes are particularly difficult to implement.

The following issues, that is, new reserved words and Report Writer syntax change, should be checked before moving the applications and databases to the Ingres development installation.

New Reserved Words

Check for and fix reserved word conflicts as a database cannot be built on an Ingres installation until word conflicts are corrected. Additionally, check for reserved word conflicts in dynamically-created tables and views in application code.

Ingres has a number of additional reserved keywords, mostly for support of the SQL additions implemented by Ingres 2.0. If words like *level*, *key*, or *comment* are used as column names, it will be necessary to change them.

See the “Keywords” appendixes in the *SQL Reference Guide* and in this guide for a complete list of Ingres reserved words.

Report Writer Syntax Change

In order to support new Report Writer syntax, a space is required after all dot-commands. For example “.NL3” must be changed to “.NL 3”. The following “sed” commands can be used to fix such occurrences automatically:

```
sed -e 's/\([<space><tab>]\.[a-zA-Z][a-zA-Z]*\)\([0-9]\)/\1 \2/' foo.rw | \
```

```
sed -e 's/^\([a-zA-Z][a-zA-Z]*\)\([0-9]\)/\1 \2/' >newfoo.rw
```

The old and new files (foo.rw and newfoo.rw) should be compared to ensure that only the expected changes have occurred, for example an unwanted “fix” to a literal string.

An alternative to alter Report Writer files is to sreport them into a database, then copyrep the reports back out.

Loading the Ingres Development Installation

This step is where an initial database and application checkout occurs.

Create a copydb script for each user who has tables in the production system, and create the necessary copy scripts for ABF and Vision applications, and so on. Run these scripts into the development Ingres installation. At this point, no data need be loaded into the databases, since there will probably be more work to do before the databases and applications load cleanly.

Once the databases and applications load cleanly, a subset of the data can be loaded, this will highlight any problems with data loading.

If the upgrade is from Ingres 6.4/04 or an earlier release, it is necessary to resolve “trailing quote missing from copyapp/copyform output.” Ingres 6.4/04 and earlier releases of Ingres would omit the trailing quote from format strings when copying out an ABF/Vision form. This would occur with both copyform and copyapp out. The solution is to manually fix the output files before copying the forms back into Ingres.

Moving Databases

Procedures should have been created for moving between the Ingres 6.4 development environment and the Ingres environment. If they do not exist create them as described previously in the document:

1. “set64” and “cd” to a directory with enough space to hold the data, allow for the Ingres System catalogs.
2. Create a directory for each database that is to be exported.
3. “cd” to the directory for the database that is to be exported.
4. Execute unloaddb against the Ingres 6.4 database to be unloaded.
5. Execute unload.ing to export the front-end catalogs and data.
6. Edit the cp_ingre.in file and remove the line:

```
\include /ing64/ingres/files/iiud64.scr
```

(Directory paths may be different.)
7. “set20” to the Ingres installation.
8. Create the database there, but without any front-end catalogs:

```
createdb databasename -f nofeclients
```
9. If the Ingres database name is not the same as the Ingres 6.4 database name then edit the reload.ing script.
10. Execute reload.ing for that database.

Tip: When using reload.ing, capture the output to a file in case any errors occur, on UNIX this can be done using “tee”:

```
reload.ing |& tee /temp/reload.log
```

At this point, the front-end catalogs in the Ingres database are in the Ingres 6.4 format. To get them into Ingres form, run:

```
upgradefe <databasename> INGRES
```

The above assumes that you intend the data to be copied from the Ingres 6.4 development database to Ingres as well as the catalogs. If the data is not wanted, the scripts can be edited so that unloaddb does not do the COPY INTO of some or all of the tables.

Be sure to review the output of the reload. If there are any reserved word conflicts, these should show up in the reload. Correct any problems on the Ingres 6.4 side and try again.

If there are Ingres/Star databases, unloaddb the CDB, (the coordinator database) it usually starts with ii. The unloaddb of the CDB will unload any locally stored tables that do not exist in other local databases.

Then unloaddb/star on the DDB, the DDB is the distributed database, usually accessed by ddbname/star. The unloaddb of the DDB unloads registrations and distributed view definitions.

Upgrading Databases

You can unload your database from Release 6.4/1.x/2.0/2.5 and reload into Ingres 2.6, or run the upgratedb utility to upgrade your database. Make sure that the new 2.6 reserved words are not used in your database or application.

Note: Unloading and reloading your database is not a recommended way of upgrading your database and should only be used as a last resort. There are additional complications with this upgrade method when upgrading to 2.6 because of the addition of *system_maintained* as a key word. The Ingres system catalog ii_atttype contains a column named *system_maintained* in releases before 2.6. In order to avoid problems with the new keyword, this column was renamed to *sys_maintained* in Ingres 2.5 and Ingres 2.6.

Upgrading from Earlier Releases

Note: All checkpoints taken with previous Ingres versions are not compatible with Ingres 2.6/0201. Checkpoints of the iidbdb are strongly recommended at frequent intervals. Generally, these checkpoints require little processor overhead. For table-level rollforwarddb, the -noblobs option has been removed as it makes the table physically inconsistent and unusable. Rollforwarddb no longer supports the -noblobs option.

When upgrading from a release older than 2.0, the database "imadb" is not created. After the upgrade has been completed, create this database by hand by executing the following commands as the user ingres:



```
rmcmdstp
createdb -fnofclients -u$ingres imadb

sql -u$ingres imadb < %II_SYSTEM%\ingres\vdba\makiman.sql >
%II_SYSTEM%\ingres\vdba\makiman.out

rmcmdgen

ingstart -rmcmd
```

When upgrading from a release older than 2.0, the database "imadb" is not created. After the upgrade has been completed, create this database by hand by executing the following commands as the user ingres:

UNIX

```
rmcmdstp  
  
createdb -fnofclients -u\$ingres imadb  
  
sql -u\$ingres imadb < $II_SYSTEM%/ingres/vdba/makimau.sql >  
$II_SYSTEM%/ingres/vdba/makimau.out  
  
rmcmdgen  
  
ingstart -rmcmd
```

The `unloaddb` command in releases prior to 2.5 will create a `copy.in` script that creates the catalog `ii_atttype` with a column named `system_maintained`. When running `reload.ing`, this script will fail because of the keyword restriction. In order for the reload to work, the `copy.in` script must be modified so that it creates the `ii_atttype` table with a column named `sys_maintained`. If you wish, you can edit the `copy.in` file and make this change by hand, otherwise there is a script in the Ingres `sig` directory that will perform the task for you. Before running the script, you should ensure the your current directory is the unload directory (that is, the directory that contains the `copy.in` file), you should execute the following command at the command line:

UNIX

```
$II_SYSTEM/ingres/utility/iiunloadfix
```

Windows

```
%II_SYSTEM%\ingres\utility\iiunloadfix
```

If you are upgrading a 1.x database, any tables having long varchar, long binary, or long spatial data must be unloaded under 1.x and reloaded into Ingres 2.6. The format of blob extension was changed for Ingres. The minimum transaction log file size was increased for Ingres 2.5 and beyond. If you have an existing transaction log file that is less than 16 MB, you must destroy the transaction log and recreate it with a size of at least 16 MB before upgrading to Ingres 2.6.

Additional Application Preparation

After successfully creating databases and applications in the Ingres development installation, check for the following application issues.

UPDATE . . . FROM Semantics Change

The “ambiguous replace” test in Ingres 6.4/05 and earlier, allowed the update as long as each target row was being updated with an unambiguous value. Ingres 6.4/06 and higher releases tests for multiple FROM rows and generates an ambiguous replace error message even if all the FROM rows generate the same replacement value.

For example:

```
UPDATE    table_1
FROM      table_2
SET       column_3 = 3;
```

Ingres 6.4/05 and earlier would allow this, even though there is no “WHERE” qualification joining the tables, since the replacement value was non-ambiguous. In later releases of Ingres, an “ambiguous replace” error message displays.

The best way to handle this, is to review all applications for ambiguous updates and change them to use EXISTS or IN, instead of a join. If this is not feasible, the original UPDATE . . . FROM handling can be restored by to set the DBMS parameter “ambig_replace_64compat” to ON within CBF.

Binary Level Support

Ingres 2.6 provides support for 1.x, 2.0, and 2.5 applications at the binary level. You can run 1.x, 2.0, and 2.5 applications that access an Ingres 2.6 server without rebuilding the application. However, we recommend that you rebuild all 1.x, 2.0, and 2.5 applications with Ingres 2.6.

Note: 6.4 binaries can be run using Ingres Net. If the binaries need to be run locally, they must be rebuilt. If you do not rebuild the executables, they may not be able to read the error message file.

Decimal Constant Semantics Change

With the introduction of the DECIMAL data type, fixed-point literals such as 1.0 are now considered DECIMAL, rather than FLOAT. Normally, this does not matter, as Ingres does appropriate type conversions. It is important when doing a CREATE TABLE . . . AS SELECT with a constant in the SELECT result list.

For example:

```
CREATE TABLE table_1
AS SELECT column_1, column_2, column_3 = 1.0
FROM table_2;
```

In Ingres 6.4, the column_3 is created as FLOAT8. In Ingres it is created as a DECIMAL(2,1) column, this may result in overflow in an application.

The best way to handle this is to examine uses of fixed-point constant usage in applications and change them to floating point constants, or add an explicit FLOAT8 type conversion.

A less thorough but easier alternative is to set the environment variable II_NUMERIC_LITERAL to FLOAT:

```
setenv II_NUMERIC_LITERAL FLOAT
```

Ingres then interprets fixed-point constants as floats rather than decimals. If you decide to use II_NUMERIC_LITERAL, it will be necessary for *every* user of the application(s) to set II_NUMERIC_LITERAL within their environment.

Greater Sensitivity to BYREF Errors

Ingres 6.4 4GL programs were insensitive to length and type errors when returning BYREF values to a calling program. Ingres is more sensitive to return values that are too long or are of the wrong type. In some cases, this can result in programs aborting and/or segmentation violations. The cure is to ensure that the called and calling routines return values of compatible length and type.

Journaling ON by Default

In Ingres 6.4, if a database were journaled, newly-created table would not be journaled unless WITH JOURNALING was explicitly stated.

In Ingres, journaling is on by default. This means that if an application creates temporary tables, those tables will be journaled; this may consume more system resource, resulting in Ingres applications running more slowly than expected.

By changing the CBF parameter “default_journaling,” it is possible to turn default journaling off. Alternative options are to issue a SET NOJOURNALING statement at the beginning of an application, create temporary tables WITH NOJOURNALING, or to use session tables.

Greater Sensitivity to Arithmetic Errors

Ingres 6.4 ignores a number of arithmetic error conditions (such as floating point overflow, divide-by-zero). Ingres correctly reports arithmetic errors on all platforms. If an application generates arithmetic exceptions when tested with Ingres, it is probable that the application had problems in Ingres 6.4 that were not reported. The application needs to be corrected.

4GL TABLE_KEY Type Conversions

Conversion of 4GL VARCHAR variables to the TABLE_KEY type gives length errors. Avoid this by converting to char first:

```
TABLE_KEY(CHAR(varcharVariable))
```

User-Defined Data Type Changes

If you are using Object Management Extension to declare user-defined data types in the server, be aware of some changes in calling sequences. See the Ingres *Object Management and Extension User Guide* for details.

Report Writer Runtime Parameter Errors


 UNIX

If string parameters that contain quotes are passed to Report Writer, runtime errors may occur. This may be caused by a change to the UNIX command parameter control file utexe.def.

If this error occurs, it is possible to change the command parameter back to the Ingres 6.4 utexe.def settings:

Edit \$II_SYSTEM/ingres/files/utexe.def

Search for the string: '(%S)'

Change it to: param '(%S)'

Save the file, retest, and see whether the error still occurs. ❏

Many of the changes required for Ingres are backward compatible with Ingres 6.4. Make application changes in the Ingres 6.4 installation, and bring them forward to the Ingres installation for testing. In this way, it is not necessary to freeze application development while preparing for Ingres.

At this stage, resist the temptation to make Ingres-specific application changes. While an outer join or a session temp table may do wonders for performance, there is plenty of time to add performance enhancements *after* the upgrade.

System Administration Preparation

Some Ingres upgrade issues involve system or Ingres administration. Coordinate these changes with the system administrator.

Shared Library Search Path

On many UNIX platforms, Ingres uses shared libraries. Since there is no default installation directory for Ingres, it is necessary to tell applications and tools where Ingres has been installed so that the shared libraries can be found. This is done in one of two ways:

- Depending on the platform set, for all users who access any Ingres programs or applications, set the environment variables `LIBPATH`, `LD_LIBRARY_PATH`, `SHLIB_PATH`, to include the Ingres library directory, `$II_SYSTEM/ingres/lib`.

Failure to have `LD_LIBRARY_PATH` set will result in an error message:

```
ld.so.1:          /ing20/20/ingres/bin/tm: fatal:
libframe.1.so:   open failed: No such file or directory
```

In Ingres 6.4, it does not matter if `$II_SYSTEM/ingres/lib` is included in `LD_LIBRARY_PATH`.

- Link the Ingres library directory to `/usr/lib`.

For example:

```
in -s /ing20/ingres/lib/libframe.1.so /usr/lib
```

This does not require application wrappers or user environment changes. The disadvantage of this approach is that the system administrator has to link individually each Ingres library, and after a subsequent Ingres upgrade, it is necessary to check the validity of these links.

UNIX Kernel Parameters

Review the UNIX kernel parameter settings, particularly the maximum shared memory size. It may be necessary to increase the size of a shared memory segment because Ingres builds a larger shared memory segment for locking and logging than Ingres 6.4. A 40 megabyte shared memory segment will accommodate most migrated installations. Each platform has its own way of modifying the shared memory limits, discuss this with the system administrator or read the platform-specific information in the Release Notes.

Ingres Startup and Shutdown

Ingres uses different startup and shutdown commands: `ingstart` and `ingstop` instead of `iistartup` and `iishutdown`. If there are shell scripts that start and stop Ingres, it will be necessary to change them. Verify the changes in the development Ingres installation and have the revised scripts ready for the production environment at time of upgrade.

ingprenv Replaces ingprenv1

In Ingres 6.4, the `ingprenv1` command displayed one Ingres environment variable. This command no longer exists in Ingres. Shell scripts that use `ingprenv1` will need to be changed. It is possible to recreate `ingprenv1` as follows:

```
echo 'exec $II_SYSTEM/ingres/bin/ingprenv $*' >/usr/local/bin/ingprenv1
chmod +x /usr/local/bin/ingprenv1
```

System Monitoring Shellscripts

Production systems may have tools to provide the system administrator with early warning of Ingres problems. If these tools have been developed in-house, they will need to be reviewed for Ingres. Some of things to check for are:

- Are there still IO Slaves on the UNIX platform?
- Has Lockstat or Logstat output changed?
- Have `II_RCP.LOG` and `II_ACP.LOG` been renamed to `iircp.log` and `iiacp.log`?
- Are Ingres 6.4 parameters that were held in `rundbms.opt` now held in `config.dat`?
- If you are using a commercial monitoring tool, contact the vendor to see if an upgrade is needed to support Ingres.

Checkpoint Template Changes

The Ingres checkpoint template file, `cktmpl.def`, is similar in structure to the Ingres 6.4 version, but has been expanded and is therefore not compatible. If any changes were made to the Ingres 6.4 checkpoint template, those changes will need to be recreated for Ingres. The “Backup and Recovery” chapter of the *Database Administrator's Guide* discusses the new format of the checkpoint template file.

The Ingres development installation can be used to develop and test the new `cktmpl.def`.

If the Ingres 6.4 checkpoint template had been modified to do parallel multiple location checkpoints, be aware that Ingres now directly supports parallel multiple location checkpoints; this will simplify the checkpoint processing.

Archiver Exit Shellscript

Ingres comes with a sample Archiver exit script, called `acpexit.def`. If the Ingres 6.4 `acpexit` script has been customized, these changes will have to be carried over to the Ingres installation.

See the “Customization Options” chapter in the *System Administrator’s Guide* for information about the `acpexit` script.

Transaction Log Size

Ingres uses less transaction log file space than Ingres 6.4. A few operations may use more, (for example `MODIFY TO MERGE`).

The force-abort limit cannot be set as close to log-full as was possible in Ingres 6.4.

If your Ingres 6.4 transaction log was just large enough, it may be advisable to increase the size to allow for reserve space due to an improved transaction recovery algorithm.

Backup and Restore

As part of the upgrade, it is important to have a system backup. If something goes wrong, it will be necessary to restore from that backup. Make sure that the system administrator knows *how* to take a complete system backup and how to restore that backup. Verify that the backup is readable and that it is stored in a secure location.

Testing and Practicing

As changes are made to the application for Ingres compatibility, bring the changes over to the development Ingres installation and test.

The amount of testing done should at the least test the mission critical application functions. It would be sensible to test with a representative amount of data.

If time and resource is available, the more testing that is done the better. Since resources are not often available, it may be more cost effective to plan to react to bugs and incompatibilities instead of trying for 100 percent test coverage.

The key word here is *plan*. Have resources in place to fix bugs for a period after the upgrade. Avoid new feature development. Have abbreviated change control procedures so that problems can be resolved quickly if a problem occurs.

Test your system administration procedures, too. Crash test the Ingres installation when it is busy, pulling the power plug, or issuing a kill -9 to crash the servers. Make sure that recovery occurs correctly. Do at least one rollforwarddb of the most important databases and make sure it works.

Run an upgrade as early as possible in the conversion cycle. The first upgrade test should be done on a test Ingres 6.4 installation. Ideally, upgrade tests should be done more than once, so an isolated environment is desirable. Take notes on what went wrong or what should be done in a different way. Keep doing practice upgrades until no more problems are encountered. Give the annotated upgrade procedure to someone who can verify the upgrade plan.

It is not necessary to have a complete live data set for the practice upgrade. At least one of the practices should be on a full data set so that you will have an indication of how long the upgrade will take.

As the date for the live upgrade approaches, freeze all changes, delete *all* application objects and images from the development Ingres installation and re-image everything. Subject this refreshed copy to at least a critical functions test.

This build will be used for the live upgrade.

OpenROAD 4.0 Image File Formats

The image (.img) file format for OpenROAD images has changed between OpenROAD 3.5 and OpenROAD 4.0. Image files built using the two releases are not compatible. Attempting to run an image from a different version will produce unpredictable results. OpenROAD 3.5 applications should be re-imaged for use under OpenROAD 4.

Unload/Reload Upgrade

The unload/reload upgrade avoids the `upgradedb` program (except for `iidbdb`), in favor of unloading the Ingres 6.4 databases to flat files, recreating the databases under Ingres and then reloading the databases. This approach has the advantage of starting with a clean Ingres installation. This approach needs more disk space and time to perform the conversion.

The hardest part of the unload/reload upgrade is dealing with the front-end catalogs. These are unloaded in Ingres 6.4 format, and cannot be loaded into an Ingres database. To circumvent this problem, the Ingres database is created without front-end catalogs. The catalogs are then loaded in the Ingres 6.4 format and upgraded using the `upgradefe` program.

The `iidbdb` is not unloaded and reloaded. That would be a cold install of the entire installation. Instead, `iidbdb` is upgraded with `upgradedb`. This way, your existing users, locations, groups, roles, and the like, are preserved. They do not need to be reentered.

If the Ingres installation is already Ingres, unload/reload is simple, given sufficient time and hardware resources.

Unload/Reload Upgrade Procedure

In this procedure, the notation [Each DB] means: “For each database, not including the `iidbdb`: Become the DBA user for that database; `cd` to the unload directory for that database that was created in Step 1; and perform this step.”

If you are using Ingres/Star, include the coordinator database in the list of databases.

Step 1 [Each DB] Create Unload Directories

Create a directory for each database. This directory will be used for holding scripts from the unloaded database and requires large amounts of disk space. As an estimate, the unloaded data is about the same size as the Ingres database; however, compressed data can take up much more space than the Ingres database.

UNIX

```
mkdir /someplace/dbname
```

```
chmod 777 /someplace/dbname
```

Windows

```
md \someplace\dbname
```

Step 2 [Each DB] Run Unloaddb

Run unloaddb against each database. Remember that unloaddb does not unload any data; it simply creates copy-in and copy-out scripts.

Note: For Ingres/Star databases, do a regular unloaddb of the CDB but an unloaddb/star of the DDB.

```
unloaddb dbname <regular db or CDB>
```

```
unloaddb ddbname/star <STAR distributed db>
```

Step 3 [Each DB] Bogus User Check

Examine the unload.ing and reload.ing scripts that unloaddb created. Each script contains one line for each user who owns a database object. Make sure that all the users listed are valid; old databases may contain objects created by users who are no longer with the company.

If obsolete users are found, delete those lines from unload.ing and reload.ing; delete the cp{user}.in and cp{user}.out files; and go into the database and clean out these unwanted objects.

Step 4 [Each DB Including iidbdb] Optional Checkpoint

This step is optional. Checkpoint each database then copy the checkpoint files to tape. Validate the tapes to ensure that they can be read.

Remember to checkpoint the iidbdb.

Step 5 Disable User Access

Disable user access.

Step 6 Ingres Shutdown and System Backup

Shut down Ingres. This *must* be a clean shutdown, leaving non-flushed transactions in the Ingres transaction log.

This can be achieved by shutting Ingres down, restarting Ingres, then shutting it down again. Then check the recovery process log (II_RCP.LOG) for “RCP Shutdown completed normally”.

Take a system backup, using whatever command is appropriate to the platform. Ensure that all Ingres directories: data, checkpoint, journal, dump areas, and the \$II_SYSTEM/ingres directory containing Ingres files and executables are included. Back up the application directories.

Watch out for symbolic links and cross-mounts; make sure that it is real data that is saved and not a symbolic link.

If Ingres is normally started up at boot time, include the root file system in the backup. Alternatively, print a copy of any Ingres boot time startup and shutdown scripts.

For security, perform the preceding procedures twice. Check the tapes after the backups to ensure that the tapes can be read.

After the completion of the backup, restart Ingres.

Step 7 [Each DB] Unload

For each database, run the unload.ing script created by unloaddb. This unloads the database into your unload directory.

Step 8 [Each DB] Optional Statdump

If you have sufficient time to run `optimizedb` within the upgrade plan for a full `optimizedb` against your databases, you can omit this step. Only do this step if there is insufficient time within the upgrade plan for a full `optimizedb` against your databases. Using this shortcut may result in some of the new Ingres metrics not being available.

Dump the existing optimizer statistics:

Run `statdump` with the `-o` flag to a file for each database:

```
statdump -o dbname.stats dbname
```

Step 9 [Each DB] Record infodb

Run `infodb` against each database, saving the output. The output may be needed later. For example, you may need to know whether the database was journaled, the data locations where the database resides, and in what order the locations were configured.

```
infodb dbname >infodb.out
```

Step 10 [Each DB] Destroy the Database

Destroy each database using `destroydb`.

