Ingres® 9.2

Migration Guide

INGRES

Contents

Chapter 3: Upgrading Using Upgradedb

Chapter 4: Upgrading Using Unload/Reload

Appendix A: Upgrading from Ingres 6.4   89

Appendix B: Keywords   119

Appendix C: Features Introduced in Ingres 9.2   143

Appendix D: Features Introduced in Ingres 2006 Release 2   155

Appendix E: Features Introduced in Ingres 2006   165

Chapter 1: Planning the Upgrade

This section contains the following topics:

This guide, when used with the other guides in the Ingres® documentation set, will assist in the planning and execution of a successful upgrade of Ingres.

This chapter describes how to plan for the upgrade, methods of upgrading, considerations for specific Ingres releases, required installations and hardware when upgrading, overall strategy for the upgrade, and application issues.

Use of New Features

After the upgrade is complete and running successfully for a suitable period, you can consider using the new features.

For a description of new features for the current release, see the *Release Summary*.

New features for past releases are described in appendixes in this *Migration Guide*.

The Upgrade Plan

The key to a successful upgrade is to prepare a detailed plan. A detailed plan can prevent problems when upgrading. The plan should include items such as how long it will take to complete a backup and how to verify that the data is complete and secure.

The plan should then be tested, preferably with a copy of the production system data. Testing reveals areas that may cause problems during the upgrade of the production system.

You should then implement the plan, but **only** after preliminary testing is complete.

The best strategy for upgrading is to first implement any compatibility fixes in the current environment. When the databases and applications are ready, test them in that environment, practice the upgrade, and then perform the upgrade.

Do not use any new features until the upgrade is successfully implemented. Doing so keeps to a minimum the number of variables at each step.

Upgrade Types

There are two options for upgrading your production systems:

- The upgradedb utility
- The unload/reload method

You can mix the two upgrade types, upgrading some databases while reloading others.

Upgradedb Method

The upgradedb utility allows for a fast, in-place upgrade path for an older version Ingres database, with no additional disk space requirements. Because upgradedb is faster, it is typically the recommended way of upgrading.

Preparing for a safe and reliable upgradedb, however, can take time, especially when upgrading from Ingres 6.4.

Databases using the system-maintained logical key feature are best upgraded using upgradedb. Tables that contain SYSTEM_MAINTAINED table_key or object_key columns cannot be safely unloaded and reloaded without additional work. The reload step generates all new logical key values. If there are other tables referencing the logical key columns, the new values must somehow be manually propagated to those other tables.

Unload/Reload Method

The database unload/reload method ensures a clean start with a fresh database. Depending on the kind of table data, additional disk space may be needed to perform the unloading and reloading; the space could be as large as three to five times the space of the database that is to be upgraded. For example, 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, it ensures a clean final installation.

A database that has been running for years, perhaps surviving a number of system crashes and hardware failures, may have suffered hidden damage that can confuse the upgradedb utility. 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. Such a database may have work tables owned by a user who no longer exists, or may be missing table data files. An unload/reload upgrade may be a better choice for this database.

The typical unload/reload upgrade uses the original Ingres installation as a base. The system databases iidbdb and imadb are upgraded in-place with upgradedb, even if user databases are unloaded/reloaded. A variation of the unload/reload method uses a brand new installation (perhaps even on a different machine). When this is done, additional work is needed to transfer iidbdb information (users, groups, roles, and database and installation privileges) to the new installation.

Upgrade Method and Ingres Releases

The release of Ingres that you are upgrading from can affect the type of upgrade you choose.

From Releases Prior to Ingres 6.4

If you are upgrading from a version of Ingres prior to release 6.4, you must use an unload/reload upgrade. Furthermore, you must install Ingres into a new, fresh installation; the original installation cannot be upgraded in-place.

From Ingres 6.4

If you are upgrading from Ingres 6.4, you can use either the upgradedb or the unload/reload method.

Ingres 2006 provides an improved upgradedb utility that allows 6.4 databases to be upgraded with minimal database preparation effort. You can follow the standard upgradedb procedure described in the chapter "Upgrading Using Upgradedb."

Since most 6.4 databases are several years old, you may choose to use an unload/reload upgrade. Keep in mind that an unload/reload upgrade takes substantially more time and resources. If you choose the unload/reload method, follow the procedures in Unload/Reload Procedure for Upgrading from 6.4 (see page 94).

Regardless of the method chosen, however, an upgrade from Ingres 6.4 requires more planning and preparation than upgrades from newer versions, and you must follow the application and system preparation procedures described in Considerations for Ingres 6.4 (see page 89).

An alternative upgradedb procedure that requires extensive preparation, but will result in a successful upgrade under almost any circumstances, is described in Alternate Upgradeb Procedure (see page 106).

From Releases Newer than Ingres 6.4

When upgrading from OpenIngres 1.2 or 2.0, Ingres II 2.0 or 2.5, or Ingres 2.6, we recommend the upgradedb method. The upgrade is internally much simpler than the upgrade from 6.4. In addition, there are fewer application-level incompatibilities among newer versions.

An unload/reload upgrade is possible, but is slower and requires more disk space than an upgradedb upgrade.

OpenIngres 1.2: If you are starting with OpenIngres 1.2, any tables having long varchar, long binary, or long spatial data must be unloaded under 1.x and reloaded into the new Ingres version. The format of the blob extension tables has changed. The remainder of the database can be upgraded with upgradedb; however, it is probably simplest to use a full unload/reload upgrade with any databases containing "long" datatypes.

Note: Upgrading from a pre-9.2 installation with character set setting II_CHARSETxx=UTF8 requires normalization of character data in non-Unicode columns. Contact Ingres Services to help with a migration plan.

From a 32-bit to a 64-bit Release

You can upgrade your 32-bit Ingres database for use with 64-bit Ingres by running the upgradedb utility. The 32-bit to 64-bit database conversion process redefines views, rules, integrities, and QUEL permits. The data in user tables is **not** affected by the 32-bit to 64-bit upgrade.

The upgradedb program does the following:

- Redefines the standard catalog views (iitables, iicolumns, and so on)

- Generates an SQL script to drop and redefine views, rules, integrities, and QUEL permits

- Executes the SQL script

The generated SQL scripts, and the SQL output, can be found in the directory $II_SYSTEM/ingres/files/upgradedb/UPGRADEUSER (where UPGRADEUSER is the user who is running the upgradedb program). There will be files DBNAME.i01 (SQL input) and DBNAME.o01 (SQL output). Depending on the specifics of the database, there might also be files DBNAME.g01 (grant inputs) and DBNAME.go01 (grant SQL output), and files DBNAME.r01 (referential constraint input) and DBNAME.ro01 (referential constraint output).

If your database contains an object that cannot be redefined, the upgradedb may fail to redefine all objects. You can use the SQL script and output in $II_SYSTEM/ingres/files/upgradedb to determine the point of failure. If necessary, contact customer support for assistance.

To Member-Aligned Alpha OpenVMS (axm.vms)

If you are using OpenVMS on Alpha hardware, and are upgrading to the member-aligned version of Ingres (axm.vms) from a non-member-aligned version (axp.vms), you must use unload/reload. Upgradedb is not available due to shifts in table data positions caused by the new alignment. For instructions, see the chapter "Considerations for Alpha OpenVMS."

Required Installations for Upgrading

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

- Original version production installation

- Original version development installation

- Installation for testing the upgrade

- New version development installation for preparing and testing applications

If possible, keep the installations away from the production machine. You may temporarily need additional hardware to accommodate the required installations during the upgrade.

Possible Hardware Setups for Upgrading

Possible hardware setups are from one to four machines.

A four-machine setup can be used, with each installation on its own machine. More commonly, however, the two development installations share a machine. Because there is usually some traffic between these two installations during preparation, sharing a machine is convenient.

Note: If you are using Windows, you need a separate machine for each installation. Versions prior to Ingres 2.6 do not support multiple installations on one Windows machine.

A three-machine setup is the recommended minimum, as follows:

- Development (both old and new versions)

- Test

- Production

A two-machine setup is possible, as follows:

- Development (possibly including a test installation)

- Production

The two-machine setup is not recommended because the test installation shares a machine with development, so it will not mimic your production installation as closely. In addition, the more installations on a machine, the more chance for error.

A single machine setup is possible, but not recommended, since you may accidentally work in the wrong installation and damage production.

Note: There is no remote installation procedure for Ingres. The machine must have local media support (CD-ROM or tape); otherwise, you will have to copy the distribution files from wherever the CD-ROM or tape drive is situated.

How You Perform the Upgrade

Note: Back up all data before starting.

The overall strategy for upgrading is as follows:

1. Copy databases and applications.

   Copy the databases to be upgraded into the new version development installation, and make a copy of all associated applications.

   It is important that your original version development installation remain; if the upgrade is unpredictably delayed, you will still have your original environment in which to fix mission-critical applications, if necessary.

2. Change databases and applications.

   Make any changes needed to the database definition or the application source code so that they function with the new version.

   If you are upgrading from a recent version (for example, from Ingres II 2.0 to Ingres 2.6), few or no changes are necessary. If you are upgrading from Ingres 6.4, this step can be lengthy.

   All compatibility changes will be reflected back into the original version development installation. Thus, if the upgrade is delayed for some reason, no work will be lost.

3. Test applications.

   Test your critical applications in the new-version development environment.

   Fix any problems or performance issues before your production upgrade. The fix will nearly always be compatible with your original version as well, and therefore can be reflected back into your standard development environment

4. Practice the upgrade.

   Practice the upgrade using the test installation. Ideally, the test installation should be a duplicate of production. Repeat the trial upgrade as often as necessary to achieve a trouble-free upgrade.