Step 11 Clean iidbdb

Become user Ingres, and run a subset of the Object Cleaning step against the master database `iidbdb`. It is assumed that there are no objects created by users in the `iidbdb`.

```
statdump '-u$ingres' -zdl iidbdb
```

```
sysmod -s iidbdb
```

```
verifydb -mrun -sdbname iidbdb -opurge
```

```
verifydb -mrun -sdbname iidbdb -odbms
```

Messages from `verifydb` should be handled in the same way as in Step 9 (Object Cleaning).

Step 12 Record Ingres Configuration

As user `ingres`, execute the “`showrcp`” command and record the results.

Record the contents of the `rundbms.opt` file in `$II_SYSTEM/ingres/files`.

Use this information as a guide for configuring Ingres. The Ingres installation procedure does not preserve the existing Ingres tuning parameter settings. As the `ingres/files` directory is going to be deleted, save the information.

Step 13 Ingres Shutdown

Ingres 6.4 Shut down Ingres using `iishutdown`.

OpenIngres and later Shut down Ingres using `ingstop`.

Step 14 Disable Ingres Startup

If the machine starts Ingres automatically on bootup, turn auto-starting off.

On most UNIX platforms, there will be a file in `/etc/init.d` or `/sbin/init.d` that does Ingres startup and shutdown; place an “`exit 0`” at the top of that file. This may require root privilege, and the system administrator may need to perform this step.

Make sure that the operating system is correctly configured for Ingres (see the System Administration Preparation section). If a reboot is necessary following parameter changes, it should be done at this time.

Step 15 Preserve Site Modifications

Sites customize the `$II_SYSTEM` directory tree or files within it. The most common customization is to change the `termcap` and keyboard map files in `$II_SYSTEM/ingres/files`. Also, check for local collation sequence files. Ideally, save the original collation definition files; and the compiled files in `$II_SYSTEM/ingres/files/collation` should be saved as well.

Copy any customized files to a safe place, *not* `/tmp` and *not* anywhere in `$II_SYSTEM/ingres` directory. If you are not sure that you can identify customized files, do:

1. Delete all `*.log` and `*.LOG` files from `$II_SYSTEM/ingres/files`.
2. Copy the entire contents of the `$II_SYSTEM/ingres/bin`, `$II_SYSTEM/ingres/files`, and `$II_SYSTEM/ingres/utility` directories somewhere safe.

This is copying more than needed, but the copy can be deleted when Ingres has been running live for a period. It may not be discover that a Vision template or keyboard map file is missing for many weeks after the final conversion.

This tar command copies everything that is needed:

```
cd $II_SYSTEM/ingres
```

```
tar cf - bin files utility | (cd /someplace/safe;tar xf -)
```

Step 16 Login Fixups

If needed, make sure that the Ingres users' login sets LD_LIBRARY_PATH or platform equivalent. Make sure that the users' login does not use ingprev1 or install your ingprev1 substitute. See the System Administration Preparation section in the "Getting Started" chapter. Check all your database owner (DBA) logins to ensure that they are properly set up for Ingres.

Step 17 Record Default Locations

The upgrade runs more smoothly if the Ingres executables, control files, and the Ingres environment variables are deleted. Since this means that the default Ingres settings will need to be reentered, make sure you know what they are.

Ingres 6.4 Copy \$II_SYSTEM/ingres/files/symbol.tbl to a safe area not in the Ingres directory tree.

OpenIngres and later Copy \$II_SYSTEM/ingres/files/symbol.tbl, config.dat, and protect.dat to a safe area not in the Ingres directory tree.

Step 18 Clean Up Ingres

To guarantee a clean environment for Ingres, invoke the following commands as appropriate:

```
cd $II_SYSTEM/ingres
```

Ingres 6.4 **rm -rf bin files lib utility dbtmp1t version.rel admin**
mkdir files
Copy symbol.tbl back into the \$II_SYSTEM/ingres/files directory.

OpenIngres and later **rm -rf bin files lib utility admin**
mkdir files
Copy config.dat, protect.dat, and symbol.tbl back into the \$II_SYSTEM/ingres/files directory.

Step 19 Create Work Location

The Ingres installation asks for an Ingres location for temporary files and sorting. The installation creates the directories if they do not exist. However, the installation procedure may not properly set the protections for the directories, which can lead to `upgradedb` failing when upgrading `iidbdb`. To avoid this, consider creating the work location manually. See the “Using Work Locations” chapter of the *Database Administrator's Guide* for information on placement of your default work location. As user `Ingres`, assume a work location called `/mywork`:

UNIX

```
/mywork:
mkdir /mywork/ingres
mkdir /mywork/ingres/work
mkdir /mywork/ingres/work/default
mkdir /mywork/ingres/work/default/iidbdb
chmod 755 /mywork/ingres
chmod 700 /mywork/ingres/work
chmod 777 /mywork/ingres/work/default
chmod 777 /mywork/ingres/work/default/iidbdb
```

Windows

```
md \mywork\ingres
md \mywork\ingres\work
md \mywork\ingres\work\default
md \mywork\ingres\work\default\iidbdb
```

Step 20 Install Ingres

UNIX


See the Ingres platform-specific installation instructions. Enter the directory locations for the `ii_database`, `ii_checkpoint`, `ii_journal`, and `ii_dump` that were recorded in Step 18 (Record Default Locations).

During the DBMS server setup, it will ask if all databases are to be upgraded, answer **No**. The install procedure automatically upgrades the `iidbdb`. If the upgrade of `iidbdb` fails, see the `Upgradedb Problems` section. It is better to complete the Ingres setup, and then use `upgradedb` to upgrade the user databases.

If a patch needs to be installed for Ingres, the patch may contain a fix for `upgradedb` or installation setup. Do the following:

1. Answer **No** when `ingbuild` prompts for set up of Ingres.
2. Exit `ingbuild`.
3. Install the patch according to the instructions.

4. Rerun `ingbuild`.
5. Select `Current`, then `SetupAll`.

Setup now uses the fixed version. 

Step 21 Restore Site Modifications

If the checkpoint template file `cktmpl.def` has been modified, the modifications may need to be carried forward into Ingres. The `cktmpl.def` from Ingres 6.4 cannot be used in Ingres as the file format has changed. This means that the changes need to be recreated using the Ingres 6.4 `cktmpl.def` as a guide. See the *Ingres Database Administrator's Guide*.

If the archiver exit script `acpexit` was changed in Ingres 6.4, the changes need to be made to the Ingres template (`acpexit.def`) and the file moved to `$II_SYSTEM/ingres/files/acpexit`.

Set the default to the save directory that was created in Step 15 (Preserve Site Modifications), and restore any site-specific files, for example, the local collation sequence files.

Step 22 Configure Ingres

Run Configuration-By-Forms (CBF) and do a first pass configuration for the Ingres installation. Use the `rundbms.opt` and `showrcp` information from Ingres 6.4 as a guideline. For information about CBF and the various tuning parameters, refer to your *System Administrator's Guide*.

Derived parameters are recalculated when values they depend on are changed. If derived parameters are set, they can be “protected” against change.

Ingres 2.0 calculates very large lock and resource limits parameters. Consider reducing these limits to the Ingres 6.4 settings.

On platforms that support OS-threads, do not turn on `async_io`, and do not declare the `II_NUM_SLAVES` Ingres variable. Check the Readme file for your platform for further information about OS-threads.

Ingres supports larger `QEF_SORT_MEM` values than Ingres 6.4. Ingres does not need as much `QSF_MEMORY` as in Ingres 6.4. OS-thread platforms should not reduce quantum. Reducing quantum does not improve performance.

Step 23 Ingres Net Setup

Run netutil to create the vnode definitions for the remote installations. If installation passwords are needed, it is necessary to run mkvalidpw. See the *System Administrator's Guide* or the Readme file for your platform.

If there are NFS client-only installations that have not been set up, run ingmknfs to set them up.

Step 24 Start Ingres

Run ingstart to start the Ingres installation.

Step 25 [Each DB] Recreate Databases

Before creating each database, refer to the infodb output saved in Step 9 (Record infodb). If the database "ROOT" location is not ii_database, it will need to be specified in the createdb command. If the "ROOT" data location is ii_database, omit the -d option.

Recreate each user database, omitting the front-end catalogs. (The front-end catalogs will be reloaded and upgraded in a later step.)

createdb dbname -dlocation -f nofeclients

Note: For any Ingres/Star database, run createdb/star for the DDB. Do not run createdb for the CDB.

Step 26 [Each DB] Extend Databases

Refer to the infodb output saved in Step 9 (Record infodb). If the database was extended to locations other than the default locations, run accessdb as user Ingres and extend the newly-created databases to the same locations. The locations will already exist; it is only necessary to extend the databases to use them.

Step 27 [Each DB] Fix FE Reload Script

Edit the file cp_ingre.in and locate the line:

```
\include /ing64/ingres/files/iiud64.scr
```

The directory path may differ. Delete this line and save cp_ingre.in. Since the database was not created with front-end catalogs, it is not necessary to drop them.

Step 28 [Each DB] Reload

UNIX

Run reload.ing for each database. Tee the reload to a log file so that it can be checked for errors.

Using the Cshell:

reload.ing | & tee reload.log

Note: If using Ingres/Star, reload the CDB and all “real” local databases before reloading the DDBs. ■

If you are reloading data from a 2.5, 2.0, 1.x, or 6.4 databases, and your Ingres 2.6 installation is on a different platform, you must unload your original databases using **unloaddb -c <database>**. Before running reload.ing on the Ingres 2.6 installation, edit the cp_ingre.in file if unloading from 6.4 (or the copy.in file if unloading from 2.5, 2.0, or 1.x), and change the path references.

Change:

\include /<64path>/ingres/files/iiud.scr

\include /<64path>/ingres/files/iiud64.scr

or

\include /<OpInpath 1.x>/ingres/files/iiud.scr

\include /<OpInpath 1.x>/ingres/files/iiud65.scr

or

\include /<Inpath 2.0>/ingres/files/iiud.scr

\include /<Inpath 2.0>/ingres/files/iiud65.scr

or

```
\include /<Inpath 2.5>/ingres/files/iiud.scr
```

```
\include /<Inpath 2.5>/ingres/files/iiud65.scr
```

to

```
\include /<Inpath 2.6>/ingres/files/iiud.scr
```

```
\include /<Inpath 2.6>/ingres/files/iiud65.scr
```

After the reload is complete, verify that the table `ii_id` has only 1 row. Type `isql <database>`, and `select * from ii_id`. If more than 1 row is returned, delete the row with the lowest `object_id`.

Step 29 [Each DB] Front-End Upgrade

Run `upgradefe` on each database, which brings the front-end catalogs up to Ingres:

```
upgradefe dbname INGRES
```

The word `INGRES` should appear in uppercase.

Step 30 [Each DB] Reapply Optimizer Statistics

Regenerate optimization statistics using the `optimizedb` command (if time is short), and use the statistics that were dumped from the Ingres 6.4 installation in Step 8 (Optional `Statdump`):

```
optimizedb dbname -i dbname.stats
```

Step 31 [Each DB including iidbdb] Checkpoint

Checkpoint each database. If the database was journaled previously, use the `+j` flag to turn journaling on. Refer to the `infodb` output from Step 10 to see which databases were journaled; `iidbdb` should always be journaled.

Step 32 Application Upgrade

Install the Ingres versions of the applications. Then restore user logins and resume normal operation.

This completes the Unload/Reload upgrade procedure.

Upgrade Using Upgradedb

Upgrade using `upgradedb` transforms your database in-place from Ingres 6.4 to Ingres. This is an intricate process due to the large number of enhancements made. `Upgradedb` has been carefully written and tested. Most sites should be able to use it.

The idea behind the following `upgradedb` procedure is that the less work `upgradedb` does, the better. Each database is prepared by dropping all objects that can be recreated, that is, by dropping everything but the base tables. Each base table must be checked to make sure it is valid and has no internal damage. After the upgrade, the various database objects are recreated.

The procedure allows you to cut and paste the output of an `unloaddb` run to generate SQL that recreates database objects and storage structures. If procedures already exist to recreate database objects and storage structures, these can be used instead. Make sure that the procedures recreate all the relevant objects. If users or applications create database objects, it might be better to cut and paste from `unloaddb`.

The `upgradedb` procedure assumes that you can become any user who owns objects in any database (using `login` or UNIX `"su"`). If this is not feasible, you can run as user `ingres`, and use the `-u{user}` flag to pretend to be that user any time you have to run an Ingres command.

Upgradedb Upgrade Procedure

In this procedure, the notation [Each DB] means: "For each database, not including the `iidbdb`: Become the DBA for that database; `cd` to the `unload` directory for the database that you create in Step 1; and perform this step."

Do not include the `iidbdb` or Star distributed databases in the list unless instructed. If you are using Ingres/Star, remember to include the coordinator database in the list of databases.

Step 1 [Each DB Including Star DDBs] Create Unload Directories

Create a directory for each database. Use this directory for holding various scripts (but no data), the amount of disk space needed is small. A megabyte per directory is generous. Make the directory world writeable.

```
mkdir /someplaceldbname
```

```
chmod 777 /someplaceldbname
```

Steps 2 through 4 can be omitted if procedures already exist to recreate all database objects and storage structures. However, it will be necessary to make the appropriate changes to `oi_prep.sh` for remodifying all tables.

Step 2 [Each DB Including Star DDBs] Run Unloaddb

Run `unloaddb` against each database. The `unloaddb` command does not unload the database; it simply creates copy in and copy out scripts. These scripts can be edited to produce a collection of scripts that recreate various database objects and storage structures.

Note: If using Ingres/Star, unload the CDB in the same way as for a local database. For a DDB use `unloaddb/star`:

```
unloaddb dbname (regular db or CDB)
```

```
unloaddb dbname/star (Star distributed db)
```

Step 3 [Each DB Including Star DDBs] User Check

Examine the `unload.ing` and `reload.ing` scripts that were created in Step 1. Each script contains one line for each user who owns a database object. Make sure that all the users listed are valid; old databases may well have objects created by users who are no longer around.

For any unwanted users, delete the relevant lines from `unload.ing` and `reload.ing`. Delete the `cp{user}.in` and `cp{user}.out` files, and go into the database and clean out that user's objects.

Step 4 [Each DB] Editing the Unloaddb Output

The `unloaddb` output needs to be modified for recreating just the database objects and storage structures.

This step is best done manually with a text editor. Edit each `cp{user}.in` file except for the `cp_ingre.in` file. Starting at the end, there are blocks of:

- create rule statements—extract these statements into a file named {user}_rule.sql.
- create procedure statements and any accompanying “grant” statements—extract them into a file named {user}_dbp.sql.
- create dbevent statements and associated “grant” statements—extract them into {user}_event.sql.
- create view and/or QUEL define view and associated grant statements—extract them into {user}_view.sql.
- For each user with a cp{user}.in file, extract all the modify statements into a file and all modify and all create index statements into another. This can be accomplished using the commands:

```
sed -n -e '/^modify/,/\p\p\g/p' cp{user}.in >{user}_modify.sql
```

```
cp {user}_modify.sql {user}_modindex.sql
```

```
sed -n -e '/^create index/,/\p\p\g/p' cp{user}.in >>{user}_modindex.sql
```

SQL scripts have now been created that can recreate any database object or storage structure owned by any user in any database.

Step 5 [Each DB Including iidbdb] Checkpoint

This step is optional since you are going to take a system backup and another checkpoint later.

Checkpoint all databases. Copy the checkpoint files to tape and verify that the tape can be read.

Step 6 Disable User Access

From this point through the end of the upgrade, the production system is not available for use.

Step 7 Ingres Shutdown and System Backup

Shut down Ingres. This *must* be a clean shutdown, leaving non-flushed transactions in the Ingres transaction log.

This can be achieved by shutting Ingres down, restarting Ingres, then shutting it down again. Check the recovery process log (II_RCP.LOG) for “RCP Shutdown completed normally.”

Take a system backup, using whatever command is appropriate to the platform. Ensure that all Ingres directories—data, checkpoint, journal, dump areas, and the `$II_SYSTEM/ingres` directory containing Ingres files and executables are included. Back up the application directories.

Watch out for symbolic links and cross-mounts; make sure that it is real data that is saved and not a symbolic link.

If Ingres is normally started up at boot time, include the root file system in the backup. Alternatively, print a copy of any Ingres boot time startup and shutdown scripts.

For security, perform this step twice. Check the tapes after the backups to ensure that the tapes can be read.

After the completion of the backup, restart Ingres.

Step 8 [Each DB] Optional Statdump

You can omit this step if you have sufficient time to run `optimizedb` within the upgrade plan for a full `optimizedb` against your databases. Using this shortcut may result in some of the new Ingres metrics not being available.

Dump the existing optimizer statistics. Run `statdump` with the `-o` flag to a file for each database:

```
statdump -o dbname.stats dbname
```

Step 9 [Each DB] Object Cleaning

Drop all non-table objects from the database including:

- Optimizer statistics
- Views
- Rules
- Database procedures
- Database events
- Secondary indexes

In addition, modify all tables to heap. The shell script provided in the "UNIX-Specific Shellscrip" appendix can be used to perform this task automatically.

Using the C shell:

```
oi_prep.sh dbname |& tee oi_prep.log
```

If there are any dependent views, “drop” errors messages may be reported on those views; (oi_prep.sh does not bother to drop views in reverse dependency order). Ignore those “drop” errors.

Run verifydb checks against the database. The verifydb -odbms command may output messages:

```
S_DU1611_NO_PROTECTS iirelation indicates that there are protections for table (owner), but none are defined.
```

```
S_DU0305_CLEAR_PRTUPS Recommended action is to clear protection information from iirelation, and S_DU1619_NO_VIEW iirelation indicates that there is a view defined for table (owner), but none exists.
```

```
S_DU030C_CLEAR_VBASE Recommended action is to clear view base specification from iirelation.
```

The messages can be ignored. Also, ignore the “patch warning” message that warns of the loss of user tables in “runinteractive” mode. This mode is not going to be used.

Some databases may produce a “verifydb failed” message and then abort. If this happens, run the Terminal Monitor with the update system catalogs flag:

```
sql +U dbname
```

```
SELECT * FROM iistatistics;\go
```

No rows should be returned. If there are rows, this is the likely cause of the verifydb problem. Delete the rows:

```
DELETE FROM iistatistics;COMMIT;\go\quit
```

Rerun the verifydb command as shown at the end of the oi_prep.sh. If error messages are returned from verifydb, correct the problems before continuing. Contact Technical Support for help if necessary. Do not process Ingres/Star distributed databases.

Step 10 [Each DB] Record infodb

Run infodb against each database, saving the output. The output may be needed later. For example, you may need to know whether the database was journaled, the data locations where the database resides, and in what order the locations were configured.

```
infodb dbname >infodb.out
```

Step 11 Clean iidbdb

Become user `ingres`, and run a subset of the Object Cleaning step against the master database `iidbdb`. It is assumed that there are no objects created by users in the `iidbdb`.

```
statdump '-u$ingres' -zdl iidbdb
```

```
sysmod -s iidbdb
```

```
verifydb -mrun -sdbname iidbdb -opurge
```

```
verifydb -mrun -sdbname iidbdb -odbms
```

Messages from `verifydb` should be handled in the same way as in Step 9 (Object Cleaning).

Step 12 [Each DB Including iidbdb] Checkpoint -j

Checkpoint each database now that they have been cleaned up. This is done for two reasons:

- To turn off journaling for all databases
- If `upgradedb` fails, this checkpoint can be used to recover and try again.

Save the configuration file stored in the dump area after each *checkpoint*. The configuration file is small.

```
ckpdb -d -j dbname
```

```
cp $II_DUMP/ingres/dmp/default/{dbname}/aaaaaaa.cnf {somewhere secure}
```

Tip: Check point `iidbdb`. `iidbdb` does not have an “unload” directory, store the `aaaaaaa.cnf` file in a safe place.

Step 13 Record Ingres Configuration

As user `ingres`, execute the “`showrcp`” command and record the results. Record the contents of the `rundbms.opt` file in `$II_SYSTEM/ingres/files`. Use this information as a guide for configuring Ingres. The Ingres installation procedure does not preserve the existing Ingres tuning parameter settings. As the `ingres/files` directory is going to be deleted, save the information.

Step 14 Ingres Shutdown

Ingres 6.4 Shut down Ingres using `iishutdown`.

OpenIngres and later Shut down Ingres using `ingstop`.

Step 15 Disable Ingres Startup

If the machine starts Ingres automatically on bootup, turn auto-starting off.

On most UNIX platforms, there will be a file in `/etc/init.d` or `/sbin/init.d` that does Ingres startup and shutdown; place an “`exit 0`” at the top of that file. To do this may require root privilege; therefore, the system administrator may need to perform this step.

Make sure that the operating system is correctly configured for Ingres (see the System Administration Preparation section of the “Getting Started” chapter). If a reboot is necessary following parameter changes, it should be done at this time.

Step 16 Preserve Site Modifications

Sites customize the `$II_SYSTEM` directory tree or files within it. The most common customization is to change the termcap and keyboard map files in `$II_SYSTEM/ingres/files`. Also check for local collation sequence files. Ideally, save the original collation definition files; and the compiled files in `$II_SYSTEM/ingres/files/collation` should be saved as well.

Copy any customized files to a safe place, *not* `/tmp` and *not* anywhere in `$II_SYSTEM/ingres` directory. If you are not sure that you can identify customized files:

- Delete all `*.log` and `*.LOG` files from `$II_SYSTEM/ingres/files`
- Copy the entire contents of the `$II_SYSTEM/ingres/bin`, `$II_SYSTEM/ingres/files`, and `$II_SYSTEM/ingres/utility` directories to a safe place

This is copying more than needed, but the copy can be deleted when Ingres has been running live for a period. You may not discover for weeks that you need a Vision template or keyboard map as part of the `$II_SYSTEM/ingres` directory tree.

This tar command copies everything that is needed:

```
cd $II_SYSTEM/ingres
```

```
tar cf - bin files utility | (cd /someplace/safe;tar xf -)
```

Step 17 Login Fixups

If it is needed, make sure that the Ingres users' login sets LD_LIBRARY_PATH or the platform equivalent. Make sure that the users' login does not use ingprev1 or install your ingprev1 substitute. See the System Administration Preparation section in the "Getting Started" chapter. Check all your database owner (DBA) logins to ensure that they are properly set up for Ingres.

Step 18 Record Ingres Settings

The upgrade runs more smoothly if the Ingres executables, control files, and the Ingres environment variables are deleted. Since this means that the default Ingres settings will need to be reentered, make sure you know what they are.

Ingres 6.4 Copy \$II_SYSTEM/ingres/files/symbol.tbl to a safe area not in the Ingres directory tree.

OpenIngres and later Copy \$II_SYSTEM/ingres/files/symbol.tbl, config.dat, and protect.dat to a safe area not in the Ingres directory tree.

Step 19 Clean Up Ingres

To guarantee a clean environment for Ingres, invoke the following commands as appropriate:

```
cd $II_SYSTEM/ingres
```

Ingres 6.4 **rm -rf bin files lib utility dbtmplt version.rel admin**
mkdir files
Copy symbol.tbl back into the \$II_SYSTEM/ingres/files directory.

OpenIngres and later **rm -rf bin files lib utility admin**
mkdir files
Copy config.dat, protect.dat, and symbol.tbl back into the \$II_SYSTEM/ingres/files directory.

Step 20 Create Work Location

The Ingres installation asks for an Ingres location for temporary files and sorting. The installation creates the directories if they do not exist. However, the installation procedure may not properly set the protections for the directories, which can lead to upgradedb failing when upgrading iidbdb. To avoid this, consider creating the work location manually. See the "Using Work Locations" chapter of the *Database Administrator's Guide* for information on placement of your default work location. As user ingres, assume a work location called /mywork:

UNIX

```

/mywork:
mkdir /mywork/ingres
mkdir /mywork/ingres/work
mkdir /mywork/ingres/work/default
mkdir /mywork/ingres/work/default/iidbdb
chmod 755 /mywork/ingres
chmod 700 /mywork/ingres/work
chmod 777 /mywork/ingres/work/default
chmod 777 /mywork/ingres/work/default/iidbdb

```

Windows

```

md \mywork\ingres
md \mywork\ingres\work
md \mywork\ingres\work\default
md \mywork\ingres\work\default\iidbdb

```

Step 21 Install Ingres

Refer to the Ingres platform specific installation instructions. Enter the directory locations for the `ii_database`, `ii_checkpoint`, `ii_journal`, and `ii_dump` that were recorded in Step 18 (Record Default Locations).

During the DBMS server setup, a prompt requests whether all databases are to be upgraded; answer **No**. The install procedure automatically upgrades the `iidbdb`. If the upgrade of `iidbdb` fails, see the Upgradedb Problems section of the “Getting Started” chapter. It is better to complete the Ingres setup, and then use the `upgradedb` command to upgrade the user databases.

If a patch needs to be installed for Ingres, run a complete `ingbuild`. The patch may contain a fix for `upgradedb` or installation setup. Do the following:

1. Run `ingbuild`.
2. Select `Current`, then `SetupAll`.
3. Install the patch according to the instructions.

Setup now uses the fixed version.

Step 22 Restore Site Modifications

If the checkpoint template file `cktpl.def` has been modified, the modifications may need to be carried forward into Ingres. The `cktpl.def` from Ingres 6.4 cannot be used in Ingres, as the file format has changed. This means that the changes will have to be recreated using the Ingres 6.4 `cktpl.def` as a guide. See the *Ingres Database Administrator's Guide*.

If the archiver exit script `acpexit` was changed in Ingres 6.4, the changes need to be made to the Ingres template (`acpexit.def`), and then move the file to `$II_SYSTEM/ingres/files/acpexit`.

Set the default to the save directory that was created in Step 16 (Preserve Site Modifications), and restore any site-specific files, for example, local collation sequence files.

Step 23 Start Ingres

Run `ingstart`.

Step 24 Upgradedb

Databases can be upgraded one at a time:

```
upgradedb <dbname>
```

or all databases can be upgraded:

```
upgradedb -all
```

See the “Upgradedb Problems” section if errors occur. Correct the errors and rerun the `upgradedb` utility.

Log `upgradedb` output to a file:

```
upgradedb -all |& tee upgradedb.log
```

Step 25 Configure Ingres

Run Configuration By Forms (CBF) and do a first pass configuration for the Ingres installation. Use the `rundbms.opt` and `showrcp` information from Ingres 6.4 as a guideline. For information about CBF and the various tuning parameters see the *System Administrator's Guide*.

Derived parameters are recalculated when values they depend on are changed. If derived parameters are set, they can be “protected” against change.

Ingres 2.0 calculates very large lock and resource limits parameters. Consider reducing these limits to the Ingres 6.4 settings.

On OS-thread platforms, do not turn on `async_io`; and do not declare the `II_NUM_SLAVES` Ingres variable.

Ingres supports larger QEF_SORT_MEM values than Ingres 6.4. Ingres does not need as much QSF_MEMORY as in Ingres 6.4. OS-thread platforms should not reduce quantum. Reducing quantum does not improve performance.

Step 26 Ingres Net Setup

Run netutil to create the vnode definitions for the remote installations. If installation passwords are needed, it is necessary to run mkvalidpw. See the *System Administrator's Guide* or the README file for your platform.

If there are NFS client-only installations that have not been set up, run ingmknfs to set them up.

Step 27 [Each DB] Recreate Objects

Using the scripts generated by Step 4 (Editing the Unloaddb Output), recreate the views:

```
sql -u{user} dbname <{user}_view.sql
```

Recreate in the following sequence:

1. Dbevents
2. Database procedures
3. Rules

Step 28 [Each DB] Reapply Storage Structures

For each {user}_modindex.sql script generated by Step 4 (Editing the Unloaddb Output), reapply storage structures and indexes:

```
sql -u{user} dbname <{user}_modindex.sql
```

Step 29 [Each DB] Reapply Optimizer Statistics

Regenerate optimization statistics using the optimizedb command (if time is short), and use the statistics that were dumped from the Ingres 6.4 installation in Step 8 (Optional Statdump):

```
optimizedb dbname -i dbname.stats
```

As Ingres computes additional statistics, it is better to have run optimizedb.

Step 30 [Each DB including iidbdb] Checkpoint

Checkpoint each database. If the database was journaled previously, use the +j flag to turn journaling on. Refer to the infodb output from Step 10 to see which databases were journaled; iidbdb should always be journaled.

Step 31 Application Upgrade

Install the Ingres versions of the applications. Then restore user logins and resume normal operation.

This completes the upgradedb upgrade procedure.

Upgradedb Problems

The procedures discussed in this chapter will help prevent most upgradedb problems. This chapter also provides a list of problems and how to recover from them. The problems in this list are the result of many types of upgrades. Not all problems exist in all versions of Ingres.

Problem 1: “Can’t delete database from server”

Upgradedb will sometimes start to upgrade a database, then quit after writing a conversion and upgrade message with: “Can’t delete database from server.”

This problem was seen in OpenIngres 1.2. The resolution is to re-issue the command “upgradedb–all.” No harm is done to databases not upgraded.

Problem 2: “Duplicate Key” Upgrading iidbdb

If upgradedb is rerun after iidbdb has been upgraded, a “Duplicate key on insert” message appears followed by a warning about iidbdb. The upgradedb then continues normally.

This message is related to upgradefe, which upgrades the front-end catalogs. This message can be ignored. No harm is done, and this problem is corrected in the Ingres releases.

Problem 3: “Product <name> has been made un-installable”

Upgradedb prints a message “Product <name> has been made un-installable by an incompatible dictionary upgrade.”

This message is related to upgradefe, which upgrades the front-end catalogs. This message can be ignored. The message appears to be caused by a database that has gone through a failed product installation.

Problem 4: Hang with "open() Error 13"

Upgradedb will start upgrading iiddb, and then hang part way through with no message. In `errlog.log` there is the message "open() error 13."

This problem is caused by an error in the permissions on the work directory structure—the iiddb work directory has protections 664 rather than 777. The problem may be caused by the Ingres user's `.cshrc` file not containing a `umask 0` command.

Problem 5: File Extend Conversion Loop

Upgradedb will loop, printing "file extend converting *table name*."

This problem is seen when a damaged database has a system catalog entry for a table, but the underlying file was missing. If the situation occurs, it can be corrected by aborting upgradedb; shutting down Ingres; copying any valid not-yet-upgraded table file to the missing file name—as shown in `errlog.log`; starting up Ingres; and rerunning upgradedb.

Problem 6: "Failed, aborting" Creating Internal Procedure

If upgradedb is aborted, perhaps by a system crash, rerunning upgradedb may produce the message, "creating internal procedure `iiqef_alter_extension`," followed by a "failed, aborting" message.

This problem occurs in the OpenIngres 2.0/9712. It is caused by a failure to detect that the procedure `iierror` already exists. A fix is available from Computer Associates Technical Support.

A workaround is to restore the database from the checkpoint taken in Step 12, and rerun the upgrade:

1. Shut down Ingres.
2. Delete the files from each data location of the database.
3. Un-tar the checkpoint files into their corresponding data locations.
4. Copy the `aaaaaaa.cnf` file into the root data location and the dump directory; restart Ingres.
5. Rerun upgradedb.

It may be preferable to ask Technical Support for help; or abort the entire upgrade, restore from backup, and restart the upgrade.

Problem 7: iifile_info View Not Recreated

Upgradedb drops the iifile_info system catalog view but does not recreate it. This problem appears in Ingres 2.0/9712. It is a patch fix; check with Technical Support. The workaround is to create the view manually:

```
sql '-u$ingres' +U dbname
```

```
----- iifile_info view definition (2.0/9712) -----
create view iifile_info(table_name, owner_name, file_name, file_ext,
location, base_id, index_id)as select r.relid, r.relowner,
ii_tabid_di(releid, reletdx), 't00', r.reloc, r.releid, r.reletdx
from "$ingres". iirelation r where r.reletdx=0 and(mod((r.relstat/32),
(2))=0)union all select r.relid, r.relowner, ii_tabid_di(releid,
reletdx), 't00', r.reloc, r.releid, r.reletdx from "$ingres".
iirelation r where r.reletdx!=0 union all select r.relid, r.relowner,
ii_tabid_di(releid, reletdx), 't' +charextract('0123456789abcdef',
mod((d.devrelocid/16), (16)) +1) +charextract('0123456789abcdef',
mod((d.devrelocid), (16)) +1), d.devloc, r.releid, r.reletdx from
"$ingres". iirelation r, "$ingres". iidevices d where r.reletdx=0
and(mod((r.relstat/32), (2))=0)and d.devreleid=r.releid and
d.devreleid=r.reletdx and(mod((r.relstat/4194304), (2))!=0)union all
select r.releid, r.relowner, ii_tabid_di(releid, reletdx), 't'
+charextract('0123456789abcdef', mod((d.devrelocid/16), (16)) +1)
+charextract('0123456789abcdef', mod((d.devrelocid), (16)) +1),
d.devloc, r.releid, r.reletdx from "$ingres". iirelation r, "$ingres".
iidevices d where r.reletdx!=0 and(mod((r.relstat/4194304), (2))!=0)
and d.devreleid=r.releid and d.devreleid=r.reletdx
----- end of iifile_info view definition -----
```

Problem 8: All to Public Grants Not Created

Upgradedb preserves the ALL-to-ALL flag in iirelation that allows ALL to PUBLIC access; but it fails to create the permits for Ingres in the iipermit table.

The symptom is that the affected tables are not visible in the QBF table listing except to the owner.

This problem was present in OpenIngres 1.2. The resolution is to issue the Grant All to Public on the affected tables.

UNIX-Specific Shellscript

This appendix documents the `oi_prep.sh` shellscript that is available only on UNIX systems. This shellscript is also mentioned in the `upgradedb` procedure, Step 9 (Object Cleaning).

```
----- start of oi_prep.sh -----
#!/bin/sh
# Prepare for Ingres 6.4 -> Ingres update.
# Usage: oi_prep.sh dbname
# This script implements Step 9 ("Object Cleaning") of the upgradedb procedure.
# You run it in the "unload" work directory that contains the extracts of the
# unloaddb for the database. It should be run as the DBA user for the database.
# Its job is to prep a database for the Ingres update, by dropping all
# nonessential database objects and running some verification procedures. The
# following is done for the database:
# - Drop all statistics
# - Drop all views, procedures, rules, and dbevents for all users.
# - Reapply all storage structures.
# - Sysmod the database
# - Run verifydb -odbms_check to double-check the system catalogs.
# - Run infodb and make sure the header says VALID.
The idea is that the less work upgradedb has to do, the more likely it is that it
will do it right!
This script expects to see files named according to the naming conventions given
in the upgradedb procedure section. It should not be too hard to adapt to
different circumstances if necessary.
Remember the database name:
dbName=$1
if [ -z "$dbName" ] ; then
echo "Usage: oi_prep.sh dbname"
exit 1
fi
User has to have II_SYSTEM defined, and Ingres must be up.
if [ -z "$II_SYSTEM" -o ! -d "$II_SYSTEM/ingres" ] ; then
echo 'II_SYSTEM must be defined for Ingres.'
exit 1
fi
sql iidbdb </dev/null >/dev/null 2>&1
if [ $? != 0 ] ; then
echo 'Ingres is not running, or the path is not set up properly.'
echo 'Please make sure that Ingres has been started.'
exit 1
```

```

fi
dba=`infodb $dbName | sed -n -e '/Database */:s/^.*(.*\([^\)]*\)).*$/\1/p`

# User has to be tmsdba.

userName=`IFS="(";set - `id`;echo $2`
if [ "$userName" != "$dba" ] ; then
echo "You are not the dba $dba for database $dbName"
exit 1
fi

# Decide which awk to use

AWK=awk
mach=`uname -s`
if [ "$mach" = 'SunOS' ] ; then
AWK=nawk
Fi

# Dump optimizer statistics

statdump -zdl $dbName

# Generate list of all views, rules, procedures, and dbevents by owner.

# Drop them all, they will be reapplied eventually (after the upgrade).

# Note that by just dumping out all the view names, we may get errors if a base
view is dropped before a dependent view. It seems a lot easier to just accept
that and tell the user to ignore any drop errors. The alternative is to grind
around in iidbdepends to detect and deal with dependent views. Ugh.

While we're at it we might as well pick up the names of tables with uncompressed
HEAP storage structure. In Ingres 6.4, unloaddb does not output any MODIFY
command for such tables. In order to touch all tables (to ensure their goodness),
we'll find uncompressed heaps and arrange to re-heap them while we drop other
objects for that owner.

sql $dbName <<!EOF!
\script /tmp/stuff.$dbName.$$
SELECT table_owner,'VIEW',table_name FROM iitables
WHERE table_type='V' AND table_owner <> '\$ingres'
UNION ALL SELECT dbp_owner,'PROCEDURE',dbp_name FROM iiprocedure
WHERE dbp_owner<>'\$ingres'
UNION ALL SELECT rule_owner,'RULE',rule_name FROM iirule
WHERE rule_owner<>'\$ingres'
UNION ALL SELECT event_owner,'DBEVENT',event_name FROM iievent
WHERE event_owner<>'\$ingres'
UNION ALL SELECT table_owner,'HEAP',table_name FROM iitables
WHERE table_type='T' AND table_owner<>'\$ingres'
AND storage_structure = 'HEAP' AND is_compressed = 'N'
ORDER BY 1,2,3;
COMMIT;
\go
\script
\quit
!EOF!
if [ $? != 0 ] ; then
echo
echo "SQL returned error while processing database $dbName"
exit 2
fi

# Transform output into "owner WHAT object-name"

/bin/ed /tmp/stuff.$dbName.$$ <<!EOF!'
1,/^{+---}/d

```

```

1,/^{+---/d
/^{+---/, $d
1,$s|//
1,$s/ *|/ /
1,$s/ *|/ /
1,$s|//
w
q
!EOF!

```

Generate awk script to take the stuff just dumped and make DROP commands:
cat - >/tmp/awk\$\$ <<'!XX!'

```

BEGIN {curOwner=""}
{
  if ($1 != curOwner) {
    if (curOwner != "") {
      print "\\quit";
      print "!EOF!";
      print "if [ \$$? != 0 ] ; then";
      print " echo 'Error running cleanup SQL'";
      print " exit 1";
      print "fi"
    }
    curOwner = $1;
    print "sql -u" curOwner,dbName,"<<'!EOF!'"
  }
  if ($2 != "HEAP") {
    print "DROP", $2, $3, ";COMMIT;\\p\\g";
  } else {
    print "MODIFY", $3, "TO HEAP;COMMIT;\\p\\g";
  }
}
END {
  if (curOwner != "") {
    print "\\quit";
    print "!EOF!";
  }
}
!XX!

```

This awk call is known to be network on SunOS 4.x, but should be OK pretty much everywhere else. On SunOS 4.x you have to juggle the supplied variable (dbName) to a different spot in the command line.

```

$AWK -v "dbName=$dbName" -f /tmp/awk$$ /tmp/stuff.$dbName.$$
>/tmp/drop.$dbName.$$
if [ $ $? != 0 ] ; then
  echo 'awk error, perhaps your awk is strange?'
  exit 3
fi

```

Ok, drop all sorts of stuff, re-heap heaps:

```

if [ -s /tmp/drop.$dbName.$$ ] ; then
  sh /tmp/drop.$dbName.$$
  if [ $ $? != 0 ] ; then
    echo "Error dropping objects from $dbName, review the output log"
    exit 3
  fi
fi

```

Reapply storage structures, and incidentally drop indexes. Upgradedb doesn't have any problems with indexes, but you really ought to re-modify after the upgrade anyway, so save the index creates until then (Ingres makes them more quickly anyway!)

```

for i in *_modify.sql ; do

```

```
# Guard against no *_modify.sql files at all
if [ "$i" != '*_modify.sql' ] ; then
    # Guard against empty files

    if [ -s "$i" ] ; then
        theUser=`expr "$i" : '\(.*\)_modify\.sql'`
        sql "-u$theUser" $dbName <$i
        if [ $? != 0 ] ; then
            echo "Error remodifying $dbName ($i)"
            exit 3
        fi
    fi
fi
done
sysmod $dbName
if [ $? != 0 ] ; then
    echo
    echo "Sysmod error, perhaps database $dbName is still in use."
    echo 'Make sure ALL users are locked out. Shut down and restart'
    echo 'Ingres if necessary to ensure this. Then, try again.'
    exit 3
fi
Do verifydb and infodb.
verifydb -mrun -sdbname $dbName -opurge
verifydb -mrun -sdbname $dbName -odbms_check
infodb $dbName | grep VALID
if [ $? = 0 ] ; then
    echo
    echo "Database $dbName seems OK, ready for upgrade if verifydb output
was OK."
    echo
else
    echo
    echo "Database $dbName does not appear to be consistent."
    exit 4
fi
rm -f /tmp/stuff.$dbName.$$ /tmp/drop.$dbName.$$ /tmp/awk$$
----- end of oi_prep.sh -----
```

Features Introduced in Ingres 2.6

Ingres 2.6 has a number of new enhancements including the following:

- User-visible language and DBA enhancements
- Internal performance enhancements
- Locking and logging system performance improvements
- Operating system integration
- Ingres/ICE enhancements
- JDBC enhancements

User-Visible Language Enhancements

Enhancements have been made to the internal performance that concern row producing procedures.

Row Producing Procedures

This enhancement to the Ingres database procedure language addresses the ability of Ingres to read and return to the caller multiple rows from a select statement.

With server-executed database procedures, the program logic of the procedure is executed entirely in the server address space. Multiple SQL DML requests are executed in a single invocation of the procedure, with only one interaction with the client application. The ability to process and return multiple “rows” of some composite data types with a single call to a server-resident database procedure adds to the potential for improved performance of an application.

In a typical computing environment with applications executing on a variety of computers throughout a network, this approach can significantly reduce the footprint of the client application and the traffic across the network.

Increase Maximum Size of Character Data Types

Prior to Ingres 2.6, character data types were limited to a maximum size of 2000 bytes; this restriction was imposed when the maximum size of a row was limited to 2 KB. This limit has been increased to 32000 bytes, the maximum row size supported in Ingres 2.6.

User-Visible DBA Enhancements

Enhancements have been made to the internal performance that concern auditdb utility, copydb utility, raw location support, and gatherwrite threads.

Usermod Utility

Ingres now includes a usermod utility that allows users to run the modify commands on user tables. Like sysmod, which does a modify on system catalogs, this utility is useful for maintaining user tables on a regular basis.

Running this utility regularly, or when the table has excess overflow pages, improves performance of user applications.

Auditdb Utility

Various enhancements to the auditdb utility required by Ingres Journal Analyzer are included in Ingres 2.6:

- Specification of fully qualified table names
- Correct formatting of the output for -aborted_transaction when used with -b and -e flags
- Corrected -aborted_transaction flag, allowing auditdb to write correct format for BT and ET records
- Savepoint information in the auditdb output. This is achieved by printing out the abortsave record, which contains the LSN of the aborted savepoint
- The order of output for lsn low/high fields for the ASCII output of auditdb, allowing the high lsn to be printed before low LSN
- Two new auditdb options: -start_lsn=<LSN> -end_lsn=<LSN> for non -all cases

See the *Command Reference Guide* for the syntax of the auditdb command.

Copydb Utility

The copydb utility has been modified to include several options and flags that modify the copy.in and copy.out scripts based on user requirements. The user can specify the order the copy and modify statements are written to the copy.in script, for example, whether to copy the data into the tables and then run a modify statement or the other way around. Other examples include the ability to remove hard-coded paths to the copy scripts, exclusion of location names, and exclusion of user-specific permissions such as grant statements.

Raw Location Support

Ingres 2.6 adds support for raw data locations on UNIX platforms. Raw data locations provide dramatic performance improvements over cooked locations.

GatherWrite Threads

A new internal Ingres thread type, GatherWrite, is used by the Ingres buffer manager during operations that require the flushing of multiple buffers from the cache such as write behind, consistency points, and table purges. This feature is only available on platforms that offer writev() support. Please consult the appropriate README file to determine whether this feature is supported on your platform.

This feature is enabled on a per-server basis using the gather_write parameter in CBF. The default setting is ON.

XML Import/Export Utility

XML is as a cross-platform, software- and hardware-independent tool for transmitting information. The XML import/export utility imports and exports XML data from Ingres tables to and from XML files. See the *Command Reference Guide* for the syntax of the XML Import/Export utility, impxml.

Ingres Journal Analyzer

The Ingres Journal Analyzer (IJA) is a powerful graphical tool that provides an interface to the Ingres journal files. You can use the Ingres Journal Analyzer to recover data from the journals and to apply journaled transactions to other databases, both local and remote. For information on the Ingres Journal Analyzer utility, see the *System Administrator's Guide*.

Ingres Import Assistant

The Ingres Import Assistant is a wizard that simplifies the task of importing data from a standard file format to an Ingres database. For information on the Ingres Import Assistant utility, see the Visual DBA online help.

Automated Creation of Location Directories

Before Release 2.6, the Ingres DBA had to manually create the directories for alternate locations as prescribed in the section “Creating a New Area” in the *Database Administrator’s Guide*. This step had to be performed prior to creating a Location with ACCESSDB or could be deferred if Locations were created using EXEC SQL CREATE LOCATION syntax. To circumvent directory permissions problems, ACCESSDB had to be run by the Ingres user whenever Locations were being created, altered, or extended.