   Note: While practicing the upgrade, stop application development. You want the live upgrade to run exactly like the practice one, without involving new and untested factors.

5. Upgrade to the production system.

How You Plan for Application Issues

To ensure your applications can be tested in the new installation, do the following:

- Before starting the upgrade, take an application and database inventory. You must have the complete and current source code for all applications. If the source code does not match what users are running, problems can result.

- Make sure that each application can be rebuilt from the source code because you will eventually recompile your applications under the new version.

- If an application cannot be rebuilt, test the original executable under Ingres as soon as possible. If the application has no upward compatibility issues (for example, reserved words), it may be possible to run the old application against an Ingres installation and database. Otherwise, you will have to recreate the application or do without it.

- Try to synchronize the test and live Ingres upgrades with an appropriate time in the application life cycle.

  If application development is underway, plan how to coordinate new development with Ingres compatibility. Upgrades from newer versions (OpenIngres 1.2 or newer) may be able to move quickly enough to avoid the issue. Preparing an upgrade from Ingres 6.4 can take long enough to rule out a full stop in development.

  One site, for example, addressed the timing issue by synchronizing Ingres compatibility with a code release. Then, the development installation was converted to Ingres, while an Ingres 6.4 "bug fix" installation was maintained on a different machine.

The Test Plan for Applications

You must test your applications with the new version of Ingres before performing a production upgrade. The cost of testing every function in every application can be prohibitive, but fortunately, such testing is rarely necessary. A proper test plan can reduce testing time to a week or two.

A successful test plan uses the following process:

1. Rank the importance of each function in each application.

2. Test only the most important functions of each application.

3. Fix problems found after the upgrade as quickly as possible.

Categories of Application Functions

When determining the importance of an application function, ask "How long can we live without this function?" One successful testing approach divides application functions into three categories, as follows:

- Functions that are business critical, and must be operational immediately after the upgrade. No delay is permitted. Examples may include customer order entry, shipping, and production order release functions.

  Functions in this category must be tested thoroughly.

- Functions that are important, but the business can survive their loss for a few hours after the upgrade. Examples may include most inquiries and accounting functions, and high-visibility management reports, especially if management is made aware of the possibility of a one-time delay.

  Functions in this category should be tested as time and resources permit.

- Functions that can be broken for a day or two without serious impact. Examples may include reports, analysis functions, and end-of-period routines.

  Functions in this category can typically be spot checked.

Application Debugging After the Upgrade

If you properly execute your test plan, all critical functions will work after the upgrade. Less critical functions, however, may contain bugs. Be prepared to fix these bugs for a period after the upgrade. (Two weeks is usually long enough.) During this time, avoid scheduling new feature development. Have streamlined change control procedures ready, so that fixes can be installed quickly if a problem occurs.

Binary Level Support

Ingres 2006 provides support for applications built against previous versions of Ingres, back to and including Ingres 6.4.

You can run applications built with any version of OpenIngres 1.x or Ingres II 2.x, accessing an Ingres 2006 server, without rebuilding the application.

Applications built against Ingres 6.4 expect to access an older version of the Ingres message files. You can run applications built with Ingres 6.4 against an Ingres 2006 server, as long as either the application is running across Ingres Net, or Ingres 2006 was installed with the 6.4 message files. However, in all cases, we recommend that you rebuild all applications with Ingres 2006.

Chapter 2: Creating a New Ingres Development Environment

This section contains the following topics:

Purpose of this Chapter

This chapter describes how to move an existing development installation into a new development installation.

You should thoroughly test changes to databases and applications in the new development installation so that you can confidently perform the upgrade to the production system.

How to upgrade a production installation is described in subsequent chapters.

Note: If you are migrating from Ingres 6.4, read Considerations for Ingres 6.4 (see page 89) before performing the tasks in this chapter.

Platform-specific Examples in This Guide

While most of the examples used in this guide are specific to UNIX, the concepts can be applied to any environment in which Ingres runs.

For more information on upgrading in the VMS environment, see the appendix "Considerations for Alpha OpenVMS."

How You Move an Existing Development Installation into the New Development Installation

The steps for moving an existing development installation into a new development installation are as follows:

1. Create the new development installation.

2. Prepare applications for upgrades.

3. Load databases and applications into the new installation.

4. Prepare for development installation upgrade.

5. Test applications and procedures.

6. Practice the upgrade.

Create the New Development Installation

To install Ingres on the development machine, follow these steps.

Note: The following procedure assumes that the development computer will support both the original and new installations.

UNIX:

1. Create a new Ingres directory on a disk with sufficient space to hold the Ingres Home Directory. In this example, the directory is called /ing2006/ingres.

   Execute the following commands:

   ```
   mkdir /ing2006/ingres

   chmod 755 /ing2006/ingres
   ```

2. Create two scripts to set the environment to the original and new development installations. In this example, the scripts are named "setold" and "setnew."

   Note: These example C shell scripts may need to be adjusted for your specific installation. For example, the PATH settings may be different, and LD_LIBRARY_PATH may be named LIBPATH or SHLIB_PATH, depending on the platform.

In this example, the "old" installation is an Ingres 6.4 installation.

```
setold:
setenv II_SYSTEM /ing64
set path=(. /usr/local/bin /bin /usr/ucb /usr/sbin /usr/openwin/bin
$II_SYSTEM/ingres/bin $II_SYSTEM/Ingres/utility /usr/ccs/bin)
set inst=`ingprenv1 II_INSTALLATION`
setenv LD_LIBRARY_PATH /usr/lib:/usr/openwin/lib
set prompt=`whoami`.`uname -n`"[$inst]% "
echo "Switching to original Ingres 6.4 [$inst] installation"

setnew:
setenv II_SYSTEM /ing2006
set path=(. /usr/local/bin /bin /usr/ucb /usr/sbin /usr/openwin/bin
$II_SYSTEM/ingres/bin $II_SYSTEM/ingres/utility /usr/ccs/bin)
set inst=`ingprenv II_INSTALLATION`
setenv LD_LIBRARY_PATH /usr/lib:/usr/openwin/lib:$II_SYSTEM/ingres/lib
set prompt=`whoami`.`uname -n`"[$inst]% "
echo "Switching to new 2006 [$inst] installation"
```

3. If required, define aliases in the C shell or shell functions in the Bourne or Korn shell to invoke the setold and setnew scripts.

   For example:

   ```
   alias setold source ~ingres/setold
   alias setnew source ~ingres/setnew
   ```

4. Use the "setnew" alias to switch to the new Ingres environment, and then change directory to $II_SYSTEM/ingres.

5. Install a new Ingres installation by following the instructions in these Ingres guides:

   - *Installation Guide*

   - *Release Summary*

   - Readme

   - System-specific readme

   Note: The home directories can be on the same disks, but do not use the same data, checkpoint, journal, dump, or log directories for the old and new installations.

How You Prepare Your Applications

When moving a copy of applications and databases to the new installation, do the following:

- Resolve reserved word conflicts.

- Re-image Applications-By-Forms applications.

- Check for Report-Writer syntax change.

- Check for use of the ANSIDATE data type.

Reserved Keyword Conflicts

While the SQL parser recognizes most reserved keywords from context and usually resolves keyword conflicts without error, we recommend nevertheless that you avoid the use of SQL reserved keywords.

Check for and fix reserved word conflicts in databases and in application code (for example, in dynamic tables and views).

For a complete list of reserved words, see the appendix "Keywords."

Re-image ABF Applications

The introduction of column-level collation specification in Ingres 2006 caused the data descriptor that is compiled into imaged Applications-By-Forms (ABF) applications to change.

After upgrading, do the following:

1. Delete the contents of the ABF object directory, $ING_ABFDIR/*database-name*/*app-name*.

2. Re-image ABF applications.

Report-Writer Syntax Change When Upgrading from Ingres 6.4

When upgrading from 6.4, Report-Writer requires a space after all dot-commands.

For example, ".NL3" must be changed to ".NL 3".

UNIX: To fix such occurrences automatically, you can use the following "sed" commands:

```
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
```

Compare newfoo.rw against foo.rw to ensure that only the expected changes occurred.

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

Report-Writer Runtime Parameter Errors (UNIX)

The following problem occurs only with Report-Writer applications developed under Ingres 6.4.

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

If such an error occurs, the utexe.def command parameter settings can be changed back to that of Ingres 6.4.

To change the command parameter back to the Ingres 6.4 utexe.def settings

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

2. Search for the string "report report".

3. Within that section find the string "(%S)".

4. Change the string to: param '(%S)'.

5. Save the file.

6. Retest and see whether the error still occurs.

This problem may occur even if you are upgrading from a more recent version because typically the utexe.def file is replaced with every release of Ingres. Even if this issue was resolved during a prior upgrade, check to see if it has not reappeared in the latest upgrade.

Use of the ANSIDATE Data Type

In previous releases, Ingres supported one DATE data type that stored:

- Date

- Time

- Date and Time

- Time stamp

- Interval

As of Ingres 2006 Release 2, Ingres also supports the ANSI data types:

- DATE

- TIME

- TIMESTAMP

- INTERVAL

The configuration parameter date_alias controls whether the keyword DATE used for column data type refers to Ingres DATE or to ANSI DATE.

Note: This applies only to the data type of DATE and does not affect the other ANSI data types.

As client and server each interpret date_alias according to their own config.dat settings, we recommend that from Ingres 2006 Release 2, both the client and server have the same setting for date_alias.