This process is clarified and simplified in Release 2.6 with the following changes:

The Ingres Server performs all manipulations of Location directories. This resolves the permissions problems of earlier releases and allows any ACCESSDB User with the “maintain_locations” privilege to create, alter, or extend Locations.

The Server automatically creates Location directories when a CREATE/ALTER LOCATION statement is executed, whether by ACCESSDB or User-invoked SQL. Because only missing directories are created, the DBA retains the ability to manually create as much, or all, of the Location path as wanted before creating the Location.

Using the example from “Creating a New Area In UNIX”, the following directories will be verified or created automatically during the execution of
CREATE LOCATION new_loc WITH AREA='/otherplace/new_area', USAGE=(DATABASE)

Perms	Directory
	/otherplace
755	/otherplace/new_area
755	/otherplace/new_area/ingres
700	/otherplace/new_area/ingres/data
777	/otherplace/new_area/ingres/data/default

Note the following:

- Permissions are **not** changed for extant directories.
- The top-level directory “/otherplace” must exist and will **not** be created by the Server.
- Raw location directories (UNIX only) cannot be automatically created and must be made with the MKRAWAREA utility, which must be run by “root”. The Locations may be created prior to MKRAWAREA but a warning will be issued noting that the utility must be run prior to their use.

RMCMD Enhancements

Users commonly encounter problems running utilities that require exclusive access to the iidbdb database because rmcmd keeps a session open on this database. To counter this problem, rmcmd now attaches to imadb instead of iidbdb; imadb is a system database that contains no historical data; it is rarely backed up and requires little or no maintenance.

Microsoft Transaction Server Support

Support for tightly coupled XA threads and shared lock lists is now available to support Microsoft Transaction Server, using the CA-Ingres ODBC driver.

Concurrent Rollback

The concurrent recovery of multiple transactions is now possible.

Internal Performance Enhancements

Enhancements have been made to the internal performance that concern aggregate sort nodes, composite histograms, and optimizer support for hash joins.

Aggregate Sort Nodes

Improvements to Ingres aggregate handling allows Ingres to better support data-mining products such as Computer Associates DecisionBase, which make extensive use of data aggregation.

Composite Histograms

The composite histograms enhancement allows the creation of composite or multi-column histograms that model much more accurately the dependence of the values of one column on another and lead to far better selectivity estimates and, ultimately, to better query plans.

Optimizer Support for Hash Joins

Hash joins have been implemented in Ingres 2.6. A hash join is one in which a hash table is built with the rows of one of the join sources by hashing on the key columns of the join. The rows of the other join source are then read and hashed into the table on their key columns. The hashing of the second set of rows quickly identifies pairs of joining rows. This technique requires no index structures on the join columns (as does KEY join), nor does it require sorting on the join columns (as does merge join).

Locking System Performance Improvements

A number of improvements have been made to the Ingres locking system to eliminate or minimize bottlenecks identified when running various performances tests.

Preallocated RSB/LKBs

Each resource RSB now has embedded within it a lock block (LKB), removing the need for a separate, contentious LKB allocation every time a new resource is allocated.

An LLB stash of LKBs is also maintained, similar to the RSB stash.

When an RSB or LKB is freed, it is returned to the LLB's stash; when the lock list itself is freed, all stashed RSB/LKBs are returned to the free pool.

Miscellaneous Locking System Improvements

The following miscellaneous locking system improvements are included in Ingres 2.6:

- The number of RSB waiters and converters are now maintained in the RSB.
- The deadlock wait-for graph lock (lkd_dlock_lock) does not need to be held if the RSB has neither waiters nor converters.

- The LKREQ built in the stack does not need to be copied to the LKB indiscriminately.
- When a lock request is blocked, the blocker's identity is now saved in the LKREQ and formatted in SYS_ERR only when the request fails.

Logging System Performance Improvements

A number of improvements to the Ingres logging system eliminate or at least minimize bottlenecks identified when running various performance tests. These changes include elimination of contentious `current_llb_mutex`, faster log forces through forcing only what needs forcing, and improved concurrency potential (fast resume).

Buffer Manager Performance Improvements

A number of improvements have been made to the Ingres buffer manager to eliminate or minimize bottlenecks identified when running various performance tests. These include:

- Removal of stats for fixed priority pages. In Ingres 2.6, stats are tracked by buffer page type for better analysis of the BM's LRU algorithm.
- Raising a buffer's priority each time it is fixed; previously it was raised only when newly fixed.

Operating System Integration

Enhancements have been made to the internal performance that concern 64-bit operating systems and operating system thread implementation on Linux.

64-Bit Operating Systems

Now that Microsoft, Sun, HP, and Linux vendors have produced 64-bit versions of their operating systems, we are providing a 64-bit build of Ingres on these platforms. Every effort is made to exploit large memory and files in these 64-bit environments.

Operating System Thread Implementation on Linux

Ingres 2.6 provides support for operating system threads in Linux environments including Intel, Alpha, S/390, and IA64. Operating system threads perform better in most circumstances than the internal Ingres threading model.

Ingres/ICE Enhancements

Ingres/ICE development environment and setup and configuration are addressed in the Ingres 2.6 release through integration with an existing Web application development environment.

ICE Development Environment

The issue of a development environment presence is addressed in this enhancement. The Ingres/ICE macro DTD can be used with an XML aware editor to provide a development environment for ICE application development. A converter has been added to take new macro syntax into the old macro syntax during page registration.

JDBC Enhancements

The following are JDBC 2.0 extensions that have been added to Ingres:

- Compatibility with GA release (protocol levels)
- Driver allows execution in JDK 1.1 environment
- Batch processing
- javax – two-phase commit
- javax – client connection pooling
- Updateable result sets

The following are new features:

- Support for Ingres intervals
- Coalescing statement IDs
- Utilization of VNODE passwords
- Local connections without passwords

- Support for procedure table parameters
- Support for row producing procedure

Support for Unicode

This release contains the first phase of Ingres support for Unicode; further Unicode support will be added to future releases. In this release the DBMS supports three new data types, `nchar`, `nvarchar`, and `long nvarchar`. These data types store character data using two bytes for each character. Collation of these data types uses the standard collation algorithm as defined by the Unicode organization, and the data types may be used in indexes and database statistics.

The native two-byte (UCS2) format is supported and maintained through the entire Ingres process, from the front-end application, through the DBMS, to the data representation on disk. The embedded SQL preprocessor for C and C++ supports declaration of `wchar_t` variables, which are assumed to contain multi-byte Unicode character strings. VDBA also supports these new data types. Support for the ODBC and JDBC drivers is present through their normal Unicode interfaces. These new data types are not supported in any of the Ingres character-based tools or any of the terminal monitors. This release does not support coercion between Unicode data types and non-Unicode data types such as `char` and `varchar`.

New Character Sets for Euro Currency Symbol Support

Two new character sets that contain the Euro currency sign (€, Unicode U+20AC) have been added: IS885915 and WIN1252.

To set the Ingres Money Format to the Euro currency symbol you must issue the following command:

```
ingsetenv II_MONEY_FORMAT L:€
```

Alternatively, this value may be set in the Ingres Visual Manager (IVM).

Windows

WIN1252 corresponds to Windows code page 1252 Latin 1. This is the common character set of most American and Western European Windows PCs and also includes the Euro sign. Users wishing to use the Euro symbol in a Windows GUI environment will need to select the WIN1252 character set at installation time. To set this code page in a Windows command prompt environment, you must issue the following Windows command:

```
chcp 1252
```

The default font in a Windows command prompt does not provide support for the Euro currency symbol; for a workaround, set the font to Lucida Console. The Lucida Console font has moved the line drawing characters, used in Ingres forms, into an area not accessible to Ingres binaries, so we have provided rudimentary line drawing in the IBMPCD terminal entry. To set this terminal type, you must issue the following command:

```
ingsetenv TERM_INGRES IBMPCD
```

or set TERM_INGRES through IVM or specify this terminal type at install time. 



IS885915 corresponds to the ISO 8859-15 Latin 9-character set that is almost identical to the ISO 8859-1 Latin 1 set, except for eight characters; chief among them is the Euro currency sign (€, Unicode U+20AC).

If you have an existing installation and would like to change the installation's character set, be aware that this is not normally supported since characters already in your databases could be displayed incorrectly by the new character set. However, since the ISO 8859-15 only has eight characters that are different from ISO 8859-1, if you can verify that none of the eight characters are already present in your databases, you could safely change the set (by changing `II_CHARSETxx`). The following table details these differences and provides the corresponding Unicode character names:

Hex		ISO 8859-1		ISO 8859-15
A4	■	CURRENCY SYMBOL	€	EURO SIGN
A6		BROKEN BAR	Š	LATIN CAPITAL LETTER S WITH CARON
A8	¨	DIAERESIS	š	LATIN SMALL LETTERS WITH CARON
B4	'	ACUTE ACCENT	Ž	LATIN CAPITAL LETTER Z WITH CARON
B8	,	CEDILLA	ž	LATIN SMALL LETTER Z WITH CARON
BC	¼	VULGAR FRACTION ONE QUARTER	Œ	LATIN CAPITAL LIGATURE OE
BD	½	VULGAR FRACTION ONE HALF	œ	LATIN SMALL LIGATURE OE
BE	¾	VULGAR FRACTION THREE QUARTERS	ÿ	LATIN CAPITAL LETTER Y WITH DIAERESIS

Features Introduced in Ingres 2.5

Ingres 2.5 introduced the following enhancements and functionality:

- Sort enhancements
- New SQL functionality
- Optimizedb enhancements
- Read-only database support
- ANSI/ISO constraint enhancements
- Large cache support
- Dynamic Write Behind threads
- Partitioned transaction log file
- Replicator enhancements

In addition, this appendix describes the following features:

- Ingres 2.5 Net Features
- Ingres/ICE 2.5 Features
- Visual DBA Features

Sort Enhancements

Changes were made to improve the performance of both the in-memory (QEF) sort and disk (DMF) sort of Ingres.

QEF Sort Enhancements

QEF was improved by fine-tuning the sort algorithm, resulting in fewer comparisons between sort rows. The sort algorithm is a major consumer of CPU time in a sort. QEF was also improved with a change that results in the rows being partially sorted as they arrive in the sort. This change introduces two distinct benefits. First, duplicate rows are detected and discarded more quickly from duplicate removal sorts (as required by "select distinct..."). This in turn increases the number of rows that can be processed in memory for a duplicates removal sort, avoiding more expensive disk sorts in many instances. The second benefit is that the first rows in the sort sequence can be returned before the remaining rows are completely sorted. Tests show that the first sorted row is available with as few as 20% of the overall comparisons required to complete the sort. This means that browsing/scrolling applications see the first set of rows in less time than before.

DMF Sort Enhancements

The first set of DMF sort enhancements also involve fine-tuning of the sort algorithms, which should result in a 5-10% reduction in CPU time of typical sorts. As with the QEF sort, duplicate rows are detected and discarded sooner in duplicates removal sorts. This should result in smaller disk work files and faster overall sort performance. Prior to release 2.5, the entire result of a DMF sort was spooled to an internal temporary table before the sorted rows were returned to the caller. In Ingres 2.5, the temporary table has been eliminated and the rows are returned directly from the sort structures to the caller. This has the same effect as the early return of sorted rows described above for the QEF sort. That is, the first rows should be returned much sooner than was previously the case.

The final DMF sort enhancement is the introduction of a "parallel sort" technique. Sorts that exceed a user-configurable threshold spawn additional threads. The sort is split up and its rows delivered to the sub-threads for sorting. The sorted subsets of the rows are then delivered back to the parent thread executing the query, where they are merged to form a single sorted stream of rows. On multi-CPU machines, this results in a significant reduction in the elapsed time required to sort (between 25% and 50% in testing). An added benefit to the parallel sort technique is that it is encapsulated within the DMF sort. This sort is used for the execution of queries with sorting requirements (such as for order by, group by, and distinct requests, or for implementing certain join algorithms). However, it is also used to sort rows for index creation/update in modify, create index, and copy operations. All users of the DMF sort derive the performance benefit of the parallel sort.

Parallel Sort Techniques

The “parallel sort” technique outlined above is used to sort rows for parallel index creation, greatly reducing the time taken for index creation in multi-CPU environments.

ANSI/ISO Constraint Enhancements

Ingres referential and unique/primary key constraints result in the creation of indexes “under-the-covers” to improve the performance of the constraint enforcement mechanisms. Prior to release 2.5, these indexes were plain B-tree indexes stored in the default location of the database. However, B-tree is not always the best choice (for example, hash is better for many unique key applications), and use of the default location can degrade performance if many large indexes are created.

Release 2.5 solves these and other problems by including a “with” clause for constraint definition. The “with” clause allows the overriding of default index options with anything normally coded in an index creation “with” clause. For example, the index structure and location, as well as fillfactor and other index options can be explicitly specified for each constraint. The “with” clause applies to column and table constraints defined with both the create and alter table statements. A unique/primary key constraint can be generated to use the base table structure for its enforcement rather than a separate secondary index.

Release 2.5 also introduces the ANSI/ISO notion of referential actions for the definition of referential (foreign key) constraints. In releases prior to Ingres 2.5, the attempt to delete a referenced row for which matching referencing rows exist, or to update the primary key of a referenced row to some other value while matching referencing rows still exist for the old value, was met with an error and the request was aborted. Either operation had to be preceded by a delete of the matching referencing rows or an update of the foreign keys to some value that exists in another referenced row.

Release 2.5 allows the definition of referential actions for each referential constraint, which defines alternative actions to be taken in the circumstances defined above. A separate action can be defined for both the delete case (deletion of a referenced row with matching referencing rows) and the update case (updating the key of a referenced row with matching referencing rows). The options include *cascade*, in which case the delete or update is cascaded to the matching referencing rows (so that the referencing rows are also deleted or updated to the same value), and *set null*, in which case the foreign key of the matching referencing rows is set to null. These actions permit a more complete definition of the semantics of the referential relationship and allow the application to execute more efficiently.

Large Cache Support

In Ingres 2.5 the total number of pages in all caches has been revised from an un-enforced limit of 65536 to $2^{32}-1$. Ingres 2.5 supports a 4 GB cache.

In previous releases, when those pages belonging to a specific table needed to be located, the buffer manager sequentially searched every buffer in every cache to find them. Even in installations with small caches, this was an expensive operation, especially in those frequent instances in which there were no table-pages in any cache. This operation occurred, for example, when a table's TCB was about to be released, typically when all referencing transactions had **no immediate need to use the table**.

In Ingres 2.5, a cross-cache table hash queue has been added to the buffer manager to which pages are added as they are faulted in and removed when they are tossed. Thus, when the need to know a table's pages arises, a hash on the table's database ID and table ID is made and that list searched for matching pages. This change results in a significant decrease in the number of cache pages visited and is most dramatic in installations configured with very large or multiple caches.

Dynamic Write Behind Threads

In releases prior to Ingres 2.5, a fixed number of Write Behind threads were configured in each server in an installation and initiated when the server started. These threads served all caches and were awakened when the number of modified pages in any cache exceeded a predefined threshold. In a shared cache environment, all Write Behind threads in all servers were simultaneously activated when this threshold was reached. This led to a “thundering herd” phenomenon in which n Write Behind threads concurrently pounded through the caches, competing for modified pages to flush.

The optimum number of Write Behind threads is the minimum number required to:

- Maintain the modified buffer count below the Write Behind start threshold
- Supply sufficient free pages to avoid synchronous writes.

The optimum number of Write Behind threads varies with the instantaneous demand for free pages in a particular cache; it always begins at one when the threshold is first breached and a Write Behind event signaled. In Ingres 2.5, if the single Write Behind thread is unable to keep up with the demand, additional Write Behind threads are created until either equilibrium is achieved or the upper limit on thread numbers is reached. If better than equilibrium is achieved (more modified pages are being freed than are in demand), the excess Write Behind threads terminate one-by-one, while the remaining threads continue to monitor the free buffer demand to achieve the write-behind end threshold.

Ingres 2.5 introduces cache-specific Write Behind threads, resulting in increased concurrency and eliminating the chance of free page starvation.

Partitioned Transaction Log File

The structure of the Ingres log file in Ingres 2.5 is changed from a single file to 1-16 logically striped files of equal size. Properly configured, partitioning in this manner encourages better log performance by spreading disk contention across multiple disks instead of concentrating it on a single device. The full log file paths are now defined through Configuration Manager or CBF rather than through the symbol table.

Optimizedb Enhancements

A variety of enhancements have been made to optimizedb. The new flag `-zlr` causes optimizedb to retain the original repetition factor when rebuilding an existing histogram. This is useful when a histogram (and its repetition factor) is built once by reading the whole table, then updated later using sampling (which can produce inaccurate repetition factors). A minor bug was fixed to allow the `"l"` flag to request an exclusive lock on the database during optimizedb processing (just as for other command line utilities). The current limit of 1000 parameters coded in a separate file using the `-zf` parameter has been lifted. There is now no limit to the number of such parameters. An `-o filename` option (similar to that in `statdump`) has been added to optimizedb. It creates a `statdump`-style output file, which can then be input back to optimizedb with the `-i filename` option. But more importantly, it does not update the catalog `iistatistics` and `iihistogram` tables. This allows a busy shop to construct histograms at anytime, with no worry about update conflicts. Then at a convenient later time, optimizedb can be run with the `-i` option to add the histograms to the catalog.

In addition to the flag enhancements, an enhancement has been introduced to allow more accurate histograms to be built on columns with significant skew in their value sets. Specifically, a column with many distinct values, which would generate an inexact histogram, now produces exact cells for values that occur significantly more than the average. This permits much more accurate estimates in the compilation of queries with restrictions on such columns, and, therefore, better query plans.

Miscellaneous Enhancements

Expressions are now permitted in both the order by and group by clauses. Several synonyms have been added to existing Ingres data types (such as character long object and clob for long varchar and binary long object and blob for long byte). The query compiler has been enhanced to compile more efficient strategies for complex queries involving aggregate views and unions.

Read-only Database Support

Ingres 2.5 provides the ability to distribute a read-only database on a CD ROM or other read-only media.

Example:

To create a read-only database called mydatabase on UNIX:

1. Log in as ingres.
2. Change location to the staging directory:
cd /stagingarea
3. Create directory and subdirectories:
mkdir ingres
mkdir ingres/data
mkdir ingres/data/default
4. Place appropriate permissions:
chmod 755 ingres
chmod 700 ingres/data
chmod 777 ingres/data/default
5. Change location to the database files:
cd /install/inst1/ingres/data/default

6. Copy the database directory and its subdirectories to the new area:
cp -r mydatabase /stagingarea/ingres/data/default/
7. Copy the directory structure to the CD-ROM or other device:
cp -r ingres/data/default/mydatabase /cdrom
8. Create a new database location using the create location statement or using the accessdb utility:
create location cdromloc with area=/cdrom, usage=(database);
9. Use the createdb command to access the read-only database in the installation:
createdb -r cdromloc mydatabase

New SQL Functionality

Enhancements have been made to the internal performance that concern bit-wise operator support and miscellaneous functions.

Bit-wise Operator Support

The following functions have been added to Ingres 2.5 to provide support for bit-wise operations:

bit_add	Logical "add" of two byte operands
bit_and	Logical "and" of two byte operands
bit_not	Logical "not" of two byte operands
bit_or	Logical "or" of two byte operands
bit_xor	Logical "exclusive or" of two byte operands

For all of these bit functions, all operations proceed right to left. The shorter of two operands is padded with hex zeroes on the left. The result is a byte field equal in size to the longer operand.

Miscellaneous Functions

The following miscellaneous functions have been added to Ingres 2.5:

- intextract—Extract the number at the given location.
- ii_ipaddr—Convert an IP address to a byte 4.

Extended Date Support

Ingres 2.5 allows users to insert dates in the range 01-Jan-0001 to 31-Dec-9999.

64 Bit File Support

A major enhancement to Ingres 2.5 on operating systems that support 64-bit file systems is the ability to support file sizes greater than 2 GB. This means that larger table, dump, work, journal, and checkpoint files can be accommodated in a single location. It also removes the 2 GB limit on the size of the transaction log file.

Large Catalogs

In this release, Ingres allows system catalogs to use pages larger than 2 KB. This means that the user does not have to configure a 2 KB cache size in the DBMS just for system catalogs.

Row Locking for System Catalogs

For improved concurrency, the Ingres 2.5 DBMS automatically uses row locking on system catalogs when catalogs are created using pages larger than 2 KB. This feature is keyed from the system default page size, which is configurable through CBF or the Ingres Configuration Manager; createdb creates a database with system catalogs that have the default page size. Running sysmod on an existing database, however, does not automatically convert the system catalogs to use the default page size. The user must use the “with page_size” keyword to achieve this.

Update Mode Locking

Prior to Ingres 2.5, when the DBMS fetched a row for a cursor mode update, it would acquire an exclusive lock. In serializable mode, this lock would be held until the end of transaction, even when the row was not updated. In this release, the Ingres DBMS acquires an update mode lock for the row that is a candidate for update. If the row is actually updated, the update mode lock is converted to logical exclusive lock; otherwise, it is converted down to shared lock or released, depending on the current isolation level. Update mode locking is now the default for cursor updates.

Value Locking for Serializable Transaction with Equal Predicate

Prior to Ingres 2.5, the Ingres DBMS would hold a page lock on the leaf pages on B-tree tables for a serializable update transaction, even when using row locking. This was done to prevent other users from inserting a row in the qualified range to be updated. In Ingres 2.5, the DBMS uses a value locking protocol for serializable transactions with an equal predicate, thereby allowing better concurrency.

Case Expression Support

The SQL '99 case expression is implemented in this release. This is a particularly useful addition to SQL, allowing column or expression values to be decoded using a case statement embedded in select statement. For details of this new SQL addition, see the *SQL Reference Guide*.

Expression in Order By

In Ingres 2.5, an order by clause may now include columns not present in the select list of the corresponding query, as well as expressions including at least one column from one of the tables in the from clause. The clause now conforms to the SQL 99 standard. See the *SQL Reference Guide* for details.

Expression in Group By

Ingres 2.5 allows expressions in the group by list of an SQL statement. This implementation allows both expressions in the group by list (as long as they do not contain set functions) and an ordinal "entry in select list" number, as well as the standard simple column notation. The select list and having clause can contain non-set function expressions as long as all columns are also contained singly or in other expressions in the group by list. See the *SQL Reference Guide*.

Query Optimization/Execution Enhancements

Ingres 2.5 incorporates a variety of enhancements to the compilation and execution of queries.

Ingres Star Enhancements

In Ingres 2.5 the Ingres/Star server now contains support for the 1Pcplus commit protocol, which allows a single site that is not capable of supporting two-phase commit protocols to participate in a multi-site update within Star. This change allows a single database that is being accessed through an Ingres Enterprise Access gateway that does not support the SQL prepare statement to participate in a multi-site Star update.

Ingres 2.5 Net Features

GCF has responsibility for authenticating and validating clients for Ingres servers. Previously, Ingres security was built around operating system capabilities. The following improvements have been made to Ingres security for Ingres 2.5:

- Support for third-party security systems such as Kerberos
- Enhancements for data encryption and direct network server connections
- Improved existing security by addressing known problems
- Backward compatibility for existing applications

Support for third-party security systems requires dynamic configuration capabilities since these systems are not a requirement for installation. In a design emulating the emerging standard GSS-API, the Ingres 2.5 GCF security architecture is built around independent modules called *mechanisms*. Standard default mechanisms are provided for basic Ingres security and backward compatibility. Third-party security systems are supported through additional mechanisms, which are dynamically loaded as needed.

GCF security mechanisms provide the following capabilities:

- User authentication and validation
- Password validation
- Trusted server authentication and validation
- Distributed (single sign-on) authentication and validation
- Data encryption

Management of GCF security has been enhanced with new configuration parameters viewable through CBF and the Configuration Manager. Ingres 2.5 also sees the addition of attributes to the Ingres/Net VNODE database, and new IMA objects (many of which can be set at runtime) for enhanced IMA support.

Ingres/ICE 2.5 Features

New features added to the internal performance of Ingres/ICE concern security, session management, storage management, and macro language extensions.

Security

The security model for Ingres/ICE has been enhanced to provide additional levels of control for access and for content. The model also assists in the maintenance of large numbers of Internet users by defining profiles and roles.

Profiles enable Internet users to register themselves by automatically transferring a predefined set of privileges to the user when the first login is attempted.

Password authentication for intranets may be specified as OS for use with domain servers and authentication servers.

Session Management

Ingres/ICE uses cookies to identify individual sessions. The session identifier enables maintenance of session context between pages. Sessions have a configurable inactivity timeout that when reached rolls back any uncommitted transactions.

Storage Management

HTML Templates	Document content is made available through template files. In Ingres 2.5, access to the template files is abstracted to prevent exposure of queries and schema.
Document Cache Management	All files accessed within Ingres/ICE are subject to cache management, which specifies when the file may be closed or removed.

Macro Language Extensions

Macro language extensions in Ingres 2.5 include:

- User identification
- State maintenance
- Variables
- Conditional statements
- Variable testing using the IF or SWITCH macros to provide conditional flow within template pages

- Include statements
- The FUNCTION macro, providing a mechanism for invoking C callable shared libraries and dynamic link libraries

Visual DBA Features

Visual DBA features in Ingres 2.5 include:

- DOM windows—new design and features
- SQL/test—new design and features
- Star management
- Full Replicator management
- ICE management
- Enterprise access
- Miscellaneous new features

Ingres 2.5 Replicator Enhancements

Enhancements have been made to the internal performance that concern the generic replicator server and increased replicator concurrency.

Generic Replicator Server

Ingres 2.5 introduces a generic Replicator server that is functionally identical to the custom repserver built by the user in previous Ingres releases. The generic Replicator server can automatically handle any table that is configured by RepManager or Visual DBA without the need to compile or link a custom executable.

Increased Replicator Concurrency

In previous releases of Ingres, updates to the Replicator shadow, archive, and input queue tables performed by the DBMS as a result of a replicated user update would use the same isolation level as the original update (serializable by default). This isolation level is unnecessary, since the unique key value in the base table must have already been locked for the update to take place. In this release, the isolation level has been decreased to read committed, allowing for improved concurrency of replicated updates.

Features Introduced in Ingres 2.0

Ingres 2.0 introduced the following features and enhancements:

- Larger tuple support
- Distributed multi-cache management
- Enhanced performance of locking/logging and buffer manager
- Row level locking and transaction isolation levels
- Alter table support

Variable Page Size

Ingres 2.0 introduced support for multiple page sizes. The supported page sizes are 2 KB, 4 KB, 8KB, 16 KB, 32 KB, and 64 KB. The following semantic change has been made to DDL to support the variable page sizes. A `PAGE_SIZE` clause has been added to create table, create index, modify, and declare global SQL statements:

```
CREATE TABLE ..... WITH PAGE_SIZE=n;  
CREATE INDEX ..... WITH PAGE_SIZE=n;  
MODIFY TABLE ..... WITH PAGE_SIZE=n;  
DECLARE GLOBAL .... WITH PAGE_SIZE=n;
```

Valid page sizes are 2048, 4096, 8192, 16384, 32768, and 65536.

If the `WITH PAGE_SIZE` clause is omitted, Ingres uses the configuration parameter `default_page_size` to determine the page size. For compatibility reasons, `default_page_size` defaults to 2048 (2 KB). The different page sizes are not automatically available. The DBA must configure the buffer cache of the installation to a specific page size; otherwise an error occurs. See the *Database Administrator's Guide* for new buffer manager configuration parameters, IMA changes, and standard catalog interface changes.

New Page Format for Larger Page Size

Ingres 2.0 uses a different physical page format for larger page sizes. The page headers have been modified for larger page sizes to provide 64-bit TID support in the future. Additional features such as larger tuples, alter table support, and row level locking utilize the new page formats and are available only for larger page sizes. With larger pages, for each tuple a 24-byte tuple header is stored and the line table entry for larger page sizes has been increased to 4 bytes. See the *Database Administrator's Guide* and the *System Administrator's Guide*.

Larger Tuple Support

Ingres 2.0 supports larger tuples (up to 32 KB) with larger page sizes (64 KB). The availability of larger tuples is dependent on the availability of larger pages, for example, through buffer manager configuration. With larger pages, maximum tuple length of the installation can be configured by the configuration parameter `max_tuple_length`. If `max_tuple_length` is set to a larger value, DBMS server/Star server and Recovery server requires more memory. For compatibility reason, the `max_tuple_length` parameter defaults to 2008 bytes. A value between 1 to 2008 is not recommended. If the `max_tuple_length` is set to 0, `max_tuple_length` is dependent on the availability of the larger buffer manager.

Distributed Multi-Cache Management

Ingres 2.0 added support for Distributed Multi-Cache Management. You may use multiple servers using private buffer caches and fast-commit. The Distributed Multi-Cache Management exploits the Cluster Technology using a shared disk. An installation can use either shared cache or distributed multi-cache. No mixture is permitted. Unlike non-fast-commit servers, Distributed Multi-Cache Management allows DBAs to configure the servers using fast-commit. Distributed Multi-Cache Management should not be used in a single CPU machine or when a single (sole) server is used. A new configuration parameter, `dmcm`, has been added to configure Distributed Multi-Cache Management.

Enhanced Performance of Locking/Logging and Buffer Manager

Ingres 2.0 is optimized for performance using multiple mutexes to protect the internal critical data structures for the DBMS. Changes in the log records render all the previous checkpoints invalid. After you upgrade to Ingres 2.0, we recommend that you checkpoint all your databases for further recovery. Old checkpoint files from CA-Ingres 6.4 and CA-Ingres 1.2 cannot be rolled forward in Ingres 2.0.

Interval Based Deadlock Detection

Ingres 2.0 uses an interval-based deadlock mechanism to detect deadlocks. This also optimizes the contention within the Ingres lock manager and reduces CPU usage by the Ingres DBMS. Prior to Ingres 2.0, all locks caused a deadlock search when blocked. In Ingres 2.0, we check for deadlocks at a regular interval. A new special thread in the recovery server has been created to do this Interval Based Deadlock Detection. The interval for deadlock detection is controlled dynamically by the Ingres 2.0 server based on the server utilization.

Row Level Locking and Transaction Isolation Levels

Ingres 2.0 supports row locking for tables created with page size greater than 2 KB. The user may request row locking in cases where page locking causes unnecessary contention. Row locking is only available to fast-commit servers. Row locking cannot be used with distributed multi-cache (DMCM) servers. The set lockmode statement allows you to set the lock granularity, overriding the default locking strategy selected by the optimizer. The syntax for the set lockmode statement has been changed to support level=row.

```
SET LOCKMODE SESSION | on table  
WHERE LEVEL = row | page | table | session | system
```

Since it is difficult for an optimizer to determine when to use row locking, the decision must be made by the user with the set lockmode statement. The default lock granularity is page.

Row locks and intended page locks are counted per transaction, and escalation to TABLE locking occurs if this count reaches the maximum locks per transaction. Row locks and intended page locks are not counted for the MAXLOCKS per query. Ingres 2.0 supports four SQL-92 transaction isolation levels. The supported levels are serializable, repeatable read, read committed, and read uncommitted. The new SET TRANSACTION statement specifies the isolation level for a transaction. The syntax of the statement is:

```
<set transaction statement> ::= set transaction <isolation level>  
<isolation level> ::= isolation level <level of isolation>  
<level of isolation> ::= read uncommitted | read committed | repeatable read |  
serializable
```

The default isolation level is SERIALIZABLE. The default isolation level for a DBMS server may be configured by editing a new DBMS server configuration parameter, system_isolation. Acceptable values are serializable, repeatable_read, read_committed, and read_uncommitted. The isolation level for a session may be specified with a new SET SESSION option:

```
set session isolation level < level of isolation >  
<level of isolation> ::= read uncommitted | read committed |  
repeatable read | serializable
```

Alter Table Support

In Ingres 2.0, the alter table statement supports add and drop column functionality. The following semantic changes have been added to SQL to support the addition and deletion of columns from the existing tuple:

alter table

```
add [column]column_name format [null_clause]  
[default_clause] [column_constraint] |  
drop [column]column_name restrict | cascade
```

In this release, NOT NULL in the null_clause and WITH DEFAULT in the default_clause is not supported. Alter Table add/drop column is only available to page sizes larger than 2048 KB. Ingres 2.0 uses a versioning technique to support the above SQL-92 compliant alter table statement. For performance reasons, the statement does not update user data. The newly added column defaults to NULL value for the existing rows. Please note that alter table statement does not reclaim space for the deleted column(s). Please modify the table to reclaim the space for the deleted column(s).

Async I/O Support

On operating systems that support asynchronous I/O, Ingres allows the use of either I/O slaves or in-process asynchronous I/O, which may provide a performance improvement for those systems. The basic tenet for a DBMS (and RCP) server is that disk I/O should never cause the server to block while there is other work pending. To achieve this, Ingres has passed all disk I/O requests to separate processes (called *I/O slaves*), up to a limit of 30 per server, to do the actual disk operations. The slave processes perform the task synchronously and notify the server when it is completed. Each I/O operation using I/O slaves involves, among other things, a number of context switches from server to slave and back again. Context switching between processes is an expensive task when compared to context switching within a process. Performing the same non-blocking I/O within the server process provides a more efficient way to achieve non-blocking I/O.

Ingres 2.0 provides a new per-server configuration parameter called `async_io` that can be set in CBF to enable the in-process asynchronous I/O and disable I/O slaves. In a multiple-server installation, each server can be configured separately to use either slaves or asynchronous I/O.

Parallel Backup and Restore

Parallel backup and restore enhancements in Ingres 2.0 include:

- Parallel checkpointing to disk
- Parallel checkpointing to tape
- Parallel rollforwarddb from disk
- Parallel rollforwarddb from tape

Parallel Checkpointing to Disk

To checkpoint a multi-location database to disk in parallel, use the `#m` flag followed by the number of parallel checkpoints to be run:

```
ckpdb #m2 dbname
```

This saves two data locations at a time to the `IL_CHECKPOINT` location.

Parallel Checkpointing to Tape

To checkpoint a multi-location database to tape in parallel, list the devices to be used with the `-m` flag:

```
ckpdb -m/dev/rmt/0m,/dev/rmt/1m dbname
```

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

Parallel Rollforwarddb from Disk

To roll forward a multi-location database to disk in parallel, use the `#m` flag followed by the number of parallel restores to be run:

```
rollforward #m2 dbname
```

This restores two data locations at a time from the `II_CHECKPOINT` location.

Parallel Rollforwarddb from Tape

To roll forward a multi-location database from tape in parallel, list the devices to be used with the `-m` flag:

```
rollforward +c -m/dev/rmt/0m,/dev/rmt/1m dbname
```

The first location is restored from device 0m. The second location is restored from device 1m. The third location is restored from whichever device is finished first. The remaining locations is restored from the next free device. The operator is instructed to insert the numbered tape into the free device.

Fast Load Support

Ingres 2.0 supports a new fast bulk loading of data into a single table in a single database; the `fastload` command can load binary formatted files into database tables. In contrast to the `copy` statement, the `fastload` command is run from the command line and uses Ingres low level record management functions directly to load the data. It does so without the use of a connection to an Ingres DBMS, and thus avoids passing data through communications channels on the system. The user's process reads the input file and writes to the table in the database. The loading of some complex data types and the use of some table structures are not supported in all situations. See the *Database Administrator's Guide* for details.

R-tree Support: A Spatial Index for Ingres 2.0

This is a new access method for secondary indexes on ordered pair data. This is directed at spatial data types, but it can be extended to user-defined types. R-tree indexes can be created for any base table type: B-tree, hash, heap, and isam, using the create index command with structure=rtree syntax.

This access method provides lightning lookup to solve range queries and spatial joins for large numbers of objects. Record pointers are stored in spatial sequence (that is, consecutive objects in the index represent objects that are physically close). An R-tree index participates in the usual query optimization and is used with the operators inside, intersects, and overlaps.

Spatial Data Types and Operators

There are integer forms of point, box, lseg, line, polygon, and circle. Up to 249 ipoints can be used in an iline, versus 124 points in a line. Line, iline, polygon, and ipolygon are stored in compressed variable length form when "compression=data" is defined for the table. Spatial data types can be constructed from their base components. For example, mybox = box (llpoint, urpoint).

New data types, nbr and hilbert, can be used to order spatial data. A new overlaps operator determines if two objects have any points in common. Spatial operators support infix notation. For example:

```
select count(*) where property_location intersects pond-area;
```

Statement Level Rules

Prior to Ingres 2.0, row level rules support resulted in the invocation of the rules procedure for every row qualified by the triggering statement. This can entail a significant amount of overhead, particularly for auditing rules applications that may be interested in the number of rows touched rather than their actual contents. The statement level rules feature of 2.0 allows users to create rules that call the corresponding procedure exactly once for each execution of the triggering statement, rather than once for each row touched. In fact, the same information is still made available to the procedure as with row-level rules. For each qualifying row touched by the triggering statement, Ingres inserts a row into an internal temporary table. This row contains the data from the parameter list of the rule definition's execute procedure statement. The entire temporary table is then passed to the rule procedure for processing (see the Temporary Tables as Procedure Parameters section).

Temporary Tables as Procedure Parameters

Ingres database procedures currently have limited ability to process bulk information. This is largely the result of the restriction requiring that scalar parameters be passed between the caller and the procedure. Release 2.0 permits a global temporary table to be passed as a parameter to an Ingres database procedure. The temporary table can contain as many rows as desired upon entry to the procedure, permitting the procedure statements to process all the data with a single call. Likewise, the temporary table can be empty on entry to the procedure, to be filled by retrieval statements inside the procedure. The declaration of the temporary table inside the procedure allows it to be treated as any other Ingres table. It can be referenced in the “from” clause of a select or update statement and can be the target of an insert, delete, or update statement within the procedure. The potential effect of the feature is to reduce the number of procedure calls required to process a given quantity of data.

Other Optimizer Enhancements

Several changes were made to the Ingres optimizer in Release 2.0 to improve the quality of the generated query plans. The changes include the following:

- Reduction in size of generated query plans
- Improvement of join cardinality estimates
- Improvement of selectivity estimates for <> (not equals) restriction predicates

Following similar improvements in Release 1.2 of Ingres, changes were made to various internal query plan structures to further reduce the size of query plans generated by Release 2.0. The combined effect of the 1.2 and 2.0 changes reduces typical query plans by 40% to 60%. This frees QSF cache for use by more queries, resulting in significantly improved cache utilization.

A new statistic and new formulas have been introduced into Release 2.0 which should result in more accurate estimates of the number of rows produced by the join of two tables in an Ingres query. Since row estimates are a major component of the algorithms used to generate query plans, these enhancements should result in improved query optimization. The new statistic is a “per-cell” repetition factor. It reflects the average number of rows per distinct column value for EACH cell of a histogram, thus allowing more accurate join estimates. The new statistic is automatically computed by Release 2.0 `optimizedb` requests, and is displayed by Release 2.0 `statdump` requests. Moreover, Ingres 2.0 uses the default for prior release histograms, so that no statistics upgrade is required.

Ingres has always used histogram values to estimate the number of rows qualified by =, <, <=, >=, and > restriction predicates. As with the join cardinality estimates, these accurate row counts allow Ingres to build optimal query plans. Release 2.0 introduces histogram-based selectivity estimates for <> (not equals) restriction predicates, as well. This allows Ingres 2.0 to generate even better query plans.

Operating System Thread Support

A major enhancement to Ingres on operating systems that support operating-system threads is greater support for symmetrical multi-processor (SMP) machines. The database server takes full advantage of the ability of the operating system to balance processing time among all the available CPUs. By mapping database tasks to operating-system threads, individual sessions are dynamically balanced among available CPUs, providing greater throughput and scalability.

Table Cache Priorities

Database page cache priorities are normally assigned and managed by algorithms within the Ingres buffer manager. This addition allows tables to be assigned fixed priorities, which reduces the page replacement rate for those tables in a properly configured cache. The syntax for assigning fixed cache priority is:

```
CREATE TABLE table_name ... WITH PRIORITY = <cache_priority>
CREATE INDEX index_name ... WITH PRIORITY = <cache_priority>
DECLARE GLOBAL TEMPORARY TABLE ... WITH PRIORITY =
    <cache_priority>
MODIFY <table_name> | <index_name> TO PRIORITY =
    <cache_priority>
MODIFY <table_name> | <index_name> TO <storage_structure>
WITH PRIORITY = <cache_priority>
```

The TO PRIORITY ... form of MODIFY permits the assignment of a cache priority without having to also change the storage structure <cache_priority> must be an integer in the range 0 through 8, with 0 being the lowest and 8 being the highest priority. A specification of 0 causes the table to revert to normal cache management algorithms, and is the default value.

If an explicit priority is not set for an index belonging to a table to which an explicit priority has been assigned, the index inherits the base table's priority. HELP TABLE | INDEX <name> displays the cache priority.

Transaction Access Mode

A transaction's access mode may be defined by the syntax:

```
SET SESSION <session access mode>
<session access mode> ::=
    READ ONLY | READ WRITE
```

Rules:

- If unspecified, READ WRITE is implicit.
- `<session access mode>` remains in effect until changed by a subsequent SET SESSION `<session access mode>`.

```
SET TRANSACTION
<transaction mode> [ { <comma> <transaction mode> }... ]
<transaction mode> ::=
    <transaction access mode>
    <isolation level>
<transaction access mode> ::=
    READ ONLY | READ WRITE
<isolation level> ::=
    ISOLATION LEVEL <level of isolation>
```

See the Row Level Locking and Transaction Isolation Levels section for a discussion of `<isolation level>`. When specified, `<transaction access mode>` overrides `<session access mode>`.

Rules:

- SET TRANSACTION may not be issued within a transaction.
- If `<transaction access mode>` is not specified and `<level of isolation>` is READ_UNCOMMITTED, then READ ONLY is implicit; otherwise, READ WRITE is implicit.

When a `<transaction access mode>` of READ ONLY is in effect, database modifications (insert, update, delete, load, and ddl operations) are disallowed, and SQLSTATE of 25000 (invalid transaction state) is returned. Temporary tables are immune to this test and are always writable.

Soundex Function

Ingres 2.0 supports a new string function—SOUNDEX. The results of SOUNDEX are a char(4), and similar sounding words have the same SOUNDEX value. This is particularly useful when searching for names. For example:

```
SELECT fname, lname, addr1, addr2 from customer_table
```

```
WHERE SOUNDEX(fname) = SOUNDEX('kathy');
```

Note: The SOUNDEX function is case-insensitive.

Ingres 2.0 Star Features

Ingres 2.0 Star server supports variable page size and larger tuples as well as connectivity with previous CA-Ingres releases. To support variable page size, the standard catalog interface has been changed to include page sizes and other catalog changes. See the *Database Administrator's Guide* and the *System Administrator's Guide* for a description of the new standard catalog interfaces.

Ingres 2.0 Net Features

For information about Ingres 2.0 Net features, see the following sections.

Protocol Bridge Support

Ingres Protocol Bridge resolves the problem of connecting an Ingres client on one network to a server on a different type of network. Under the current architecture, a client and server must be able to communicate over the same network protocol (such as TCP/IP or SNA_LU62). The protocol bridge will “bridge” a client using one network protocol to a server using another. For example, a PC on a TCP/IP network could communicate through the protocol bridge to an EDBC gateway (DB2, IMS, CA-Datcom/DB, and so on) on an SNA network.

The protocol bridge provides communication capability between incoming connections and outgoing connections using a *different underlying protocol*. It listens for and accepts incoming connection requests and establishes corresponding connections to a local or remote communication server, allows bi-directional data transfer over the established connections, and terminates connections in an orderly way.

The protocol bridge does not provide any security checking when passing the messages through; security is still handled as usual on the server.

See the *Ingres System Administrator's Guide* for information on installing, starting and stopping, monitoring, and tracing the protocol bridge server.

Data-Stream Compression Support

Ingres 2.0 supports data-stream compression for variable-length data types, such as varchar and byte varying. Previously, when a client performed a COPY FROM or SELECT operation, the DBMS would send varchars at their maximum lengths, even if only a small fraction of the varchars contained usable data. In Ingres 2.0, the DBMS sends only the usable data. This is true for both local and network connections.

Database administrators may disable data-stream compression by invoking CBF, and setting the vch_compression parameter of the DBMS to OFF. This disables compression when the DBMS is re-started. Individual clients may disable compression by setting the environment variable II_VCH_COMPRESS_ON to N. Setting II_VCH_COMPRESS_ON to Y, or unsetting II_VCH_COMPRESS_ON, re-invokes data-stream compression only if the vch_compression parameter of the DBMS is ON.

SNA Duplex Support

In previous releases of Ingres/Net, a single SNA LU 6.2 conversation was used to support each client-server SNA connection. Since LU 6.2 conversations are inherently half-duplex, and Ingres transactions cannot be guaranteed to be, a large amount of overhead was created in order to maintain the flexible Ingres client-server dialog. This was a problem only with the SNA LU 6.2 protocol driver since the other protocols supported by Ingres/Net use full duplex protocol.

In this release of Ingres/Net, the overhead of maintaining the half-duplex nature of SNA LU 6.2 has been eliminated by using two conversations per client; one conversation stays in send state, the other in receive state. There are two important consequences of this implementation:

- Response time shows an approximate 50% improvement as the result of avoiding all request-to-send and prepare-to-receive calls.
- Only half the number of clients can be supported.