When migrating from an earlier version of Ingres, the existing date data in the database is not affected as the data is in Ingres DATE format.

If date_alias is set to ANSIDATE, the DATE data type is now expecting data in the ANSI date format "YYYY-MM-DD." If the column still expects data in Ingres DATE format, the column format must be changed from DATE to INGRESDATE.

The same applies if date_alias is set to INGRESDATE and you want to use the ANSI DATE data type. The column must be declared as having a format of ANSIDATE and not DATE, as Ingres date format will be expected.

How You Load Databases and Applications into the New Installation

After creating a new installation, you are ready to move the databases from the old into the new installation.

This process consists of the following steps:

1. Create users, profiles, groups, and roles.

2. Move databases.

3. Move Ingres Star databases.

4. Fix the SYSTEM_MAINTAINED column names.

5. Compile applications in the new installation.

Note: It is assumed that the installations have identical Ingres Location names. If this is not correct, then it is necessary to edit the copy in files of unloaddb to change location names.

Create Users, Profiles, Groups, and Roles

After the new development installation is running, create the Ingres USERs, PROFILEs, GROUPs, and ROLEs. Depending on the version of Ingres, this may have to be done manually or from within the accessdb utility under "users" in the old installation.

- Use the "SQLscript" option to create an SQL file of users.

- PROFILEs, GROUPs, and ROLEs must be created manually.

- The SQL script will contain users that may already exist in the new installation, for example: ingres, system, or administrator. These users can be removed from the SQL script.

- Do not remove other users from the script without ensuring that the user does not own any database objects (tables, views, and so on) in any of the databases that are to be moved.

Look in the config.dat file (which can be found in the ingres\files directory under the Ingres home directory in the old installation) for lines that may have been included to give users any of the following privileges:

- SERVER_CONTROL

- NET_ADMIN

- MONITOR

- TRUSTED

The lines look like this:

```
ii.<node>.privileges.user.administrator:
SERVER_CONTROL,NET_ADMIN,MONITOR,TRUSTED
ii.<node>.privileges.user.any_user:
SERVER_CONTROL,NET_ADMIN,MONITOR,TRUSTED
```

These lines must be included in the config.dat file of the new installation.

Move Databases

Use unloaddb to upgrade a database from its original to a new installation.

Note: If the old and the new installations are on the same machine, it is necessary to have created a means of switching between the original and new installations. This section uses the setold and setnew scripts, which are described in Create the New Development Installation (see page 24).

Note: This section assumes that all objects within the database are to be copied from the original to the new installation. If, for example, the data is not wanted, the copy.in and copy.out scripts can be edited. The same applies to any other database object.

To upgrade a database from the old to the new installation

1. **Setold** and **cd** to a directory with enough space to hold the unloaded databases in the installation.

2. Create a directory for each database that is to be unloaded.

3. Change directory to the directory of the database that is to be unloaded.

4. Execute unloaddb against the database that is to be unloaded.

   Note: If the database is to be reloaded on a machine architecture that is different from the original (for example, OpenVMS to Windows), use unloaddb with the −c flag.

5. Unload the database by executing the resulting script:

   UNIX:

   Use "tee," or pipe as follows:

   ```
   unload.ing | tee /temp/unload.log
   ```

   or

   ```
   unload.ing > /temp/unload.log
   ```

   Windows:

   Pipe the output to a file, as follows:

   ```
   unload.bat > c:\temp\unload.log
   ```

   VMS:

   ```
   define/user sys$output unload.log
   ```

   ```
   @unload.ing
   ```

6. Review the output file for problems. Problems indicate that the database has not been unloaded correctly. Fix the problems and rerun the unload before continuing.

7. Edit the copy.in scripts as follows:

    For Ingres v6.4:

    Edit the cp_ingre.in file and remove the lines:

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

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

    Directory paths may be different

    For Ingres v1.2 and higher:

    Edit the copy.in file and remove the lines:

    ```
    \include /ing12/ingres/files/iiud.scr
    ```

    ```
    \include /ing12/ingres/files/iiud65.scr
    ```

    Directory paths may be different

8. Fix the system_maintained column name (see page 34) problem if upgrading to Ingres II 2.5.

9. **Setnew** to the new installation and create the new database. Check the copy.in scripts: if there are DROP statements before the CREATE TABLE statements for the system catalogs then:

    ```
    createdb databasename
    ```

    Otherwise use:

    ```
    createdb databasename –f nofeclients
    ```

10. If the database name is not the same as the original *databasename,* edit the reload script and change the *databasename*.

11. Run reload for the database. Capture the output of the reload script so that it can be checked for errors.

    UNIX:

    Use "tee," or pipe as follows:

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

    ```
    reload.ing > /temp/reload.log
    ```

    Windows:

    Pipe the output to a file, as follows:

    ```
    reload.bat > c:\temp\reload.log
    ```

    VMS:

    ```
    define/user sys$output reload.log
    ```

    ```
    @reload.ing
    ```

12. Review the output from the reload for any problems. Problems will mean that the database has not been correctly built.

  - If the errors can be corrected in the copy.in scripts, then fix them; otherwise go back to the original database and correct the errors.

  - Destroy the database in the new installation.

  - Restart the "Move Databases" procedure from the top.

13. If problems still occur, it may be necessary to contact technical support for assistance.

14. Having successfully moved the database, the front end catalogs in the database must be upgraded to the latest version. To do this run:

    ```
    upgradedb databasename -tree
    ```

    or

    ```
    upgradefe databasename INGRES
    ```

15. Review the output from the upgrade for any problems. The existence of problems means that the database has not been correctly built. Handle the problems as in Step 11 above.

Move Ingres Star Databases

For Ingres Star databases, use the same method as described above except that two databases must be unloaded and moved to the new installation:

1. Use unloaddb to unload the coordinator database (CDB) for the *stardatabasename*. This process unloads any locally stored tables.

   ```
   unloaddb ii stardatabasename
   ```

2. Use unloaddb on the distributed database (DDB). This process unloads registrations and distributed view definitions.

   ```
   unloaddb stardatabasename/star
   ```

Note: When registering a table in Star, it is necessary to declare a "with NODE = node_name". When moving to the new installation, it is necessary to either take across the existing Ingres Net vnode definitions or to manually create new ones that correspond to the existing vnode names; or having created new vnode names, edit the copy.in files.

The system_maintained Column Name

Databases created prior to Ingres II 2.5 may contain the system catalog ii_atttype, which has a column named SYSTEM_MAINTAINED. As of Ingres II 2.5 SYSTEM_MAINTAINED becomes a reserved word. At Ingres II 2.5, the column name SYSTEM_MAINTAINED was changed to SYS_MAINTAINED.

Loading a database that contains ii_atttype into version 2.5 will fail. Loading into Ingres II 2.6 and higher does not have the problem because they use a context sensitive keyword parser.

For reload to work with Ingres II 2.5, it is necessary to edit the copy.in script changing the column name in ii_attype from SYSTEM_MAINTAINED to SYS_MAINTAINED.

While this change can be made using a utility such as sed, beware of inadvertently changing the keyword SYSTEM_MAINTAINED in other places within copy.in.

Compile Applications

After databases have been successfully loaded into the new installation, applications must be compiled in this new environment.

- Keep a backup of application source code, libraries, and other important files.
- Ensure that compile scripts, linker command files, and similar files point to the new installation.
- Resolve compiler errors, syntax changes, and others.
- Test
- Resolve any differences between the old and the new installations.

Note: If you are upgrading from Ingres 6.4, check for the additional application issues under Considerations for Ingres 6.4 (see page 89).

How You Prepare for Development Installation Upgrade

To prepare a development installation for upgrade, follow these steps:

1. Back up installation.

2. Review Ingres monitoring tools and scripts.

3. Review third party products used.

4. Review checkpoint template changes.

5. Understand checkpoint and rollforward changes.

6. Define UNIX shared library search path.

7. Review UNIX kernel parameters.

Installation Back Up

When upgrading, it is important to have an off line system backup. If anything goes wrong, the installation can be restored from this backup.

The system administrator should know how back up and restore from the backup, either the complete system or just the Ingres parts of the installation.

After backup has been done:

- Verify the backup as readable, regardless of whether the backup was to tape or disk.

- Make no changes to the Ingres Installation.

- Store the backup in a secure location.

Ingres Monitoring Tools and Scripts

If there are tools or scripts that are used to monitor the Ingres installation (for example, Unicenter), they must be reviewed for compatibility with the new release of Ingres.

Check for these items, for example:

- On the UNIX platform are I/O slaves still used?

  OS-thread architectures such as Windows and Sun Solaris do not use I/O slaves.

- Does the tool parse iimonitor, logstat, lockstat, or trace point output?

  The detailed wording and positioning of logstat and lockstat output can change from release to release.

- Have log file name changed?

  For example, if upgrading from Ingres 6.4, the log files II_RCP.LOG and II_ACP.LOG are renamed to iircp.log and iiacp.log.

- Does the tool parse Ingres parameters?

  If upgrading from Ingres 6.4, the location of configuration parameters has changed and are now in config.dat and protect.dat.

- Are the tools and scripts provided by a third party?

  Contact the third party to find out if the tool or scripts must be upgraded to support the new version of Ingres.

- Do the monitoring tools use IMA (Ingres Monitoring Architecture)?

  There may have been changes to class IDs, which may affect the tools.

Third Party Products Used

If third party products interface to Ingres (for example, third party Replication tools or ERP applications), they must be reviewed for compatibility with the new release of Ingres. Contact the third party product provider to find out if the product needs to be upgraded to work with the new version of Ingres.