This feature can be controlled by a new environmental variable, II_HALF_DUPLEX. If this variable is set to 1, classic CA-Ingres/Net half-duplex operation is enabled. If this variable is undefined or set to 0, Ingres/Net 2.0 full duplex operation is enabled.

DECNet/OSI Support

On VMS systems, users may enter Phase-V-style node names, which may be greater than seven characters, for node addresses.

Ingres 2.0 OpenAPI

For information about Ingres 2.0 OpenAPI see the following sections.

Multi-Threaded OpenAPI

Ingres 2.0 OpenAPI supports multi-threading development and runtime environments for multi-threaded applications. Multi-threading support is platform-dependent. See the platform-specific Release Notes Supplement for further details. There is no external change required for multi-threaded OpenAPI support.

OpenAPI Support for Autocommit

Ingres 2.0 OpenAPI supports the SET AUTOCOMMIT statement through special autocommit transaction handles. Autocommit transactions are enabled/disabled through the new OpenAPI function, `Ilapi_autocommit()`. See the Ingres 2.0 *OpenAPI User Guide* for further details.

Enhanced OpenAPI Support for Database Events

Ingres 2.0 OpenAPI introduces enhanced support for database events with the `Ilapi_getEvent()` function. This function permits an application to wait for database event notification when client/server activity is idle. Previously, applications would only receive database event notifications when other non-event related client-server communications were occurring.

Ingres/ICE 2.0 Features

New Ingres/ICE features for Ingres 2.0 include:

- Support for the Microsoft Internet Information Server on Windows NT through a version of Ingres/ICE written for Microsoft ISAPI (Internet Server API)
- Support for NSAPI (Netscape API)
- Connection caching for ISAPI, ADI, and NSAPI interfaces, providing improved performance
- Requests and result pages can be up to 2 GB in size, subject to available disk and memory space
- Support for SQL select statements.

The result set of the select is automatically formatted as a table and displayed on the client browser.

- Support for BLOB data

Images stored as binary database objects may be selected and displayed on the client browser.

- A new HTML macro feature

Template HTML files that contain simple macros specifying SQL statements can be created. When Ingres/ICE loads the macro file, it replaces the macros with the result of the SQL statements, allowing data from different tables and databases to be combined on a single page.

- Improved security

Ingres/ICE interoperates with the security configuration of the target Web server. The HTTP basic authentication protocol is fully supported. This means that user IDs and passwords can be passed in an encoded form across the network.

- HTML-based installation and configuration utility

Visual DBA 2.0 Features

Visual DBA 2.0 differentiates between the following installations:

- Ingres 2.x installation
- CA-Ingres 1.x installation
- Local CA-Ingres/Desktop server
- Other installations

Users can connect and open simultaneously several windows displaying these installations. The ability to drag and drop tables is available between CA-Ingres 1.x, Ingres 2.x, and local CA-Ingres/Desktop server windows.

The new Visual DBA features for Ingres 2.0 installations include:

- Alter table support through Visual DBA 2.0
- Support of variable page size for Create Table, Create Index, and Modify commands, as well as the space calculations
- Management of table-level unique constraints and table-level check constraints both in Create and Alter Table
- References sub-dialog of Create Table is redesigned
- Support of table-level checkpoint and rollforwarddb

- Support for CA-Ingres Replicator versions 1.0/03, 1.0/05, 1.1, or 2.0

The following new features are also available for 1.x installations:

- Management of table level unique constraints and table level check constraints in create table
- Create table as select in Visual DBA 2.0 (with the Assistant to build the select statement)

Ingres 2.0 Server-Based Replication

The initial data capturing of Ingres/Replicator is performed inside the Ingres DBMS. The use of database rules and an application external to the DBMS to maintain the Replicator input and distribution queues has been superseded by internal DBMS functions and system threads. Any lock contention caused by previous versions of CA-Ingres Replicator has been eliminated, and Replicator data capture work is performed at a much lower level, producing more streamlined and faster performing replication.

The Ingres/Replicator product is delivered as part of the base Ingres 2.0 release, and the configuration and installation tools are integrated into the base Ingres setup tools. See the Ingres *Getting Started* guide.

Keywords

The following provides a complete table listing of Ingres keywords and indicates the contexts in which they are reserved. This list enables you to avoid assigning object names that conflict with reserved words.

Note: The keywords in this list do not necessarily correspond to supported Ingres features. Some words are reserved for future or internal use, and some words are reserved to provide backward compatibility with older features.

In the table that follows, the column headings have the following meanings:

NON 6.4	These keywords were not included in Ingres 6.4 keyword reserved lists.
ISQL	Interactive SQL. These keywords are reserved by the DBMS.
ESQL	Embedded SQL. These keywords are reserved by the SQL preprocessors.
IQUEL	Interactive QUEL. These keywords are reserved by the DBMS.
EQUEL	Embedded QUEL. These keywords are reserved by the QUEL preprocessors.
4GL	These keywords are reserved in the context of SQL or QUEL in 4GL routines.

Note: The ESQL and EQUEL preprocessors also reserve forms statements.

Reserved in:	NON SQL			QUEL			
	6.4	ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
abort		*	*	*	*	*	*
activate			*			*	
add		*	*	*			
addform			*			*	
after	*			*			*

Reserved in:	NON SQL			QUEL			
	6.4	ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
all		*	*	*	*	*	
alter		*		*			
and		*	*	*	*	*	
any		*	*	*	*	*	
append					*	*	*
array	*			*			
as		*	*	*	*	*	*
asc		*		*			
at		*	*	*	*	*	*
authorization		*	*				
avg		*	*	*	*	*	
avgu			*		*	*	
before				*			*
begin		*	*	*	*		*
bell	*		*	*			
between		*	*	*			
breakdisplay			*			*	
by		*	*	*	*	*	*
byref	*	*	*	*			*
call			*	*		*	*
callframe	*			*			*
callproc	*	*		*			*
cascade	*	*	*				
check		*	*	*			
clear			*	*		*	*
clearrow			*	*		*	*
close		*	*		*		
column		*	*			*	
command			*			*	

Reserved in:	NON SQL			QUEL			
	6.4	ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
comment	*			*			
commit		*	*	*			
committed	*	*	*				
connect			*	*			
constraint	*	*	*	*			
constraints	*	*					
continue		*	*				
copy		*	*	*	*	*	*
count		*	*	*	*	*	
countu			*		*	*	
create		*	*	*	*	*	*
current		*	*	*			
current_user	*	*	*				
cursor		*	*				
datahandler	*		*				
dbms_password	*		*	*			
declare		*	*	*			*
default	*	*	*	*			*
define	*	*			*		*
delete	*	*	*	*	*	*	*
deleterow			*	*		*	*
desc				*			
describe	*	*	*				
descriptor			*				
destroy					*	*	*
direct	*			*			*
disable	*	*		*			
disconnect			*	*			
display			*	*		*	*

Reserved in:	NON SQL			QUEL			
	6.4	ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
distinct		*	*	*			
distribute	*				*		
do		*		*			*
down			*			*	
drop		*	*	*			
else		*		*			*
elseif		*		*			*
enable	*	*		*			
end		*	*	*	*	*	*
end-exec	*		*				
enddata			*			*	
enddisplay			*			*	
endfor		*					
endforms			*			*	
endif		*		*			*
endloop		*	*	*		*	*
endrepeat		*					
endretrieve						*	
endselect			*				
endwhile		*		*			*
escape		*	*	*			
exclude	*				*		
excluding	*	*	*		*		
execute		*	*	*	*		
exists		*	*	*			
exit				*		*	*
fetch		*	*				
field			*	*		*	
finalize			*			*	

Reserved in:	NON SQL			QUEL			
	6.4	ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
first		*	*				
for		*	*	*	*	*	
foreign	*	*	*	*			
formdata			*			*	
forminit			*			*	
forms			*			*	
from		*	*	*	*	*	*
full	*	*	*	*			
get	*			*			
getform			*			*	
getoper			*			*	
getrow			*			*	
global	*	*	*	*			
goto			*				
grant		*	*	*			
granted	*	*	*	*			
group		*	*	*			
having		*	*	*			
help			*		*	*	
help_forms	*			*			*
help_frs			*			*	
helpfile			*	*		*	*
identified			*	*			
if		*	*	*			*
iimessage	*		*			*	
iiprintf	*		*			*	
iiprompt	*		*			*	
iistatement	*					*	
immediate		*	*	*			*

Reserved in:	NON SQL				QUEL		
	6.4	ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
import	*	*					
in		*	*	*	*	*	
include			*		*		
index		*	*	*	*	*	*
indicator			*				
ingres						*	
initial_user	*	*	*				
initialize			*	*		*	*
inittable			*	*		*	*
inner	*	*	*	*			
inquire_4gl	*			*			*
inquire_equel						*	
inquire_forms	*			*			*
inquire_frs			*			*	
inquire_ingres	*		*	*		*	*
inquire_sql			*	*			
insert		*	*	*			
insertrow			*	*		*	*
integrity		*	*		*		*
into		*	*	*	*	*	*
is		*	*	*	*	*	*
isolation	*	*					
join	*	*	*	*			
key	*	*	*	*			*
leave		*					
left	*	*	*	*			
level	*	*	*		*	*	
like		*	*	*			
loadtable			*	*		*	*

Reserved in:	NON SQL			QUEL			
	6.4	ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
local	*	*					
max		*	*	*	*	*	
menuitem			*			*	
message		*	*	*		*	*
min		*	*	*	*	*	
mode	*			*			*
modify		*	*	*	*	*	*
module	*	*					
move	*				*		
natural	*	*	*				
next			*	*		*	
noecho	*			*			*
not		*	*	*	*	*	
notrim			*			*	
null		*	*	*		*	*
of		*	*	*	*	*	*
off	*			*			*
on		*	*	*	*	*	*
only	*				*		*
open		*	*		*		
option		*					
or		*	*	*	*	*	
order		*	*	*	*	*	*
out			*			*	
outer	*	*					
param						*	
permit		*	*		*		*
prepare		*	*				
preserve	*	*	*				

Reserved in:	NON SQL				QUEL		
	6.4	ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
primary	*	*	*	*			
print			*		*	*	
printscreen			*	*		*	*
privileges		*					
procedure		*	*	*			*
prompt			*	*		*	*
public		*	*				
purgetable	*		*	*			*
putform			*			*	
putoper			*			*	
putrow			*			*	
qualification	*			*			*
raise	*	*		*			
range					*	*	*
read	*		*		*		
redisplay			*	*		*	*
references	*	*	*	*			
referencing		*		*			
register		*	*	*	*	*	*
relocate		*	*	*	*	*	*
remove		*	*	*		*	*
rename	*				*		
repeat		*	*	*		*	*
repeatable	*	*	*				
repeated			*	*			
replace					*	*	*
replicate	*				*		
restrict	*	*	*				
result		*	*				

Reserved in:	NON SQL			QUEL			
	6.4	ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
resume			*	*		*	*
retrieve					*	*	*
return		*		*			*
revoke		*	*	*			
right	*	*	*	*			
role	*	*	*	*			
rollback		*	*	*			
row		*	*				
rows	*	*	*				
run	*			*			*
save		*	*	*	*	*	*
savepoint		*	*	*	*	*	*
schema	*	*	*				
screen			*	*		*	*
scroll			*	*		*	*
scrolldown			*			*	
scrollup			*			*	
section			*				
select		*	*	*			
serializable	*	*					
session	*	*	*	*			
session_user	*	*	*				
set		*	*	*	*	*	*
set_4gl	*			*			*
set_equel						*	
set_forms	*			*			*
set_frs			*			*	
set_ingres	*		*	*		*	*
set_sql			*	*			

Reserved in:	NON SQL			QUEL			
	6.4	ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
sleep			*	*		*	*
some		*	*	*			
sort					*	*	*
sql		*					
stop			*				
submenu			*			*	
substring		*	*				
sum		*	*	*	*	*	
sumu			*		*	*	
system	*			*			*
system_user	*		*				
table		*	*	*			
tabledata			*			*	
temporary	*	*	*				
then		*	*	*			*
to		*	*	*	*	*	*
type	*			*			
uncommitted	*	*	*				
union		*	*	*			
unique		*	*	*	*	*	*
unloadtable			*	*		*	*
until		*	*	*	*	*	*
up			*			*	
update		*	*	*	*		
user		*	*	*			
using		*	*				
validate			*	*		*	*
validrow			*	*		*	*
values		*	*	*			

	NON SQL			QUEL			
Reserved in:	6.4	ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
view		*	*		*		*
when	*	*	*				
whenever			*				
where		*	*	*	*	*	*
while		*		*			*
with		*	*	*	*	*	*
work		*		*			
write	*				*		

Double Keyword	NON SQL			QUEL			
Reserved in:	6.4	ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
add privileges	*			*			
after field	*			*			*
alter default	*		*	*			
alter group		*	*	*			
alter location	*	*	*	*			
alter profile	*	*	*	*			
alter role		*	*	*			
alter security_audit	*	*	*	*			
alter table	*	*	*	*			
alter user	*	*	*	*			
array of	*			*			
before field	*			*			*
begin declare	*		*				
begin exclude	*		*				
begin transaction		*	*	*	*	*	*
by group	*			*			
by role	*	*		*			

Double Keyword	NON SQL			QUEL				
	Reserved in:	6.4	ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
by user	*	*			*			
call on	*				*			
call procedure	*				*			
class of	*				*			
clear array	*		*					
close cursor				*		*	*	
comment on	*	*	*	*				
connect to	*				*			
copy table	*				*			
create dbevent			*	*	*			
create domain	*		*					
create group			*		*			
create integrity	*	*			*			
create link			*	*				
create location	*	*	*	*				
create permit	*	*			*			
create procedure	*				*			
create profile	*	*	*	*				
create role			*	*	*			
create rule			*	*	*			
create security_alarm	*	*	*	*				
create synonym	*	*	*	*				
create user	*	*	*	*				
create view	*	*			*			
current installation	*				*			
define cursor						*		
declare cursor							*	
define integrity						*	*	*

Double Keyword	NON SQL			QUEL				
	Reserved in:	6.4	ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
define link							*	
define location						*		
define permit						*	*	*
define qry			*			*		*
define query			*			*		
define view						*	*	*
delete cursor						*	*	
describe form	*			*				
destroy integrity						*	*	*
destroy link							*	
destroy permit						*	*	*
destroy table							*	
destroy view	*							*
direct connect				*	*		*	*
direct disconnect				*	*		*	*
direct execute				*	*			*
disable security_audit	*	*	*	*				
disconnect current	*				*			
display submenu	*				*			*
drop dbevent			*	*	*			
drop domain	*			*				
drop group			*		*			
drop integrity	*	*			*			
drop link			*	*	*			
drop location	*	*	*	*				
drop permit	*	*			*			
drop privileges	*				*			
drop procedure	*				*			

Double Keyword	NON SQL				QUEL		
	6.4	ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
drop profile	*	*	*	*			
drop role		*	*	*			
drop rule		*	*	*			
drop security_alarm	*	*	*	*			
drop synonym	*	*	*	*			
drop user	*	*	*	*			
drop view	*	*		*			
each row	*		*				
each statement	*		*				
enable security_audit	*	*	*	*			
end exclude	*		*				
end transaction		*	*	*	*	*	*
exec sql	*		*				
execute immediate	*			*			
execute on	*			*			
execute procedure	*			*			
foreign key	*			*			
for deferred		*			*		
for direct		*			*		
for readonly		*			*		
for retrieve	*				*		
for update					*		
from group		*		*			
from role		*		*			
from user		*		*			
full join	*	*		*			
full outer	*	*		*			

Double Keyword	NON SQL			QUEL			
	6.4	ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
get attribute	*		*				
get data	*		*				
get dbevent	*		*	*			
get global	*		*				
global temporary	*			*			
help all	*		*				
help comment	*	*					
help integrity				*			*
help permit				*			*
help table	*	*					
help view				*			*
identified by	*			*			
inner join	*	*		*			
is null						*	
isolation level	*			*		*	
left join	*	*		*			
left outer	*	*		*			
modify table	*			*			
not like	*	*		*			*
not null	*					*	
on commit	*	*	*	*			
on current	*	*					
on database		*		*			
on dbevent		*		*			*
on location	*	*		*			
on procedure	*	*					
only where						*	
open cursor				*	*	*	
order by					*		

Double Keyword	NON	SQL			QUEL		
	6.4	ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
primary key	*			*			
procedure returning	*			*			*
put data	*		*				
raise dbevent		*	*	*			
raise error		*					
read only	*		*				
read write	*		*				
register dbevent		*	*	*			
register table	*						*
register view	*			*			*
remote system_password	*		*				
remote system_user	*		*				
remove dbevent		*	*	*			
remove table	*						*
remove view	*			*			*
replace cursor			*		*	*	*
resume entry	*			*			*
resume menu	*			*			*
resume next	*			*			*
resume nextfield	*			*			*
resume previousfield	*			*			*
retrieve cursor			*		*	*	
right join	*	*		*			
right outer	*	*		*			
run submenu	*			*			*
send userevent	*			*			
session group	*			*			

Double Keyword	NON SQL			QUEL				
	Reserved in:	6.4	ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
session role	*				*			
session user	*				*			
set aggregate	*	*				*		
set attribute	*			*				
set autocommit	*	*				*		
set cache	*	*				*		
set connection	*			*	*			*
set cpufactor	*	*				*		
set date_format	*	*				*		
set ddl_concurrency	*	*						
set deadlock	*	*				*		
set decimal	*	*				*		
set flatten	*	*				*		
set global	*			*				
set io_trace	*	*				*		
set j_freesz1	*	*				*		
set j_freesz2	*	*				*		
set j_freesz3	*	*				*		
set j_freesz4	*	*				*		
set j_sortbufsz	*	*				*		
set jcpufactor	*					*		
set joinop	*	*				*		
set journaling	*	*				*		
set lock_trace	*	*				*		
set lockmode	*	*				*		
set logdbevents	*	*						
set log_trace	*	*				*		
set logging	*	*				*		
set maxconnect	*	*				*		

Double Keyword	NON	SQL	QUEL				
Reserved in:	6.4	ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
set maxcost	*	*			*		
set maxcpu	*	*			*		
set maxidle		*			*		
set maxio		*			*		
set maxpage	*	*			*		
set maxquery	*	*			*		
set maxrow		*			*		
set money_format	*	*			*		
set money_prec	*	*			*		
set nodeadlock	*	*			*		
set noflatten	*	*			*		
set noio_trace	*	*			*		
set nojoinop	*	*			*		
set nojournaling	*	*			*		
set nolock_trace	*	*			*		
set nologdbevents	*	*					
set nolog_trace	*	*			*		
set nologging	*	*			*		
set nomaxconnect	*	*			*		
set nomaxcost	*	*			*		
set nomaxcpu	*	*			*		
set nomaxidle	*	*			*		
set nomaxio	*	*			*		
set nomaxpage	*	*			*		
set nomaxquery	*	*			*		
set nomaxrow	*	*			*		
set nooptimizeonly	*	*			*		
set noprintdbevents	*	*					

Double Keyword	NON	SQL			QUEL		
	6.4	ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
set noprintqry	*	*			*		
set noprintrules	*	*					
set noqep	*	*			*		
set norules	*	*					
set nosql	*				*		
set nostatistics	*	*			*		
set notrace	*	*			*		
set optimizeonly	*	*			*		
set printdbevents	*	*					
set printqry	*	*			*		
set qbufsize	*	*			*		
set qep	*	*			*		
set query_size	*	*			*		
set random_seed	*	*			*		
set result_structure	*	*			*		
set ret_into	*	*			*		
set role	*		*				
setrow deleted	*	*					
set rowlabel_visible	*		*				
set rules		*					
set session	*	*			*		
set sortbufsize	*	*			*		
set sql	*				*		
set statistics	*	*			*		
set trace	*	*			*		
set transaction	*		*				
set update_rowcount	*	*			*		

Double Keyword	NON	SQL			QUEL		
Reserved in:	6.4	ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
set work	*	*					
system user	*			*			
to group		*		*			
to role		*		*			
to user		*	*	*			
user authorization	*			*			
with null					*		
with short_remark	*	*					

Migrating to Ingres II 2.0 AXM on OpenVMS

This appendix describes the steps required for upgrading Ingres on the OpenVMS platform. This includes moving from Ingres 6.4 and OpenIngres 1.x to Ingres II 2.0/0011 (axm.vms/00).

This document will be updated to include the following topics:

- Inclusion of II 2.0 for VAX hardware
- Cluster Support
- Migrating Replicator configurations

This appendix is to be used in conjunction with the following:

- Ingres II 2.0 *OpenVMS Getting Started*
http://support.ca.com/techbases/ingres/ingres_manuals/ingres/docs.html
- Ingres II 2.0/0011 (axm.vms/00) *Release Notes* supplied with Ingres II OpenVMS CD-ROM

OpenVMS Requirements

The minimum process requirements for an Ingres II 2.0 System Administrator are documented in the “System Requirements” appendix of the *Ingres II OpenVMS Getting Started* guide for 2.0. For convenience, these have been reproduced in the following section.

UAF Process Quotas

UAF Name	Description	Recommended Starting Value
BYTLM	Buffered I/O Byte Count Limit	500,000
TQELM	Timer Queue Entry Limit	40
PGFLQUO	Paging File Quota (possibly set to the system virtual page count)	200,000

UAF Name	Description	Recommended Starting Value
DIOLM	Direct I/O Count Limit	200
BIOLM	Buffered I/O Limit	400
FILLM	Open File Limit	1000
PRCLM	Sub process Creation Limit	15 or larger
ASTLM	AST Queue Limit	200
ENQLM	Enqueue Limit (non-Cluster)	2048
JTQUOTA	Job Logical Name Table Byte Limit	20,000
MAXDETACH	MAX Detached Processing Limit	0 (infinity)

UAF Process Privileges

ALTPRI	PRMGBL	SYSLCK
CMKRNL	PRMMBX	SYSNAM
IMPERSONATE	READALL	SYSPRV
EXQUOTA	SHARE	TMPMBX
OPER	SYSGBL	WORLD

Installing Ingres II

The installation process for Ingres II has changed significantly from Ingres 6.4. Changes from OpenIngres 1.x are not so obvious; however, there are differences.

Ingres II uses the VMSinstal system to install and configure its software. Using VMSinstal, it is possible to create the new Ingres System Administrator account, extract all the software required, build the license area, and configure Ingres. However, depending on how the installation progresses, some issues may develop.

Mounting the CD

Ingres II can be installed either directly from the CDRROM or from a working area on the target system. If the files are transferred to the target node via FTP they will need to be moved across in binary mode. However, if the machine has a CD-ROM drive the following can be used to mount the CD:


```
$ MOUNT/OVER=ID/MEDIA=CD/UNDEFINED=(FIX:CR:32256) <CD Device>
```

To access the release notes use the following:

```
$ MOUNT/OVER=ID/MEDIA=CD <CD Device>
```

Running VMSinstal

Ingres can be installed from any 'privileged' account, defined as holding the privileges needed to run Ingres II. (See the UAF Process Privileges section for the required Ingres II privileges.)

To run the installer issue the command:

```
@sys$update:vmsinstal * distribution_medium
```

By default the SYS\$ROOT area is used by VMSinstal to unpack the savesets in preparation for installing Ingres II. If there is insufficient space available then VMSinstal will fail. To specify an alternate working directory the 'awd' parameter can be supplied to specify this alternate area:

```
@sys$update:vmsinstal * <distribution_medium> options awd=device:[dir]
```

To log the installation process specify the option L when calling VMSinstal:

```
@sys$update:vmsinstal * <distribution_medium> options L
```

Installation Types

Ingres II 2.0 has the ability to install itself based on a predefined package, or it can be customized by users to meet their particular needs. These installation types are classified within the VMSinstal Ingres II installer as Package and Custom installs.

Package Install

The Package Install option allows you to choose from six different configuration packages:

Feature Name	Product Name
Custom (Full) Installation	custominst
Ingres Networked DBMS server	dbmsnet
Ingres Networked Client	Netclient
Ingres Networked DBMS server with Star	dbmsstar

Feature Name	Product Name
Ingres RMS Gateway	rms
Ingres Stand-alone DBMS server	standalone

Custom Install

A custom installation allows for the specific needs of a site. The user is presented with the complete list of software available for Ingres. You choose the relevant components needed for the installation. Internally, Ingres generates the required files and dependencies for each component used in the installation.

Bridge	Protocol Bridge
c2audit	C2 Security Auditing
dbms	Ingres DBMS
esql	Embedded SQL Pre-compilers
net	Ingres/NET
oldmsg	Ingres 6.4 Message Files
ome	Object Management Extension
qr_run	Query and Reporting Runtime
qr_tools	Query and Reporting Tools
rep	Ingres/Replicator
rms	Ingres/RMS Gateway
spatial	Spatial Data Types
star	Ingres/STAR
tm	Terminal Monitor
vispro	Ingres/Vision and Ingres/ABF

Known Installation Issues

Note: For more information about these issues, contact Technical Support or check the technical documents available at the Technical Support web site, <http://support.com/ingressupp.html>.

Creating the Ingres System Administrator account from within VMSinstal does not assign the correct process quotas to the account. The quotas are defined in the OpenVMS Requirements section of this guide and the *Ingres II OpenVMS Getting Started* guide. The workaround is to create the account before the VMSinstal process is started with the correct privileges and process quotas.

II_WORK is not picked up, if pre-defined in the local symbol table, when an Express installation is performed. The user needs to enter in the correct information when prompted.

If Ingres is installed from a non-Ingres System Administrator account, imadb gets created as the process owner for VMSinstal rather than the installation owner configured earlier on. When Ingres II is started, the RMCMD process will hang as it is running as a user that is unable to connect to the RMCMD catalogs in imadb. The workaround is to install Ingres as the intended Ingres System Administrator.

Utility/Terminology Differences

Ingres II has several application and terminology differences when compared to Ingres 6.4. Many of the utilities supplied with Ingres II have remained the same in name and functionality. However, a number of these utilities have changed in name and behavior. This section describes those changes in detail. Note that for OpenIngres 1.x there are no changes as Ingres II uses the same set of utility names.

Ingres 6.4	Ingres II
iistartup	ingstart
iishutdown	ingstop
iibuild	vmsinstal
iisymboldef.com	ingusrdef.com
	ingsysdef.com
	ingdbadef.com
iijobdef.com	iijobdef.exe
netu	netutil
iirundbms.com	cbf
Production Installation	System Level Installation
Test Installation	Group Level Installation
ii_authorization	LicenseIT

Within Ingres II the logicals and symbols needed for a particular type of user have been broken into 3 classes. These are User, DBA, and System Administrator; respectively. Three scripts are used to set up the process environment for each:

ingusrdef.com – general Ingres user commands

ingdbadef.com – same as ingusrdef.com plus commands used by DBAs

ingsysdef.com – same as ingdbadef.com plus Ingres System Administrator commands

Each of the above scripts makes calls to the IIJOBDEF.EXE. To turn off the output from IIJOBDEF.EXE, define the logical IIJOBDEF_OUTPUT_OFF:

```
DEFINE /SYSTEM IIJOBDEF_OUTPUT_OFF TRUE
```

Configuration Differences

One major difference between Ingres 6.4 and Ingres II is the configuration facility. All configuration should be carried out directly through the Configuration By Forms (CBF) utility. This has several advantages over the previous method used within Ingres 6.4, in which configuration information was held in several files relating to the DBMS server, locking and logging, Ingres/Net, and Ingres/Star.

CBF provides a central configuration point with only a single user being able to access the configuration at any one time. In addition, access to and changes within CBF are recorded in a separate file, II_SYSTEM:[ingres.files]config.log. CBF stores and reads the configuration from II_SYSTEM:[ingres.files]config.dat.

Certain parameters are defined as being derived from the value of one or more other parameters. Ingres II uses several rule files to calculate these values based on changes made. For example, if there is an increase in the number of users connecting to the system, Ingres knows to increase the locking and logging segment available. These rule files are to be found in the II_SYSTEM:[ingres.files] directory:

ALL.CRS	Generic Entries
BRIDGE.CRS	Ingres Protocol Bridge
DBMS.CRS	Ingres DBMS Server
NET.CRS	Ingres/NET
RMS.CRS	Ingres RMS Gateway
SECURE.CRS	C2 Security
STAR.CRS	Ingres/STAR

TZ.CRS	Timezones
--------	-----------

To prevent derived parameters from being changed, they can be locked or protected. Additionally, the parameters and values protected from change are stored in a separate file `II_SYSTEM:[ingres.files]protect.dat`.

Terminal Key Maps

CBF uses its own key map file, located in `II_SYSTEM:[ingres.files]cbf.map`. By default, this maps to a VT100 function key terminal. To re-map CBF to a different keyboard, define the logical `II_CBF_KEYS` with a new map file. The map files supplied for Ingres II can be found in `II_SYSTEM:[ingres.files]` all with a `.map` extension.

To set up CBF to use the VT220 keyboard:

```
DEFINE II_CBF_KEYS II_SYSTEM:[ingres.files]vt220.map
```

Care must be taken to make sure that the `TERM_INGRES` logical is correctly set up to match the new map file.

Schema Checking

Ingres II reserves a number of new keywords, mostly for support of SQL additions implemented by Ingres II. If names such as "level," "key," or "comment" have been used as column names, changes to the database schema will need to be made. See the *Ingres II SQL ReferenceGuide*, Keywords appendix, for a complete list of Ingres II 2.0 reserved words.

A simple script has been provided that uses VMS DCL to parse a file for the existence of reserved words See the Ingres Keywords Parser section of this appendix for the source. An alternative method would be to take a copy of the schema from the current installation and try loading it into an Ingres II database.

These checks should also be extended to the applications to verify that tables and views created at runtime are not affected by the new keywords. Conflicts found in the schema and applications will need to be removed before moving to Ingres II.

Rebuilding Applications

In addition to migrating data, you need to rebuild all applications that connect locally to an existing Ingres installation.

As documented in the Ingres II 2.0 release notes supplement, the following compilers have been tested and are known to work with Ingres II 2.0 on OpenVMS:

DEC ADA	Compaq BASIC 1.4*
DEC C	DEC COBOL
DEC FORTRAN	DEC PASCAL
DEC PL/1	

* see the Known Compiler Issues section for more information.

Building Member_Aligned Against Ingres II 2.0 AXM

Note: This section applies only to Ingres II 2.0/0011 (axm.vms/00). This document will be updated to include the Ingres II 2.0 VAX build process.

With the move to a member_aligned version of Ingres, there is the need to rebuild applications. Only those that connect directly to an installation located on the same node, or through Ingres/NET using Ingres II 2.0 AXM on the client node need, to be rebuilt. Building Vision and ABF applications member_aligned is the default behavior for the current Ingres II 2.0 release with no further changes being required from the developer.

C	/MEMBER_ALIGNED
Pascal	/ALIGN=ALPHA_AXP
Cobol	/ALIGNMENT=PADDING
Fortran	/ALIGNMENT=ALL

There is the possibility that an application cannot be built using Alpha member alignment. In such cases, it is possible to rebuild those applications with the Ingres II components naturally aligned. Below are the steps needed for C and COBOL applications.

These changes require modification only to the Ingres II supplied files and should not involve any changes to the application code. Even performing the steps listed here, there is still the requirement to recompile **all** parts of the application that interface with Ingres II or that use any structures declared within the Ingres II header files.

By default, Ingres II is supplied so that any user applications will be built using the same compiler options used to build the Ingres II libraries and applications. Care must be taken if this is not followed.

The introduction of the member_aligned version of Ingres II 2.0 came about when alignment-related memory issues were encountered in Ingres II 2.0/9808 (axp.vms/00). If any applications are built using un-aligned structures with the communicating interface to Ingres II, data corruption is likely to occur.

For C Applications

To build C applications byte-aligned with Ingres II, a number of files require modification. Any modifications made may need to be re-applied following the installation of an Ingres II 2.0 patch.

The first files to modify are the C header files supplied in the following directories:

```
II_SYSTEM: [INGRES.DEMO.API.ASC]
II_SYSTEM: [INGRES.DEMO.UDADTS]
II_SYSTEM: [INGRES.FILES]
```

At this time, the only header files that contain Ingres structure definitions need modification, these are:

```
II_SYSTEM: [INGRES.DEMO.API.ASC]ASC.H
II_SYSTEM: [INGRES.DEMO.UDADTS]UDT.H
II_SYSTEM: [INGRES.FILES]ABFURTS.H,
II_SYSTEM: [INGRES.FILES]EQSQLCA.H,
II_SYSTEM: [INGRES.FILES]EQSQLDA.H,
II_SYSTEM: [INGRES.FILES]FRAME2.H,
II_SYSTEM: [INGRES.FILES]FRAME60.H,
II_SYSTEM: [INGRES.FILES]FRAME61.H,
II_SYSTEM: [INGRES.FILES]IIADD.H,
II_SYSTEM: [INGRES.FILES]IIAPI.H,
II_SYSTEM: [INGRES.FILES]OSLHDR.H,
II_SYSTEM: [INGRES.FILES]RAAT.H,
II_SYSTEM: [INGRES.FILES]SPATIAL.H
```

On the first line in each of the above files add:

```
#pragma member_alignment save
#pragma member_alignment
```

On the last line in each of the above files add:

```
#pragma member_alignment restore
```

Note: The "#" of the #pragma instruction *must* be the first character on the line.

The purpose of these pragmas is to direct the compiler to naturally align the elements of the defined structures, then to restore the alignment strategy used before the header file was included.

One further change is required to allow ABF and VISION applications to successfully build with unaligned code. In

```
II_SYSTEM:[INGRES.FILES]DCC.COM, replace the line
```

```
$ cc/standard=vaxc/float=ieee_float/nooptimize/nolist
```

with:

```
$ cc/NOMEMBER_ALIGNMENT/GRANULARITY=BYTE -  
/standard=vaxc/float=ieee_float/nooptimize/nolist
```

For COBOL Applications

To achieve the same result for embedded COBOL applications, the following statements must be added to these files:

```
II_SYSTEM:[INGRES.FILES]EQSQLCA.COB, II_SYSTEM:[INGRES.FILES]ESQLDA.COB
```

On the first line of the above files add:

```
*DC SET ALIGNMENT
```

On the last line of the above files add:

```
*DC END-SET ALIGNMENT
```

The II_SYSTEM:[INGRES.FILES]JUTCOM.DEF file requires the removal of the qualifier "/alignment=padding" from the COBOL compile statements.

Known Compiler Issues

Note: For more information about these issues, contact Technical Support or check the technical documents available at the Technical Support web site, <http://support.com/ingressupp.html>.

Compaq BASIC

On AlphaVMS, Ingres requires Compaq BASIC 1.4 for IEEE floating-point number support. If floating-point numbers are passed to and from an Ingres application within an earlier release of the BASIC compiler, the end result will be corrupted.

DEC ADA

Attempting to link an Embedded ADA application will result in the following link errors:

```
%LINK-W-NUDFSYMS, 1 undefined symbol:  
%LINK-I-UDFSYM,          IIG4SESENEVENT
```

Unable to pass string parameters from 4GL to embedded ADA (3GL).

Converting a Production System to a Group Level Installation

When migrating to Ingres II 2.0 you may wish to convert your existing installation from a system level installation to a group level installation.

Below are the steps you need to follow to convert a production/system level installation to a group level installation. It assumes that disk and location logicals are all defined at a system level and can still be read once Ingres has been converted to a group level installation.

Ingres 6.4

To convert an Ingres 6.4 production system to a group level installation:

1. Shutdown Ingres:

```
iishutdown
```
2. Run the DCL:

```
@ii_system:[ingres]iibuild.com
```
3. When prompted to 'Enter desired INGRES installation type':
 Enter **TEST**
4. Continue through the remaining prompts, accepting the defaults where relevant to your installation.
 When complete, the iibuild.com script will have updated your installation and its files to use the group level identifier.
5. To clean up, it is recommended that any definitions to II_SYSTEM at a system level be changed to group level from login script either at a local level (LOGIN.COM) or in the system level scripts (SYS\$MANAGER:SYLOGIN.COM, SYS\$MANAGER:SYLOGICALS.COM).
6. Reboot the system to make sure all the Ingres logicals have been cleared out.

OpenIngres 1.x

For OpenIngres, the conversion process requires a few extra steps. OpenIngres stores the location of the Ingres logicals in the II_CONFIG:CONFIG.DAT. In addition, the status of the installation for various OpenIngres components is located within the same file. Changes to the CONFIG.DAT file should be made using the OpenIngres utility *iisetres*. This program records all changes made to the config.dat into II_CONFIG:CONFIG.LOG.

1. Shut down the OpenIngres installation:

```
ingstop
```
2. To change the table identifier that OpenIngres uses, run the following:

```
iisetres ii.<nodename>.lnm.table.id LNM$GROUP
```
3. To rerun the DBMS setup script, the DBMS installation status needs to change from COMPLETE to READY. This can be performed as described below:

Current installation is for OpenIngres 1.1/04:

```
iisetres ii.<nodename>.config.dbms.oi1104axpvms00 READY
```

Current installation is for OpenIngres 1.2/00:

```
iisetres ii.<nodename>.config.dbms.oi1200axpvms00 READY
```

Current installation is for OpenIngres 1.2/01:

```
iisetres ii.<nodename>.config.dbms.oi1201axpvms00 READY
```

4. The next stage requires that the environment be set up for the new logical table and to complete the change from a system to a group level installation:

```
@II_SYSTEM: [INGRES]INGSYSDEF.COM
@II_SYSTEM: [INGRES.UTILITY]IISUBBMS.COM
```

5. To clean up, it is recommended that any definitions to II_SYSTEM at a system level be changed to group level from login script either at a local level (LOGIN.COM) or in the system level scripts (SYS\$MANAGER:SYLOGIN.COM, SYS\$MANAGER:SYLOGICALS.COM).
6. Reboot the system to make sure all the Ingres logicals have been cleared out.

Unloading and Reloading Data

With the member_aligned version of Ingres II 2.0, it is **not** possible to run *upgradedb*. The reason behind this is that the data structures used internally to Ingres II 2.0 do not follow the same format as earlier versions. The only way to migrate is to unload all the databases from the existing installation and reload them in to Ingres II 2.0.

If the existing installation is using the Compaq Alpha OpenVMS operating system, it is possible to perform the unload and reload of data using binary format files. If this is not the case, binary data files must not be used.

1. Run unloaddb against existing database in either binary or ascii modes

binary:

```
unloaddb dbname
```

ascii:

```
unloaddb -c dbname
```

2. Start unloading the tables from the existing installation:

```
@unload.ing
```

3. If time allows, it is preferable to run optimizedb against the new database after a reload has been performed. If time is short then the alternative would be to unload the statistics from the existing Ingres database. This is done using the statdump command as shown:

```
statdump -o dbname_stats.out dbname
```

4. Obtain a copy of the output from infodb for each database. This will be required when you need to extend the database in Ingres II prior to reloading. By default infodb sends its output to the terminal, using redirection the information can be sent to a file.

```
infodb dbname >dbname_infodb.out
```

5. The structures of the front-end catalogs for Ingres II have changed with an extra column being added in. To allow for this the reload scripts generated in stage 1 need to be modified to cope with the change.

For Ingres 6.4 edit the cp_ingre.in file located in your unload area and remove the line, (normally line 5):

```
\include II_SYSTEM:[INGRES.FILES]iiud64.scr
```

It is possible that the II_SYSTEM translates to a concealed root logical. In this case the concealed root logical will be seen instead of II_SYSTEM.

6. Using the information obtained from infodb create the disk areas to be used as data and work locations. The disk areas need to be created prior to their definition within Ingres II:

To create a data location:

```
create/dir device:[ingres.data] -  
/protection=(s:rew,o:rew,g:re,w:r)
```

To create a work location:

```
create/dir device:[ingres.work] -  
/protection=(s:rew,o:rew,g:re,w:r)
```

Once the areas have been created, they will need to be defined to the Ingres II installation. Use the *location* option in *accessdb* to define locations to Ingres.

7. Within the Ingres II installation, create the new database without any front-end catalogs. These will be created when the database is reloaded.

```
createdb dbname -fnofclients
```

8. For a database that uses multiple data locations, the database will need to be extended to use these locations. Either use *accessdb* to extend the database into those locations, or use the following sample SQL to extend a database to a new data location:

```
execute procedure iiqef_add_location  
(database_name='dbname',location_name='locationname',  
access=8,need_dbdir_flg=1)
```

9. Once extended, the database can be moved to Ingres II 2.0. With the *unloaddb* performed in stage1, a script is generated to reload the database into Ingres II.

```
@reload.ing
```

10. At this stage, the front-end catalogs are still using the previous structure from the source database. These now need to be upgraded to use the new structure for the catalogs involved. Depending on the use of the database and the applications used against it, at least one of the following commands will need to be run:

```
upgradefe dbname INGRES
upgradefe dbname VISION
```

If the source database is from Ingres 6.4 and the new database has journaling enabled, it is possible that the upgradefe will fail, displaying the following message:

```
E_DM0050 The table requested cannot be loaded
```

To allow the upgradefe to complete, disable journaling on the database by running:

```
ckpdb -j
```

and rerun

```
upgradefe dbname INGRES
```

11. Regenerate the statistics using *optimizedb* or reload the file generated in stage3. If new statistics are being generated, the minimum level of information generation should be against the keyed columns in each of the tables. If any queries use non-keyed columns, these should also be added at this stage. For syntax and further examples of *optimizedb* see the *SQL Reference Guide*.

Building New Statistics

```
optimizedb -zk dbname
```

Reloading Previous Statistics

```
optimizedb -i dbname_stats.out dbname
```

12. Once the database has been reloaded, it is recommended – though not required – that *sysmod* be run against the newly created database. This command needs to be run as the DBA of the database in the following manner:

```
sysmod dbname
```

13. The final stage is to back up the database and to (re)enable journaling. Once this is completed, the database should be ready for use.

```
ckpdb +j dbname
```

Migrating Users, Groups, and Roles

Since *iidbdb* cannot be upgraded, another method of retaining the metadata from Ingres 6.4 is needed. Ingres 6.4 stored its user information both externally and internally to the *iidbdb* database. The external file, located in `II_SYSTEM:[ingres.files]`, can be used to load the existing set of users into Ingres II.

As the installation owner, run the following:

```
sql "+U" -u$ingres iidbdb
```

Once inside the Ingres terminal monitor, issue the following:

```
Copy table iiuser (name=char(0)!', gid==char(0)!',
mid=char(0)')', status=char(0)nl) from
<ING64 II_SYSTEM>:[ingres.files]users.:\p\g
```

where <ING64 II_SYSTEM> is the II_SYSTEM for Ingres 6.4.

Migrating groups and roles requires some SQL since the information is only stored within iidbdb. The following applies to both Ingres 6.4 and OpenIngres 1.x.

As an Ingres system administrator on the Ingres 6.4 or OpenIngres 1.x installation:

```
sql -u$ingres iidbdb
```

Once inside the Ingres terminal monitor, issue the following SQL:

```
create table temp_iirole as select * from iirole\p\g
copy temp_iirole(
    roleid= varchar(0)tab,
    rolepass= varchar(0)tab,
    reserve= varchar(0)nl,
    nl= d1)
into 'temp_iirole.dat'\p\g

create table temp_iiusergroup as select * from iiusergroup\p\g
copy temp_iiusergroup(
    groupid= varchar(0)tab,
    groupmem= varchar(0)tab,
    reserve= varchar(0)nl,
    nl= d1)
into 'temp_iiusergroup.dat'\p\g

drop temp_iirole\p\g
drop temp_iiusergroup\p\g
\q
```

Within the Ingres II 2.0 Installation the above files need to be reloaded. As the Ingres System Administrator:

```
sql "+U" -u$ingres iidbdb
```

Once inside the Ingres terminal monitor, issue the following SQL:

```
copy iirole(
    roleid= varchar(0)tab,
    rolepass= varchar(0)tab,
    reserve= varchar(0)nl,
    nl= d1)
from 'temp_iirole.dat'\p\g

copy iiusergroup(
    groupid= varchar(0)tab,
    groupmem= varchar(0)tab,
    reserve= varchar(0)nl,
    nl= d1)
from 'temp_iiusergroup.dat'\p\g
\q
```

The users, groups, and roles should now be migrated.

Ingres II Licensing

The licensing process has changed in Ingres II 2.0. Versions prior to Ingres II 2.0 use a 32-character string to license the software being used. Ingres II 2.0 now uses a common licensing procedure for all CA products. Clients are supplied with an Omni-License File (OLF) that covers all CA software licensed for the site or the machine. After installing Ingres II for the first time, the user has a grace period of 31 days, which allows any upgrade to be completed and production work to commence.

There are two methods of licensing Ingres II 2.0, either by a network discovery or collecting the information required manually. With each copy of Ingres II a CD is supplied that contains LicenseIT, the licensing software. This appendix covers the manual process for situations where the network discovery technique for licensing Ingres II is not available. If the LicenseIT CD is not available, the software can be downloaded from <ftp://ftp.ca.com/CAproducts/License98/LicenseIT/>.