Checkpoint Template Changes

The Ingres checkpoint template file, cktmpl.def, may change from release to release. If you have customized your checkpoint template file, you must review and verify your changes with the new Ingres version.

If you are upgrading from Ingres 6.4, or from OpenIngres 1.2, you must redo your template changes. The cktmpl.def file format has been expanded since Ingres 6.4 and is therefore not compatible. The OpenIngres 1.2 template file format is similar to the current one, but additional entries are required. Your old checkpoint template can serve as a guide.

Tip: If your checkpoint template was customized to do multiple location checkpoints in parallel, you may be able to remove this customization entirely. Ingres supports parallel checkpoint and rollforwarddb processing directly.

If upgrading from an Ingres II release, compare the revised checkpoint template against the one installed with the new Ingres version. The template customizations may no longer be necessary due to new features in checkpoint and rollforward. For example: newer versions of Ingres support parallel checkpoint and rollforwarddb processing, so a checkpoint template customized to do multiple location checkpoints in parallel need not be done.

For more information on the format of the checkpoint template files, see the *Database Administrator Guide*.

Checkpoint and Rollforward Changes

Checkpoints and journals typically are not compatible from one version to the next. After an installation is upgraded, you must assume that all old checkpoints and journal files are no longer usable with the new version of Ingres.

Rollforwarddb no longer supports the -noblobs option because it makes the table physically inconsistent and unusable.

Shared Library Search Path (UNIX)

On some UNIX platforms, Ingres uses shared libraries. Since Ingres does not have a default installation directory, it is necessary to point applications and tools to where the Ingres shared libraries can be found. Doing so allows authorized users to access Ingres programs or applications.

The name of the shared library search path depends on the flavor of UNIX:

- Most UNIX environments use LD_LIBRARY_PATH

- HP-UX uses SHLIB_PATH

- AIX versions 3 and 4 use LIBPATH

See the ld(1) or ld.so(1) man page documentation.

To define the Ingres installation shared library search path

Set the library environment variable to include the Ingres library directory, $II_SYSTEM/ingres/lib.

Failure to set the library variable will result in the error message:

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

UNIX Kernel Parameters

Review the UNIX kernel parameter settings.

- If upgrading from Ingres 6.4, you may have to increase the size of a shared memory segment because Ingres builds a larger shared memory segment for locking and logging than it did in Ingres 6.4.

- If upgrading from a more recent version of Ingres, you may not have to change kernel parameters.

A 100 MB shared memory segment will accommodate most installations. Each platform modifies the shared memory limits in its own way, so read the platform-specific information in the Readme file. You may need to involve the system administrator.

It is a sensible practice to configure the new development installation in a similar way to the production installation.

Testing Applications and Procedures

In the new installation, you should test the following areas:

- Applications

- Performance

- System administrator procedures

Testing should not be random. You should have a test plan that includes the expected results. For example, if a query does mathematical calculations against a data set, the result should be manually computed so that the actual results can be verified against the expected results. Comparing the results from the previous version of Ingres with the new version of Ingres may give the same answer, but both could be wrong.

Application Testing

When upgrading, changes should be kept to a minimum. It is not sensible to include new application functionality into the upgrade cycle because this making comparing previous results with current results more difficult.

Changes should only be made where there are compatibility issues between versions of Ingres. Make the changes in the development environment, and then test according to the test plan. When testing, use data volumes and distributions that are as similar to the live data as possible.

Do not just test applications merely for data validity; also perform concurrency tests, load and volume testing, and housekeeping tests.

Performance Testing

Performance testing should be a major part of the upgrade test plan. Internal and configuration changes (for example, in query optimization and using OS threads) can cause query performance differences between Ingres versions.

Performance is new releases of Ingres should a worst be no slower than the pervious version of Ingres when run on the same hardware. At best, there should be performance benefits from upgrading to newer releases of Ingres.

Performance testing should always be done against data volumes that would be found on the production system.

If performance is found to be worse than the previous version, compare the table structures and indexes and QEPs from the slow queries. If there is no apparent difference, report the problem to Ingres Technical Support.

For more information on QEPs, optimizer statistics, and performance tuning, see the *Database Administrator Guide*.

System Administrator Procedure Testing

You should test the system administration procedures.

For example:

- Crash Ingres servers.
- Crash the system.
- Crash the hardware.
- Test backup and recovery procedures.
- Test any site specific customizations, such as changes to cktmpl.def.
- Test housekeeping routines.
- Understand any new Ingres features and functions.

How You Practice the Upgrade

You should run a trial upgrade as early as possible in the conversion cycle. Ideally, trial upgrades should be run more than once. Doing so helps identify and isolate problems, and also supplies timing data, which will be useful when the production system is upgraded.

A simplified upgrade process is as follows:

1. Run a trial upgrade.

2. Take notes on what went wrong or what should be done differently.

3. Continue running trial upgrades until no more problems are encountered.

4. Give the annotated upgrade procedure to someone who can verify the upgrade plan.

Note: Perform at least one of the trial upgrades on a full live data set, which will give an indication of how long the upgrade will take. This is particularly important when doing an unload/reload upgrade.

Chapter 3: Upgrading Using Upgradedb

This section contains the following topics:

This chapter describes how to use the upgradedb utility to upgrade from any version of Ingres.

Note: If you have difficulties upgrading from Ingres 6.4 with the upgradedb procedure in this chapter, you can use the Alternate Upgradedb Procedure in the appendix "Upgrading from Ingres 6.4."

Ownership Assumptions for Running Upgradedb

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 the installation owner, and use the -u{user} flag to pretend to be that user any time you have to run an Ingres command.

How You Upgrade Using the Upgradedb Utility

Upgrading using upgradedb transforms your database in-place from the original version to the new, without requiring an unload and reload.

To upgrade using upgradedb, use the following process.

Note: In this process, the notation **[Each DB]** means: "For each database, not including the iidbdb (master database), become the DBA for that database and perform this step." Do not include the iidbdb or Ingres Star distributed databases unless instructed. If using Ingres Star, remember to include the coordinator database in the list of databases.

1. Disable user access.

2. Disable Remote Command Server.

3. Shut down Ingres and back up system.

4. [Each DB including the iidbdb] Clean the database.

5. [Each DB] Record database information.

6. [Each DB including the iidbdb] Checkpoint and turn off journaling.

7. Shut down Ingres.

8. Preserve site modifications.

9. (Optional) Delete install directory (UNIX).

10. Install Ingres.

11. Create imadb database.

12. Restore site modifications.

13. Start Ingres.

14. Run upgradedb utility.

15. Review Ingres configuration.

16. (Optional) [Each DB] Reapply optimizer statistics.

17. [Each DB including the iidbdb] Checkpoint the database.

18. Install upgraded applications.

For details on each step, see the following sections.

Disable User Access

During the upgrade, the production system is not available for use. Make sure that users are not able to access the databases until the upgrade is complete.

Disable Remote Command Server

The Remote Command Server component of Visual DBA must be disabled for the duration of the upgrade. The Remote Command Server uses the iidbdb database as a communications mechanism in versions of Ingres prior to 2.6, so it will interfere with upgrading.

Note: If you are upgrading from early versions of OpenIngres 1.x, and you do not see an entry in CBF for the Remote Command Server, skip this step.

To disable the Remote Command Server

1. Run Configuration-By-Forms.

2. Locate the row for the Remote Command Server.

3. Note the startup count and record this value for later.

4. Use the EditCount function to set the startup count to zero.

Shut Down Ingres and Back Up System

You should perform a clean shutdown of Ingres, clearing all transactions from the transaction log, and then back up your system.

To perform a clean shutdown of Ingres

1. Shut down Ingres.

2. Restart Ingres.

3. Shut down Ingres again.

4. Check the recovery process log (iircp.log) for the message "RCP Shutdown completed normally."

To back up your system

1. Use a command appropriate to the platform to perform the backup.

2. Back up all Ingres directories, including data, checkpoint, journal, dump areas, and the $II_SYSTEM/ingres directory containing Ingres files and executables.

3. Back up the application directories

   Note: Watch for symbolic links and cross-mounts; make sure real data is saved and not a symbolic link.

4. Include the root file system in the backup if Ingres is typically started up at boot time. Alternatively, print a copy of any Ingres boot time startup and shutdown scripts.

5. Perform the backup twice to ensure that you have an extra copy of your backup. This step ensures maximum safety.

6. Check the backup media to ensure that the backup can be read. If your backup medium is tape, use new tapes, and clean the tape drive before the backup.

7. Restart Ingres.

Clean the Database

To ensure the integrity of the system catalogs, issue the following commands:

```
sysmod dbname
```

```
verifydb –mreport –sdbname dbname –odbms_catalog
```

The verifydb command may issue the following messages; you can ignore them.

```
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.
```

You can also ignore the "patch warning" message that warns of the loss of user tables in "runinteractive" mode. This mode will not be used.

If verifydb issues warnings or errors other than those in Step 1, review the messages with Ingres Technical Support before upgrading that database, because there may be damage to the system catalog.

Record Database Information

For each database, you will need to know information such as whether the database was journaled, where the database resides, and in what order the data locations were configured.

To record database information

1. Run infodb against each database. Issue the following command:

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

    Save the output for later.

2. Record whether the database is public or private.

    To find out, use the catalogdb command. Select Databases, and then enter the database name. The screen that appears has an Access field that indicates whether the database is public or private.

Checkpoint and Turn Off Journaling

For each database, including the iidbdb, checkpoint each database and turn off journaling. Then save the configuration file.