The LicenseIT CD contains all the software needed to perform the information gathering for a manual license request. Assuming the LicenseIT CD is mounted as CDROM. The OpenVMS software can be found in:

```
CDROM: [LICENSEIT.UTILITIES.VMS]
```

The following steps will install the licensing software needed to perform the manual license request:

1. Create the installation area and copy the LicenseIT saveset.

```
create/dir sys$specific:[ca_lic]
copy CDROM:[LICENSEIT.UTILITIES.VMS]calicutil010.a sys$specific:[ca_lic]
```
2. If calicutil010.a has been put on the node using FTP, run the following:

```
set def sys$specific:[ca_lic]
set file /attribute=(MRS=9216, LRL=9216, RFM=fix) calicutil010.a
```

Installing LicenseIT

To install LicenseIT:

```
@sys$update:vmsinstall calicutil010 * sys$specific:[ca_lic]
```

Some of the information needed for licensing is gathered using the CAminfo executable, obtained by following the above procedure. CAminfo provides information on the node pertaining to its MAC address (machine ID) and the class of hardware:

```
run sys$specific:[ca_lic]caminfo.exe
```

The output from CAminfo will be in the following format:

```
Node name:      WODNEY
Machine ID:     082004CF4CBE
Machine Type:   DEC_AlphaServer.1000.4/200_2_*
```

Once CAminfo has been run, additional information is required to complete the license request. Some of the information can be obtained from the results returned by CAminfo.

The following information is required:

Site ID
 MAC address (outputted from CAminfo)¹
 Machine type (outputted from CAminfo)
 Machine Description (for example, Compaq Galaxy Server GS160)
 Host name
 Manufacturer
 Model
 Number of CPUs
 Speed (MHz)
 License password²
 List of products to be licensed

¹ The MAC address is 12 characters long.

² The license password is automatically allocated to a site ID. If the password is not known, it can be obtained from the accounts team. The password is case sensitive, and is required when installing the license file.

Using the output from Caminfo, the following is an example based on the above template:

Site ID	0000000
MAC address	082004CF4CBE
Machine Type	DEC_AlphaServer.1000.4/200_2_*
Machine Description	DEC AlphaServer 1000
Host name	WODNEY
Manufacturer	DEC
Model	AlphaServer 1000
Number of CPUs	2
Speed (MHz)	200MHz
License password	xxxxxxx
List of products to be licensed	Ingres II DBMS Ingres II Replicator Ingres II STAR

Once collected, the information needs to be sent to help@licensedesk.ca.com.

The Total License Care group will then return an encrypted license file that will need decoding before installing on the target node. To install the OLF file, take the following steps:

1. Transfer the encrypted license file (CAE) to the target node. For the purposes of this exercise, the file will be called *license.cae*. For simplicity move/copy the file to *sys\$specific:[ca_lic]*.
2. Decrypt and install the license file:

```
set def sys$specific:[ca_lic]
run instalit.exe -f license.cae -p <password> -filter
<password> is the site password used in the request phase of the licensing process.
```

The license file is now installed to *sys\$specific:[ca_lic]ca.olf*.

Ingres Keyword Parser

A keyword parser for Ingres and OpenIngres follows. It can be used to scan the schema of an existing Ingres 6.4 or OpenIngres 1.x database. These scripts can be used to help identify potential problems before converting a database to Ingres II or before installing replication.

The scripts identify the following problems in a local or remote database:

1. Ingres II Reserved Words—Illegally named columns or tables that contains names that use an Ingres II reserved word. These columns or tables may need to be renamed before upgrading to Ingres II.
2. Replicator Reserved Words—Illegally named columns or tables that contain names that are Replicator reserved words. These columns or tables may need to be renamed before configuring replication.
3. Column Names too Long for Replication—Columns with names that exceed the Replicator limit of 28 characters. If these columns are to be replicated they must first be renamed.

To extract:

1. First verify that you can connect to the target database from the a UNIX shell/DCL prompt with:

```
sql dbname
```
2. If the target database is remote, create a local vnode registration and ensure that you can connect with:

```
sql vnode: :dbname
```

where 'vnode' is the remote vnode name 'dbname' is the name of the target database.

3. Once you have verified that the connection succeeds, execute '@db_check.com' with two parameters indicating the target database and userid.

Examples:

```
@db_check targetdb ingres
```

```
@db_check rmtvnode::targetdb mydba
```

Results will go to 'stdout' and to a file named 'dbcheck_report.txt'.

Tables owned by user '\$ingres' are excluded, as these should be only Ingres system catalogs.

Note: For more information and for the scripts that appear in the following list, see the Technical Support web site, <http://support.com/ingressupp.html>.

```
DBCHECK.COM  
HEADER.TXT  
FIND_RES_WRDS_SQL.TXT  
FIND_REPL_WRDS_SQL.TXT  
FIND_LONG_NAMES_SQL.TXT  
INGRESSINGLERESERVEDWORDS.TXT  
REPLICATORRESERVEDWORDS.TXT
```

These scripts are provided "as is" without warranty of any kind.

Index

4GL

4GL table_key type
conversions, 2-9

A

aggregate sort nodes
Ingres 2.6 enhancement, B-5

alter table support
Ingres 2.0 enhancement, D-4

ANSI/ISO constraint
Ingres 2.5 enhancement, C-3

applications
issues, 1-4
lifecycle, 1-4
planning, 2-2
preparation, 2-2, 2-6
rebuilding in Ingres II, F-8
upgrading, 3-12

archiver exit
shellscript, 2-12

arithmetic errors
greater sensitivity to, 2-8

async I/O support
Ingres 2.0 enhancement, D-5

auditdb
Ingres 2.6 enhancement, B-2

auto commit
OpenAPI support for, D-13

automated creation of location directories
Ingres 2.6 enhancement, B-4

AXM on OpenVMS
migrating from Ingres II 2.0, F-1

B

binary level support, 2-7

buffer manager performance
Ingres 2.0 enhancement, D-3
Ingres 2.6 enhancement, B-7

BYREF errors
greater sensitivity to, 2-8

C

case expression support
Ingres 2.5 enhancement, C-9

character data types maximum size
Ingres 2.6 enhancement, B-2

checkpoints
iiddb, 3-2
template changes, 2-11
upgradedb procedure, 4-3

composite histograms
Ingres 2.6 enhancement, B-6

concurrent rollback
Ingres 2.6 enhancement, B-5

conversions
4GL table_key type, 2-9

copydb utility
Ingres 2.6 enhancement, B-3

D

- database events
 - enhanced OpenAPI support for, D-13
- databases
 - checkpoint, 3-11
 - destroying, 3-4
 - extend, 3-9
 - moving, 2-4
 - recreate, 3-9
 - unload, 3-3
 - upgrading, 2-5
- datatype changes
 - user-defined, 2-9
- decimal constant
 - semantics changes, 2-7
- DECNet/OSI support, D-12
- default locations
 - record, 4-8
 - recording, 3-6, 4-8
- destroydb, 3-4
- distributed multi-cache management
 - Ingres 2.0 enhancement, D-2
- dmf sort
 - Ingres 2.5 enhancement, C-2
- dynamic write behind threads
 - Ingres 2.5 enhancement, C-4

E

- errors
 - arithmetic, 2-8
- Euro currency symbol support
 - Ingres 2.6 enhancement, B-9
- expression in group by list
 - Ingres 2.5 enhancement, C-10
- expression in order by clause
 - Ingres 2.5 enhancement, C-9
- extended date support
 - Ingres 2.5 enhancement, C-8

F

- fast load support
 - Ingres 2.0 enhancement, D-6
- FE reload script
 - fixing, 3-10
- front-end upgrade, 3-11

G

- gatherwrite threads
 - Ingres 2.6 enhancement, B-3

H

- hardware
 - implementation, 1-3
 - planning, 1-3
 - testing, 1-3
- hash joins
 - optimizer support for, B-6

I

- ICE development environment
 - Ingres/ICE enhancement, B-8
- iidbdb
 - cleaning, 3-4
- image file formats
 - OpenROAD 4.0, 2-13
- Import Assistant, B-4
- infodb
 - record output, 3-4
- ingprenv
 - record default locations, 3-6, 4-8
 - replaces ingprenv1, 2-11
- ingprenv1
 - replaced by ingprenv, 2-11
- Ingres
 - clean off, 3-6, 4-8
 - configure, 3-8

-
- disable startup, 3-5
 - installing, 3-7
 - keyword parser, F-19
 - loading development installation, 2-3
 - Net setup, 3-9, 4-11
 - record output configuration, 3-5
 - shutdown, 2-10, 4-4
 - start, 3-9
 - startup, 2-10
 - system backup, 4-4
 - upgrade development installation, 2-3
- Ingres 2.0 enhancements
- alter table support, D-4
 - async I/O support, D-5
 - buffer manager performance, D-3
 - distributed multi-cache management, D-2
 - fast load support, D-6
 - Ingres/ICE 2.0 features, D-13
 - interval-based deadlock detection, D-3
 - larger tuple support, D-2
 - locking/logging performance, D-3
 - new features, D-1
 - new page format, D-2
 - OpenIngres 2.0 Star features, D-11
 - OpenIngres multi-threaded OpenAPI, D-13
 - OpenIngres Net, D-11, D-12
 - OpenIngres Net data-stream compression support, D-12
 - OpenIngres Net protocol bridge support, D-11
 - OpenIngres Net SNA duplex support, D-12
 - OpenIngres OpenAPI, D-13
 - operating system thread support, D-9
 - optimizer, D-8
 - parallel backup and restore, D-5
 - parallel checkpointing to disk, D-5
 - parallel checkpointing to tape, D-6
 - parallel rollforwarddb from disk, D-6
 - parallel rollforwarddb from tape, D-6
 - row level locking, D-3
 - R-tree support, D-7
 - server-based replication, D-15
 - soundex function, D-10
 - spatial data types and operators, D-7
 - statement level rules, D-7
 - table cache priorities, D-9
 - temporary tables as procedure parameters, D-8
 - transaction access mode, D-10
 - transaction isolation levels, D-3
 - variable page size, D-1
 - Visual DBA 2.0 features, D-14
- Ingres 2.5 enhancements
- 64 bit file support, C-8
 - ANSI/ISO constraint, C-3
 - bit-wise operator support, C-7
 - case expression support, C-9
 - dmf sort, C-2
 - dynamic write behind threads, C-4
 - expression in group by list, C-10
 - expression in order by clause, C-9
 - extended date support, C-8
 - generic Replicator server, C-13
 - ICE 2.5 document cache management, C-12
 - ICE 2.5 macro language extensions, C-12
 - ICE 2.5 security, C-11
 - ICE 2.5 session management, C-12
 - ICE 2.5 storage management, C-12
 - increased concurrency, C-13
 - Ingres/Star, C-10
 - large cache support, C-4
 - large catalogs, C-8
 - miscellaneous new SQL functions, C-7
 - Net features, C-10
 - new features, C-1
 - new SQL functionality, C-7
 - optimizeddb, C-5
 - parallel sort techniques, C-3
 - partitioned transaction log file, C-5
 - qef sort, C-2
 - query optimization/execution, C-10
 - read-only database support, C-6
 - Replicator, C-13
 - row locking for system catalogs, C-8
 - sort, C-1
 - update mode locking, C-9
 - value locking protocol, C-9
 - Visual DBA, C-12
- Ingres 2.6 enhancements
- 64-bit operating systems, B-7
 - aggregate sort nodes, B-5
 - auditdb utility, B-2
 - automated creation of location directories, B-4
 - buffer manager performance, B-7
 - character sets for Euro currency symbol, B-9
 - composite histograms, B-6
 - concurrent rollback, B-5
 - copydb utility, B-3
 - gatherwrite threads, B-3
 - ICE development environment, B-8
 - increased character data type size, B-2
 - Ingres Import Assistant, B-4
 - Ingres Journal Analyzer, B-3
 - Ingres/ICE, B-8
 - internal performance, B-5
 - JDBC, B-8
 - locking system performance, B-6
 - logging system performance, B-7
-

- Microsoft transaction server support, B-5
- miscellaneous, B-6
- new features, B-1
- operating system integration, B-7
- optimizer support for hash joins, B-6
- preallocated rsb/lkbs, B-6
- raw location support, B-3
- rmcmd, B-5
- row producing procedures, B-1
- thread implementation on Linux, B-8
- unicode support, B-9
- usermod utility, B-2
- user-visible DBA, B-2
- user-visible language, B-1
- xml import/export utility, B-3

Ingres 6.4

- clean off, 4-8
- compared to Ingres II, F-5

Ingres II

- building C applications, F-9
- building COBOL applications, F-10
- compared to Ingres 6.4, F-5
- compiler issues, F-10
- converting from production system, F-11
- converting to group level installation, F-11
- installation issues, F-5
- installation types, F-3
- installing, F-2
- licensing, F-16
- rebuilding applications, F-8
- reloading data, F-13
- reserved words, F-7
- schema checking, F-7
- unloading data, F-13

Ingres II 2.0

- building member_aligned against AXM, F-8
- migrating to AXM on OpenVMS, F-1

Ingres versions

- Ingres 2.0, 1-2
- Ingres 2.5, 1-2
- Ingres 2.6, 1-2
- Ingres 6.4, 1-2
- OpenIngres, 1-2
- pre-Ingres, 1-2

Ingres/ICE

- Ingres 2.0 enhancements, D-13
- Ingres 2.6 enhancements, B-8

Ingres/ICE 2.5

- document cache management, C-12
- Ingres 2.5 enhancements, C-11
- macro language extensions:, C-12

- security, C-11
- session management, C-12
- storage management, C-12

Ingres/Star

- Ingres 2.5 enhancements, C-10

installation

- development, 1-3
- Ingres, 3-7
- Ingres development, 1-3, 2-3
- production, 1-3
- testing the upgrade, 1-3

internal performance

- Ingres 2.6 enhancements, B-5

interval-based deadlock detection

- Ingres 2.0 enhancement, D-3

J

JDBC

- Ingres 2.6 enhancement, B-8

Journal Analyzer, B-3

journaling ON

- default, 2-8

K

keyword parser, F-19

keywords, E-1

- double, E-11
- reserved, 2-2
- single, E-1

L

large cache support

- Ingres 2.5 enhancement, C-4

large catalogs

- Ingres 2.5 enhancement, C-8

LicenseIT, F-17

- installing, F-17

licensing

- Ingres II, F-16

locking system performance
Ingres 2.0 enhancement, D-3
Ingres 2.6 enhancement, B-6

logging system performance
Ingres 2.0 enhancement, D-3
Ingres 2.6 enhancement, B-7

login fixups, 3-6, 4-8

M

Microsoft transaction server support, B-5

migrating
groups, F-15
Ingres II 2.0 to AXM on OpenVMS, F-1
roles, F-15
users, F-15

N

Net features
Ingres 2.5 enhancements, C-10

Net setup, 3-9

netutil, 3-9

new SQL functionality
bit-wise operator support, C-7
Ingres 2.5 enhancements, C-7
miscellaneous functions, C-7

O

OpenAPI support
auto commit, D-13
database events, D-13

OpenIngres 2.0
DECNet/OSI support, D-12
multi-threaded OpenAPI, D-13
Net, D-11
OpenAPI, D-13
OpenAPI support for auto commit, D-13
OpenAPI support for database events, D-13
Star features, D-11

OpenIngres 2.0 Net
data-stream compression support, D-12

protocol bridge support, D-11
SNA duplex support, D-12

OpenROAD 4.0
image file formats, 2-13

OpenVMS (AXM)
migrating from Ingres II 2.0, F-1

OpenVMS requirements, F-1
UAF process privileges, F-2
UAF process quotas, F-1

operating, B-7

operating system thread support
Ingres 2.0 enhancement, D-9

optimizeddb
Ingres 2.5 enhancements, C-5

optimizer
Ingres 2.0 enhancements, D-8
reapply statistics, 3-11, 4-12
support for hash joins, B-6

P

parallel backup and restore
Ingres 2.0 enhancement, D-5

parallel checkpointing
to disk, D-5
to tape, D-6

parallel rollforwarddb
from disk, D-6
from tape, D-6

parallel sort techniques
Ingres 2.5 enhancement, C-3

parameters
UNIX kernel, 2-10

partitioned transaction log file
Ingres 2.5 enhancement, C-5

preallocated rsb/lkbs
Ingres 2.6 enhancement, B-6

procedures
moving databases, 2-4
run unloaddb, 4-2
upgradedb-object cleaning, 4-4

Q

- qef sort
 - Ingres 2.5 enhancement, C-2
- query optimization/execution
 - Ingres 2.5 enhancements, C-10

R

- raw location support
 - Ingres 2.6 enhancement, B-3
- read-only database support
 - Ingres 2.5 enhancement, C-6
- reload upgrade, 3-1
- reload.ing, 3-10
- Replicator enhancements
 - generic server, C-13
 - increased concurrency, C-13
- Report Writer
 - runtime parameter errors, 2-9
 - syntax change, 2-3
- reserved words, E-1
 - conflicts, 2-4
 - new, 2-2
- rmcmd
 - Ingres 2.6 enhancement, B-5
- row level locking
 - Ingres 2.0 enhancement, D-3
- row locking for system catalogs
 - Ingres 2.5 enhancement, C-8
- row producing procedures
 - Ingres 2.6 enhancement, B-1
- R-tree support
 - Ingres 2.0 enhancement, D-7

S

- scripts, F-19
 - fixing FE reload, 3-10
- search path
 - shared library, 2-10

- semantics changes
 - decimal constant, 2-7
 - UPDATE...FROM, 2-7
- server-based replication
 - Ingres 2.0 enhancement, D-15
- shared library search path, 2-10
- shellscripts
 - archiver exit, 2-12
 - monitoring system, 2-11
 - UNIX, A-1
- showrcp command, 3-5
- shutdown
 - Ingres, 2-10
- site modifications
 - preserve, 3-5, 4-7
 - restore, 3-8
- soundex function
 - Ingres 2.0 enhancement, D-10
- spatial data types and operators
 - Ingres 2.0 enhancement, D-7
- startup
 - disable Ingres, 3-5
 - Ingres, 2-10
- statdump shortcut, 3-4
- statement level rules
 - Ingres 2.0 enhancement, D-7
- storage structures
 - reapply, 4-12
- support
 - binary level, 2-7
- syntax
 - Report Writer, 2-3
- system administration
 - backup, 2-12
 - planning, 2-12
 - practice upgrade, 2-13
 - preparation, 2-9
 - restore, 2-12
 - testing, 2-12
- system monitoring
 - shellscripts, 2-11

T

- table cache priorities
 - Ingres 2.0 enhancement, D-9
- template changes
 - checkpoint, 2-11
- temporary tables as procedure parameters
 - Ingres 2.0 enhancement, D-8
- thread implementation on Linux
 - Ingres 2.6 enhancement, B-8
- transaction access mode
 - Ingres 2.0 enhancement, D-10
- transaction isolation levels
 - Ingres 2.0 enhancement, D-3
- transaction log size, 2-12
- tuple support
 - Ingres 2.0 enhancement, D-2
 - larger sizes in Ingres 2.0, D-2

U

- unicode support
 - Ingres 2.6 enhancement, B-9
- UNIX
 - shellsript, A-1
- UNIX kernel
 - parameters, 2-10
- unload directories
 - create, 4-2
- unload upgrade, 3-1
- unload/reload procedures, 3-1
 - checkpoint databases, 3-11
 - cleaning iidbdb, 3-4
 - cleaning off Ingres, 3-6, 4-8
 - configuring Ingres, 3-8
 - create unload directories, 3-2
 - creating work location, 3-7
 - destroying databases, 3-4
 - diasbling Ingres startup, 3-5
 - extending databases, 3-9
 - fixing FE reload script, 3-10
 - Ingres Net setup, 3-9
 - Ingres shutdown, 3-3
 - Ingres system backup, 3-3

- installing Ingres, 3-7
 - login fixups, 3-6
 - optional checkpoints, 3-2
 - preserve site modifications, 3-5
 - reapply optimizer statistics, 3-11
 - record infodb output, 3-4
 - record Ingres configuration, 3-5
 - recording default locations, 3-6, 4-8
 - recreating databases, 3-9
 - reloading, 3-10
 - restoring site modifications, 3-8
 - run unloaddb, 3-2
 - starting Ingres, 3-9
 - statdump shortcut, 3-4
 - unload databases, 3-3
 - upgrade applications, 3-12
 - upgrade front-end, 3-11
 - user check, 3-2
- unload/reload upgrade, 3-1
- unloaddb
 - output, 4-3
 - run, 3-2
- update mode locking
 - Ingres 2.5 enhancement, C-9
- upgrade
 - application issue, 1-1
 - applications, 3-12
 - hardware issues, 1-1
 - planning, 1-1
 - types, 1-2
 - unload/reload, 1-2, 3-1
 - unload/reload procedure, 3-1
 - upgradedb, 1-2
 - upgradedb procedure, 4-1
- upgradedb problems, 5-1
 - all to public grants not created, 5-3
 - cannot delete database, 5-1
 - duplicate key upgrading iidbdb, 5-1
 - failed, aborting creating internal procedure, 5-2
 - file extend conversion loop, 5-2
 - hang with open() Error 13, 5-2
 - iifile_info view not recreated, 5-3
 - product uninstallable, 5-1
- upgradedb procedures, 4-1
 - application upgrade, 4-12
 - checkpoint, 4-6, 4-12
 - clean iidbdb, 4-6
 - clean off Ingres 6.4, 4-8
 - configure Ingres, 4-11
 - create unload directories, 4-2
 - create work location, 4-9

- disable Ingres startup, 4-7
- disable user access, 4-3
- editing unloaddb, 4-3
- Ingres Net setup, 4-11
- Ingres shutdown, 4-4
- Ingres system backup, 4-4
- install Ingres, 4-9
- login fixups, 4-8
- optional statdump, 4-4
- preserve site modification, 4-7
- reapply optimizer statistics, 4-12
- reapply storage structures, 4-12
- record default locations, 4-8
- record infodb, 4-6
- record Ingres configuration, 4-7
- recreate objects, 4-11
- restore site modifications, 4-10
- run unloaddb, 4-2
- start Ingres, 4-10
- unloaddb checkpoint, 4-3
- upgradedb, 4-10
- user check, 4-2

- upgrading databases, 2-5
 - reloading, 2-5
 - unloading, 2-5

- user access
 - disable, 4-3

- user check, 3-2
 - upgradedb, 4-2

- usermod utility
 - Ingres 2.6 enhancement, B-2

- user-visible DBA
 - Ingres 2.6 enhancements, B-2

- user-visible language
 - Ingres 2.6 enhancements, B-1

V

- value locking protocol
 - Ingres 2.5 enhancement, C-9

- variable page size
 - Ingres 2.0 enhancement, D-1
 - new page format, D-2

- Visual DBA
 - Ingres 2.0 enhancements, D-14
 - Ingres 2.5 enhancements, C-12

- VMSinstal
 - running, F-3

W

- work location
 - create, 3-7, 4-9

X

- xml import/export utility
 - Ingres 2.6 enhancement, B-3