To checkpoint and turn off journaling

1. Checkpoint each database, using the ckpdb command with –j option to turn off journaling. (The upgradedb process turns off journaling, so it is best to do that now.) If upgradedb fails, you can use this checkpoint to recover and try again.

   Issue the following command:

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

2. Save the configuration file stored in the dump area after each checkpoint. The configuration file is small. Issue the following command:

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

Shut Down Ingres

Shut down Ingres with the ingstop command.

Preserve Site Modifications

Files distributed as part of Ingres that you have customized will be lost during the upgrade. Any custom files you have *added* to the $II_SYSTEM directory tree will remain.

You must copy your customized files to a safe place. Do *not* copy them to /tmp or anywhere in $II_SYSTEM/ingres directory.

If local collation sequence files have been customized, save the original collation definition files and the compiled files that reside in $II_SYSTEM/ingres/files/collation.

Commonly Customized Files

The following files are typically customized:

- Termcap files in $II_SYSTEM/ingres/files
- Keyboard map files in $II_SYSTEM/ingres/files
- Local collation sequence files

Preserve Necessary Files

If you cannot identify all your customized files, you can ensure that you preserve the necessary files by performing the following procedure. This procedure copies more files than necessary, but you can delete the copy after Ingres has been running live for a period.

To preserve necessary files

1. Delete all *.log files from $II_SYSTEM/ingres/files

2. Copy to a safe place the entire contents of the following directories:

   - all .opt files

   - $II_SYSTEM/ingres/bin

   - $II_SYSTEM/ingres/files

   - $II_SYSTEM/ingres/rep

   - $II_SYSTEM_ingres/files/rep

   - $II_SYSTEM/ingres/utility

Note: Do not delete the copy immediately when the upgrade completes, because you may discover weeks later that you need the old version of a file (for example, a Vision template or keyboard map) from the original $II_SYSTEM/ingres directory.

UNIX: On UNIX, to copy these files, use commands similar to the following:

```
cd $II_SYSTEM/ingres

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

Visual DBA Configurations

When upgrading, Visual DBA configuration files (.vdbacfg) are not upwardly compatible and must be recreated.

Note: Instead of using configuration files, you can use the vdba command with command line flags to start Visual DBA with, for example, certain windows open on given nodes. For details on the vdba command, see the *Command Reference Guide*.

(Optional) Delete Install Directory (UNIX)

Note: This step is optional but recommended.

The Ingres installation procedure on UNIX starts by extracting the install subdirectory from the Ingres distribution.

You should delete the old contents of that directory first, as follows:

```
cd $II_SYSTEM/ingres

rm -rf install
```

Install Ingres

To install Ingres, see the Ingres installation instructions for your platform.

During the installation process, the DBMS Server setup asks whether all databases are to be upgraded; answer **No**. The installation procedure automatically upgrades the iidbdb. If the upgrade of iidbdb fails, see the chapter "Troubleshooting Upgradedb." It is better to complete the Ingres setup, and then use the upgradedb command to upgrade the user databases.

If you are upgrading from 6.4, and the 6.4 installation has Ingres Star databases, you *must* respond **No** to this prompt. At this point in the 6.4 upgrade, the Star Server is not yet set up.

After the iidbdb is upgraded, the DBMS Server setup attempts to upgrade imadb and install Remote Command Server objects into imadb. Some versions of upgradedb neglect to create imadb first, and you will get "Database does not exist: imadb" errors. These will be corrected in the next step.

How You Upgrade to Older Versions That Require a Patch

Newer versions of Ingres distribute service packs. You can install service packs without having to install a base release of Ingres first.

UNIX: If you are upgrading to an older Ingres version that requires an overlay patch instead of a service pack, follow this procedure:

1. Run ingbuild. When asked whether you want to set up all the Ingres components, respond **No**. Exit ingbuild.

2. Install the Ingres patch.

3. Run ingbuild again. Select Current, then SetupAll.

4. Follow the prompts to complete the Ingres setup.

Setup now uses the fixed version.

Create imadb Database

Note: Perform this step only if you received "Database does not exist: imadb" messages during the DBMS setup phase of your Ingres install. This should only occur if you are upgrading from OpenIngres 1.x to Ingres 2.6 or older.

To create the imadb database, as the installation owner, execute these commands:

UNIX:

```
ingstart
cd $II_SYSTEM/ingres/bin
createdb '-u$ingres' imadb –f nofeclients
sql '-u$ingres' imadb <makimau.sql
rmcmdgen
ingstop
```

Windows:

```
ingstart
cd %II_SYSTEM%\ingres\bin
createdb -u$ingres imadb –f nofeclients
sql -u$ingres imadb <makiman.sql
rmcmdgen
ingstop
```

As the makimau or makiman SQL scripts run, you see a series of messages such as "E_US0AC1 '*some-name*' does not exist or is not owned by you." These are normal and can be ignored.

Note: For versions before Ingres 2006, the cd command in the above examples would be:

> **UNIX**:
>
> `cd $II_SYSTEM/ingres/vdba`
>
> **Windows**:
>
> `cd %II_SYSTEM%\ingres\vdba`

Restore Site Modifications

Refer to the save directory that was created in the step Preserve Site Modifications, and review any site-specific files that were overwritten by the upgrade.

Carry Forward Checkpoint Template Modifications

If the checkpoint template file cktmpl.def has been modified, the modifications may need to be carried forward into Ingres. Your original cktmpl.def should not be used directly, because entries can be added or revised in new versions of Ingres. Compare your customized cktmpl.def with the newly installed file, and make necessary changes in the new cktmpl.def. For information about the checkpoint template, see the *Database Administrator Guide*.

Start Ingres

Run ingstart to start Ingres.

Run Upgradedb Utility

Run the upgradedb utility to upgrade databases. You can upgrade databases one at a time or all at the same time. Log the upgradedb output to a file.

To upgrade one at a time:

```
upgradedb dbname
```

To upgrade all at the same time:

```
upgradedb –all
```

Example of logging upgradedb output to a file:

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

If errors occur, see the chapter "Troubleshooting Upgradedb." Correct the errors and rerun the upgradedb utility.

Review Ingres Configuration

The upgrade preserves your original Ingres installation parameters. You should review the configuration because some parameters may change from version to version. For information on parameters that changed, check the Readme for your new version of Ingres.

Review your parameter settings by running Configuration-By-Forms or Visual Configurator. Especially pay attention to major items such as startup counts and DBMS cache settings.

Note: If you disabled the Remote Command Server in an earlier step of the upgradedb process, use EditCount to restore its startup count to the original value.

(Optional) Reapply Optimizer Statistics

Note: This step is required only if upgrading from OpenIngres 1.x or Ingres 6.4. Ingres computes additional metrics that those releases did not have.

To take advantage of the new metrics, regenerate the optimizer statistics using the procedures of your application system.

Checkpoint the Database

Checkpoints and journals from your original Ingres version will not work with the newer version, so do not omit or delay this step.

Checkpoint each database, including the iidbdb. If the database was journaled previously, use the +j flag to turn on journaling.

To know which databases were journaled, see the infodb output from the step Record Database Information.

The iidbdb should always be journaled, regardless of whether it was journaled in the original installation.

Install Upgraded Applications

To perform the last step of the upgrade procedure:

1. Install the Ingres versions of the applications.

2. Restore user logins

3. Resume normal operation.

Chapter 4: Upgrading Using Unload/Reload

This section contains the following topics:

This chapter describes how to use the unload/reload procedure to upgrade from a post-6.4 version of Ingres.

Why Use Unload/Reload?

The unload/reload upgrade avoids the upgradedb program (except for iidbdb), in favor of unloading the original Ingres databases to flat files, recreating the databases under Ingres, and then reloading the databases. This approach has the advantage of starting with clean databases, but requires more time and disk space than does the upgradedb method.

Note: Databases using the system-maintained logical key feature are best upgraded using upgradedb. Tables that contain SYSTEM_MAINTAINED table_key or object_key columns cannot be safely unloaded and reloaded without additional work. The reload step generates all new logical key values. If other tables reference the logical key columns, the new values must be manually propagated to those tables.

Variations of Unload/Reload Procedure

The unload/reload procedure has two variations:

- The **in-place upgrade**, which replaces the original installation with the new Ingres installation. The master database (iidbdb) is upgraded with upgradedb, even though other databases are unloaded and reloaded. Because the iidbdb remains, all your locations, users, groups, and roles still exist in the new installation.

- The **clean install upgrade**, which leaves the original installation alone. Ingres is installed into a completely new installation. (The new installation may even be on a different machine.) When performing a clean install upgrade, you must take extra steps to recreate locations and move users, groups, and roles from the original installation to the new one.

How You Perform an Upgrade Using Unload/Reload

A database unload/reload ensures a clean start with a fresh database.

To perform an upgrade using unload/reload, use the following process.

Note: In this process, the notation **[Each DB]** means: "For each database, not including the iidbdb (master database), become the DBA for that database, **cd** to the unload directory for the database created in Step 1, and perform this step." If using Ingres Star, include the coordinator database in the list of databases. Steps that apply to a particular upgrade type only (that is, in-place upgrade or clean install upgrade) are marked accordingly.

1. [Each DB including iidbdb] Create unload directory.

2. [Each DB] Run unloaddb.

3. [Each DB] Check for obsolete users.

4. (Optional) [Each DB Including iidbdb] Checkpoint the database.

5. Disable user access.

6. Disable Remote Command Server.

7. Shut down Ingres and back up system.

8. [Each DB] Unload the database.

9. (Optional) [Each DB] Print optimizer statistics.

10. [Each DB] Record database information.

11. Record database privileges.

12. Save users, groups, and roles.

13. [Each DB] Destroy the database.

14. Clean iidbdb database.

15. Shut down ingres.

16. Disable Ingres startup.

17. Preserve site modifications.

18. (Optional) Delete install directory (UNIX).

19. Install Ingres.

20. Create imadb database.

21. Restore site modifications.

22. Review Ingres configuration.

23. Set up Ingres Net.

24. Start Ingres.

25. Recreate users, groups, and roles.

26. Recreate locations.

27. [Each DB] Recreate the database.

28. [Each DB] Extend the database.

29. Recreate database privileges.

30. [Each DB] Fix FE reload script.

31. [Each DB] Reload the database.

32. [Each DB] Upgrade front-end catalogs.

33. [Each DB] Reapply optimizer statistics.

34. [Each DB] Checkpoint the database.

35. Install upgraded applications.

For details on these steps, see the following sections.

Create Unload Directory

You should create a directory to hold scripts and data from the unloaded database.

Note: This directory requires a large amount of disk space. As an estimate, the unloaded data is about the same size as the Ingres database; however, compressed data can expand to take up much more space than the Ingres database.

To create a directory, issue the following commands for each database:

UNIX:

mkdir /*someplace*/*dbname*

chmod 777 /*someplace*/*dbname*

Windows:

mkdir d:*someplace**dbname*

Run Unloaddb

Run unloaddb against each database. The unloaddb command does not unload the database; it simply creates scripts.

For Ingres Star databases, unload the CDB in the same way as for a local database. For a DDB, use unloaddb/star.

For a regular DB or CDB, issue this command:

unloaddb *dbname*

For an Ingres Star DDB, issue this command:

unloaddb *ddbname*/star

If doing a clean-install upgrade to a different machine that has a newer architecture, binary data may not be compatible between the two machines. If this is the case, use the unloaddb **–c** option, which causes an ASCII instead of binary unload.

Check for Obsolete Users

Old databases may have objects created by users who no longer exist. Check for obsolete users for each database.

To check for obsolete users:

1. Examine the scripts created by unloaddb in the step Run Unloaddb of the upgrade procedure.

    Each script contains **set session authorization** SQL statements for each user who owns a database object.

2. Search for the **set session authorization** statements, and make sure that all users listed are valid.

3. Delete all the lines from the unwanted **set session authorization** statement up to the next one, if obsolete users are found.

4. Go into the database and clean out these unwanted objects.

(Optional) Checkpoint the Database

Note: This step is optional. You can omit this step if you can rely on the system backup to be taken in the later step Shut Down Ingres and Back Up System.

Follow these steps:

1. Checkpoint each database, including the iidbdb

2. Copy the checkpoint files to a permanent medium such as tape. Use fresh tape.

3. Verify that the tape can be read.

Disable User Access

During the upgrade, the production system is not available for use. Make sure that users are not able to access the databases until the upgrade is complete.

Disable Remote Command Server

The Remote Command Server component of Visual DBA must be disabled for the duration of the upgrade. The Remote Command Server uses the iidbdb database as a communications mechanism in versions of Ingres prior to 2.6, so it will interfere with upgrading.

Note: If you are upgrading from early versions of OpenIngres 1.x, and you do not see an entry in CBF for the Remote Command Server, skip this step.

To disable the Remote Command Server

1. Run Configuration-By-Forms.

2. Locate the row for the Remote Command Server.

3. Note the startup count and record this value for later.

4. Use the EditCount function to set the startup count to zero.

Shut Down Ingres and Back Up System

You should perform a clean shutdown of Ingres, clearing all transactions from the transaction log, and then back up your system.

To perform a clean shutdown of Ingres

1. Shut down Ingres.

2. Restart Ingres.

3. Shut down Ingres again.

4. Check the recovery process log (iircp.log) for the message "RCP Shutdown completed normally."

To back up your system

1. Use a command appropriate to the platform to perform the backup.

2. Back up all Ingres directories, including data, checkpoint, journal, dump areas, and the $II_SYSTEM/ingres directory containing Ingres files and executables.

3. Back up the application directories

   Note: Watch for symbolic links and cross-mounts; make sure real data is saved and not a symbolic link.

4. Include the root file system in the backup if Ingres is typically started up at boot time. Alternatively, print a copy of any Ingres boot time startup and shutdown scripts.

5. Perform the backup twice to ensure that you have an extra copy of your backup. This step ensures maximum safety.

6. Check the backup media to ensure that the backup can be read. If your backup medium is tape, use new tapes, and clean the tape drive before the backup.

7. Restart Ingres.

Unload the Database

For each database, run the unload.ing script created by the unloaddb command. The database is unloaded into your unload directory.

(Optional) Print Optimizer Statistics

Note: This step applies only to a clean-install upgrade.

Print optimizer statistics for each database. If your upgrade plan allows enough downtime to run a full optimizedb against your databases, you can omit this step. If your plan does not allow enough downtime, perform this step as a shortcut.

Note: Using this shortcut may result in some of the new Ingres metrics not being available; query performance may suffer until a full optimizedb can be completed.

If you are upgrading from OpenIngres 1.x, you should regenerate new statistics instead of saving the old ones, if possible.

To print the existing optimizer statistics, run statdump with the -o flag to a file for each database, as follows:

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

Record Database Information

For each database, you will need to know information such as whether the database was journaled, where the database resides, and in what order the data locations were configured.

To record database information

1. Run infodb against each database. Issue the following command:

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

   Save the output for later.

2. Record whether the database is public or private.

   To find out, use the catalogdb command. Select Databases, and then enter the database name. The screen that appears has an Access field that indicates whether the database is public or private.

Record Database Privileges

To record database privileges

1. As the installation owner, change directories to the unload directory for iidbdb created in Step 1 of the upgrade procedure.

2. Run the following SQL to save user database privileges:

```
sql iidbdb
\script dbprivs.out
select *
from iidbprivileges
where database_name <> ''
order by database_name,grantee_name
\go
\script
\quit
```

The file dbprivs.out is created for future reference.

Save Users, Groups, and Roles

Note: This step is required only for a clean-install upgrade.

To save users, groups,and roles

1. As the installation owner, change directory to the iidbdb unload directory created in Step 1 of the upgrade procedure.

2. Run the following SQL to save users, groups, and roles:

    ```
    sql iidbdb
    copy iiusergroup (
      groupid=c0comma,groupmem=c0nl
    ) into 'groups.out'
    \go

     copy iirole(
      roleid=c0nl
    ) into 'roles.out'
    \go

    create table role_tmp as
    select role_name,grantee_name
    from iirolegrant
    where admin_option <> 'Y'
    \go
    copy role_tmp(
      role_name = c0comma,
      grantee_name = c0nl
    ) into 'rolegrants.out';
    drop role_tmp;
    \go
     \quit
    ```

3. Run accessdb, and select Users, then SQLscript.

    A file called users.sql is written that will recreate all users, as they are currently defined.

    Note: SQLscript creates users only, and not the profiles, groups, or roles associated with each user. Roles and groups must be unloaded and reloaded for the script to generate the expected results.

Destroy the Database

Note: This step is required only for an in-place upgrade.

Destroy each database using the destroydb command.

Clean iidbdb Database

Note: This step is required only for an in-place upgrade.

To clean the iidbdb database

As the installation owner, run the following steps against the master database iidbdb:

Note: It is assumed that there are no objects created by users in the iidbdb.

```
sysmod iidbdb

verifydb -mrun -sdbname iidbdb -opurge

verifydb -mrun -sdbname iidbdb –odbms

ckpdb  –j  iidbdb
```

The verifydb command may issue the following messages, which you can ignore:

```
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.
```

You can also ignore the "patch warning" message that warns of the loss of user tables in "runinteractive" mode. This mode will not be used.

Disable Ingres Startup

If Ingres starts automatically when the machine boots up, turn auto-starting off until the upgrade is complete.

To disable Ingres startup and put operating system changes into effect

1. Follow the procedures for your platform:

   UNIX: On most UNIX platforms, a file in a system startup directory performs Ingres startup and shutdown; place an "exit 0" at the top of this file. The system administrator may need to perform this step if it requires root privilege. (The system startup directory depends on your platform—/etc/init.d, or /sbin/init.d, or a similar name).

   Windows: If Ingres is run as a system service, set the service to start manually instead of automatically.

2. Make sure that the operating system is correctly configured for your new version of Ingres, as described in How You Prepare for Development Installation Upgrade (see page 35).

3. Reboot, if necessary, to put the operating system parameter changes into effect.

Note: This step is recommended even if you are doing a clean installation upgrade. By leaving the old installation shut down, you eliminate the chance that someone will connect to it by mistake later.

Preserve Site Modifications

Files distributed as part of Ingres that you have customized will be lost during the upgrade. Any custom files you have *added* to the $II_SYSTEM directory tree will remain.

You must copy your customized files to a safe place. Do *not* copy them to /tmp or anywhere in $II_SYSTEM/ingres directory.

If local collation sequence files have been customized, save the original collation definition files and the compiled files that reside in $II_SYSTEM/ingres/files/collation.

Visual DBA Configurations

When upgrading, Visual DBA configuration files (.vdbacfg) are not upwardly compatible and must be recreated.

Note: Instead of using configuration files, you can use the vdba command with command line flags to start Visual DBA with, for example, certain windows open on given nodes. For details on the vdba command, see the *Command Reference Guide*.

(Optional) Delete Install Directory (UNIX)

Note: This step is optional but recommended.

The Ingres installation procedure on UNIX starts by extracting the install subdirectory from the Ingres distribution.

You should delete the old contents of that directory first, as follows:

```
cd $II_SYSTEM/ingres

rm –rf install
```

Note: This step is required only for an in-place upgrade on UNIX.

Install Ingres

To install Ingres, see the Ingres installation instructions for your platform.

In-place upgrades only: During the installation process, the DBMS Server setup asks whether all databases are to be upgraded; answer **No**. The install procedure automatically upgrades the iidbdb. If the upgrade of iidbdb fails, see the appendix "Troubleshooting Upgradedb."

After the iidbdb is upgraded, the DBMS Server setup attempts to upgrade imadb and install Remote Command Server objects into imadb. Some versions of upgradedb neglect to create imadb first, and you will get "Database does not exist: imadb" errors. These will be corrected in the next step.

How You Upgrade to Older Versions That Require a Patch

Newer versions of Ingres distribute service packs. You can install service packs without having to install a base release of Ingres first.

UNIX: If you are upgrading to an older Ingres version that requires an overlay patch instead of a service pack, follow this procedure:

1. Run ingbuild. When asked whether you want to set up all the Ingres components, respond **No**. Exit ingbuild.

2. Install the Ingres patch.

3. Run ingbuild again. Select Current, then SetupAll.

4. Follow the prompts to complete the Ingres setup.

Setup now uses the fixed version.

Create imadb Database

Note: Perform this step only if you received "Database does not exist: imadb" messages during the DBMS setup phase of your Ingres install. This should only occur if you are upgrading from OpenIngres 1.x to Ingres 2.6 or older.

To create the imadb database, as the installation owner, execute these commands:

UNIX:

```
ingstart
cd $II_SYSTEM/ingres/bin
createdb '-u$ingres' imadb –f nofeclients
sql '-u$ingres' imadb <makimau.sql
rmcmdgen
ingstop
```

Windows:

```
ingstart
cd %II_SYSTEM%\ingres\bin
createdb -u$ingres imadb –f nofeclients
sql -u$ingres imadb <makiman.sql
rmcmdgen
ingstop
```

As the makimau or makiman SQL scripts run, you see a series of messages such as "E_US0AC1 '*some-name*' does not exist or is not owned by you." These are normal and can be ignored.

Note: For versions before Ingres 2006, the cd command in the above examples would be:

UNIX:

```
cd $II_SYSTEM/ingres/vdba
```

Windows:

```
cd %II_SYSTEM%\ingres\vdba
```

Restore Site Modifications

Restore any site-specific files that you copied in the step Preserve 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 with Ingres, as the file format has changed. This means that you must recreate the changes using the Ingres 6.4 cktmpl.def as a guide. See the Ingres 6.4 *Database Administrator's Guide*.

If the archiver exit script acpexit was changed in Ingres 6.4, you must make the changes to the Ingres template (acpexit.def), and then move that file to $II_SYSTEM/ingres/files/acpexit.

Review Ingres Configuration

If you are doing a clean install, you need to change the default Ingres configuration to match your site requirements.

If you are doing an in-place upgrade, the upgrade process preserves your original Ingres installation parameters. You should review the configuration because some parameters may change from version to version. For information on parameters that changed, check the Readme for your new version of Ingres.

Review your parameter settings by running Configuration-By-Forms or Configuration Manager. Especially pay attention to major items such as startup counts and DBMS cache settings. If you are doing a clean install, you can use your original Ingres installation configuration as a guide.

Note: If you disabled the Remote Command Server in the step Disable Remote Command Server, use EditCount to restore its startup count to the original value.

Set Up Ingres Net

Create the vnode definitions for the remote installations by using netutil. If using shadow passwords on UNIX, you must run mkvalidpw. For details, see the *Connectivity Guide*.

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

Start Ingres

Run ingstart to start Ingres.

Recreate Users, Groups, and Roles

Note: This step is required only for a clean-installation upgrade.

To recreate users, groups, and roles:

1. As the installation owner, change directory to your iidbdb unloaddb directory where you stored the files from the step Save Users, Groups, and Roles of this upgrade procedure

2. Run this SQL to recreate users and groups:

    ```
    sql '-u$ingres' iidbdb
    copy iiusergroup(groupid=c0comma,groupmem=c0nl)
    from 'groups.out'
    \go
    commit
    \go
    \read users.sql
    commit
    \go
    \quit
    ```

    Windows: Omit the quotes from the sql command line.

    The file users.sql may try to recreate some users that already exist in the installation, such as the installation owner and root user. This will cause "E_US18B6 The user '*name*' already exists" errors. You can ignore these errors.

3. If your original installation had roles defined, recreate them with the ADD ROLE SQL statement. Use the file roles.out as a guide.

    Roles cannot be reliably bulk-loaded from the original installation, so you must recreate them by hand. After you recreate each role, issue the following SQL statement:

    ```
    grant rolename to user; commit
    ```

    The most common *user* here is **public**. You can use the file rolegrants.out to determine what role grants are needed.

Recreate Locations

Note: This step is required only for a clean-install upgrade.

To recreate locations:

1. Refer to **each** infodb output saved in the step Record Database Information of this upgrade procedure.

2. Create any location that is not a default installation location (ii_database, ii_checkpoint, ii_journal, or ii_dump). For more information about creating locations, see the *Database Administrator Guide*.

Recreate the Database

Before creating each database, refer to the infodb output saved in the step Record Database Information of this upgrade procedure. Look at the location names for ROOT, JOURNAL, CHECKPOINT, and DUMP. If these are not ii_database, ii_journal, ii_checkpoint, or ii_dump, you must specify the location to createdb with the –d, -j, -c, or –b flags, respectively.

Also, refer to the database access information recorded in that step. If the database access was "private," you must use the –p flag for createdb.

If all the database locations are the default, and the database is public, you can omit the flags on the createdb command line.

Recreate each user database, omitting the front-end catalogs. (The front-end catalogs will be created as part of the reload.) Use the following command:

```
createdb dbname flags -f nofeclients
```

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

Extend the Database

To extend each database:

1. Refer to the infodb output saved in the step Record Database Information of this upgrade procedure.

2. If the database was extended to data locations other than the default location, run accessdb as the installation owner 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.

   If you prefer a non-interactive command line utility, you can use the extenddb utility instead of accessdb.

Recreate Database Privileges

To recreate database privileges:

1. As the installation owner, change to the iidbdb unloaddb directory.

2. Refer to the file dbprivs.out created in the step Record Database Privileges.

    Each row in the dbprivs.out file describes one or more database privileges given to the user *grantee-name*. A Y or N in a privilege column indicates the specific privilege. (A U in a column means "Unchanged.")

3. Start an iidbdb Terminal Monitor session:

    ```
    sql iidbdb
    ```

4. For each row, issue the statement:

    ```
    grant privilege on database database-name to grantee-name;commit
    ```

    If the privilege column is N, grant *noprivilege* instead of *privilege*.

5. When finished, use **\quit** to exit the iidbdb session.

The structure of the iidbpriv catalog did not change between OpenIngres 1.x and Ingres 2.6, so it is possible to copy the original contents of the catalog directly. However, we do not recommend this because the catalog may change in future releases.

If you have defined many privileges, or recreated many users, groups, or roles, you should run sysmod on the iidbdb, which will accelerate query processing. Issue the sysmod command, as follows:

```
sysmod iidbdb
```

Fix FE Reload Script

Because the new database was not created with front-end catalogs, it is not necessary to drop them.

To fix the front-end reload script, for each database:

1. Open the file copy.in.

2. Delete the following lines:

   ```
   \include/ing12/ingres/files/iiud.scr
   \include/ing12/ingres/files/iiud65.scr
   ```

   Note: The directory path may differ.

3. Check for the *ii_atttype* catalog definition:

   ```
   create table ii_atttype (
   .
   .
   ...about 23 lines...
   .
   .
           system_maintained char(1) not null
   ```

4. Change the name system_maintained to sys_maintained.

   Not all databases contain the *ii_atttype* catalog, so it is okay if you do not find the definition.

5. Save the modified copy.in file.

Reload the Database

To reload the database:

1. Run reload.ing for each database.

   UNIX: Redirect the reload to a log file so that it can be checked for errors. Using the C shell:

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

   Note: If using Ingres Star, reload the CDB and all "real" local databases before reloading the DDBs.

2. After the reload is complete, verify that the table ii_id has only one row.

   Type **isql** <*database*>, and **select * from ii_id**.

3. If more than one row is returned, delete the row with the lowest object_id.

Upgrade Front-End Catalogs

To upgrade the front-end catalogs to the new Ingres level, run upgradefe on each database.

Issue the following command:

```
upgradefe dbname INGRES
```

Type the word INGRES in uppercase.

Reapply Optimizer Statistics

Reapply optimization statistics for each database. You can do this by either:

■ Regenerating statistics from scratch.

If there is sufficient time, we recommend that you regenerate the optimizer statistics using the procedures of your application system.

■ Using the statistics printed from the original installation in the step Print Optimizer Statistics earlier in this upgrade procedure.

If time is short, and if you printed the original statistics, you can read them back in with the –i option to optimizedb:

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

Checkpoint the Database

Checkpoints and journals from your original Ingres version will not work with the newer version, so do not omit or delay this step.

Checkpoint each database, including the iidbdb. If the database was journaled previously, use the +j flag to turn on journaling.

To know which databases were journaled, see the infodb output from the step Record Database Information.

The iidbdb should always be journaled, regardless of whether it was journaled in the original installation.

Install Upgraded Applications

To perform the last step of the upgrade procedure:

1. Install the Ingres versions of the applications.

2. Restore user logins

3. Resume normal operation.

Chapter 5: Troubleshooting Upgradedb

This section contains the following topics:

How to Avoid Upgradedb Problems

The best way to avoid problems with the upgradedb utility is to upgrade to the most recent service pack of Ingres, and to follow the upgrade steps carefully.

Typical Upgradedb Problems

Here are some typical problems you may encounter when using the upgradedb utility.

The upgradedb utility starts to process, and then hangs with no error indication.

This condition is probably caused by the Remote Command Server interfering with the upgradedb process, which is likely if you are upgrading to Ingres II 2.0 instead of Ingres 2006. Use the rmcmdstp command to stop the Remote Command Server.

You can use Configuration-By-Forms or Visual Configurator to turn off the Remote Command Server until the upgrade is finished: select Remote Command Server and use EditCount to set the startup count to zero.

The following message occurs: "Product *name* has been made uninstallable by an incompatible dictionary upgrade."

This message is caused by extra or incorrect rows in the front-end catalog ii_client_dep_mod. The rows may have been created by very old versions of Ingres. You can ignore this message.

The following message occurs shortly after the upgradedb utility starts processing a database: "E_SC0206 An internal error prevents further processing of this query."

This message is seen when **upgradedb –all** is used, and the database data ROOT location is not the same as others processed in the same upgradedb run. The errlog.log shows the message "E_DM9004_BAD_FILE_OPEN" referencing a filename: aaaaaaaa.cnf, shortly before the E_SC0206 message.

This message has occasionally been seen in various versions of upgradedb. Simply rerun upgradedb for the one failed database, and continue the upgrade.

Warning messages appear after upgradedb announces "Reloading query- tree objects."

Although the messages are warning messages and the database is shown as having been upgraded, some views, permits, or other objects have been recreated incorrectly and are missing from the database, so treat these warnings as if they are fatal errors.

Refer to the following files found in the directory $II_SYSTEM/ingres/files/upgradedb/UPGRADEUSER, where UPGRADEUSER is the user who ran upgradedb, for example "ingres":

- upgradedb.log

- upgradedb.dbg

- *dbname*.i*nn*

    SQL terminal monitor script, where *dbname* is the name of the database being upgraded and *nn* is the sequence number of the upgrade for that database starting from 1.

- *dbname*.o*nn*

    Output from *dbname*.i*nn,* where *dbname* is the name of the database being upgraded and *nn* is the sequence number of the upgrade for that database starting from 1.

You can identify the problem by inspecting the messages found in upgradedb.log. Data is appended to this file, so more than one entry may exist for *dbname*.

A possible cause of failure is that a table used in a database procedure has been dropped and when that procedure is being recreated as part of upgradedb, the program fails due to table not found.

You may be able to resolve the problem by inspecting the output file (*dbname*.o*nn*) and then editing the *dbname*.i*nn* file. Make copies of the files before making changes. After you have made the changes to the *dbname*.i*nn* file, rerun it using the Terminal Monitor.

For assistance, contact Ingres Technical Support.

Other Upgradedb Problems

If something else goes wrong with the upgradedb utility, contact Ingres Technical Support for help.

For problems with a single database, customer support can assist you in restoring the database data files from your system backup and resetting the database information in iidbdb so that you can retry upgradedb. In the worst case, it may be necessary to restore the entire installation from your system backup, fix the database problem, and redo the upgrade.

Chapter 6: Considerations for Alpha OpenVMS

This section contains the following topics:

This chapter describes the steps required for upgrading Ingres on the Alpha OpenVMS platform from Ingres II 2.0 to Ingres 2.6/0401 or Ingres 2006 (axm.vms/100).

Use this chapter together with the appropriate edition of the Ingres *Getting Started* guide or *Installation Guide* and the Readme file.

OpenVMS Requirements

For the minimum process requirements for an Ingres system administrator, see the appendix "System Requirements for OpenVMS" in the *Installation Guide*. Also see the Readme file.

Considerations When Installing Ingres on OpenVMS

For full instructions on installing Ingres on OpenVMS, see the *Installation Guide*. The installation process has not changed significantly from Ingres II 2.0.

Ingres uses the VMSINSTAL procedure to install and configure its software. Using VMS, it is possible to create the new Ingres system administrator account, extract the software required, and configure Ingres. However, depending on how the installation progresses, some issues may develop.

You can install Ingres either directly from the CD-ROM or from a working area on the target system. If the files are transferred to the target node through FTP, they must be moved across in binary mode.

Mount the CD

If the machine has a CD-ROM drive, you can use the following command to mount the CD:

```
$ MOUNT /OVERRIDE=IDENTIFICATION /MEDIA_FORMAT=CDROM -
    /UNDEFINED_FAT=(FIXED:CR:32256) <CD Device>
```

To access the readme use the following:

```
$ MOUNT /OVERRIDE=IDENTIFICATION /MEDIA_FORMAT=CDROM -
    /UNDEFINED_FAT=STREAM:32767 CD_Device
```

Run VMSINSTAL

To run the installer, issue the following command from any privileged account that is defined as holding the privileges needed to run Ingres:

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

By default, the SYS$ROOT area is used by VMSINSTAL to unpack the savesets in preparation for installing Ingres. If there is insufficient space available, then VMSINSTAL will fail. To specify an alternate working directory, you can use the awd parameter, as follows:

```
@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
```

Known Installation Issues

Note: For more information about these issues, check the technical documents available at the Ingres Technical Support web site.

Creating the Ingres System Administrator account from within VMSINSTAL does not assign the correct process quotas to the account. (For the correct quotas, see the *Installation 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 must enter the correct information when prompted.

If Ingres is installed from a non-Ingres System Administrator account, imadb is created as the process owner for VMSINSTAL rather than the installation owner configured earlier. When Ingres is started, the RMCMD process will hang because 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.

Schema Checking

Ingres reserves a number of new keywords, mostly for support of SQL additions. If names such as *substring*, *first*, or *cache* are used as column names, you must change the database schema. For a list of Ingres reserved words, see the appendix "Keywords" and the *SQL Reference Guide*.

If you are concerned that some column names in your database may conflict with reserved words, you can take a copy of the schema from the current installation and load it into an Ingres database. You should extend these checks 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 must be removed before moving to Ingres.

Application Rebuilding

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

The following compilers have been tested and are known to work with Ingres for OpenVMS.

HP Ada

HP BASIC

HP C

HP COBOL

HP C++

HP Pascal

HP Fortran

Note: For the latest information, see the Readme.

Building Member_Aligned Against Ingres 2.6 or 2006

Note: This section applies only to migrating from releases prior to Ingres II 2.0/0011 (axm.vms/00).

With the move to a member_aligned version of Ingres, some applications must be rebuilt. You must rebuild applications that connect directly to an installation located on the same node, or through Ingres Net on the client node.

Building Vision and Application-By-Forms applications member_aligned is the default behavior, with no further changes being required from the developer.

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

If an application cannot be built using Alpha member alignment, it is possible to rebuild it with the Ingres components naturally aligned. The steps needed for C and COBOL applications are described in the following sections.

These changes require modification to the Ingres supplied files only and not the application code. Even by performing the steps listed here, you still must recompile **all** parts of the application that interface with Ingres or that use any structures declared in the Ingres header files.

By default, user applications are built using the same compiler options used to build the Ingres libraries and applications. If these options are not used, proceed carefully.

The introduction of the member_aligned version of Ingres occurred 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, data corruption is likely to occur.

Modifications Required For C Applications

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

Modifications Required 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]UTCOM.DEF file requires the removal of the qualifier "/alignment=padding" from the COBOL compile statements.

Appendix A: Upgrading from Ingres 6.4

This section contains the following topics:

How To Use This Appendix

This appendix describes special considerations when upgrading from Ingres 6.4.

Regardless of the upgrade method you are using, you should perform the tasks described in Considerations for Ingres 6.4 (see page 89).

If you are using the unload/reload method to upgrade from Ingres 6.4, follow the instructions in How You Upgrade from Ingres 6.4 Using Unload/Reload (see page 96).

If you are using the upgradedb method to upgrade from Ingres 6.4, you can follow the procedure in the chapter "Upgrading Using Upgradedb." If you have difficulties with that procedure in your testing, you can use the process described in Alternate Upgradeb Procedure (see page 106).

This appendix also lists Ingres 6.4 parameters and their corresponding names in newer releases.

Considerations for Ingres 6.4

There are additional considerations when loading Ingres 6.4 databases and applications into the new development installation. These considerations include application preparation and system preparation.

Application Preparation

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

- Semantics change in the UPDATE…FROM statement

- Decimal constant semantics change

- Greater sensitivity to BYREF errors

- Journaling on by default

- Greater sensitivity to arithmetic errors

- 4GL TABLE_KEY type conversions

- User-defined data type changes

UPDATE . . . FROM Semantics Change

In Ingres 6.4/05 and earlier, the "ambiguous replace" test allowed an update using the UPDATE…FROM statement if each target row was being updated with an unambiguous value. Ingres 6.4/06 and higher releases test for multiple FROM rows and generate an ambiguous replace error message even if all the FROM rows generate the same replacement value.

For example, Ingres 6.4/05 and earlier allowed the following update:

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

even though there is no WHERE qualification joining the tables, since the replacement value was non-ambiguous. In later releases, an "ambiguous replace" error message displays.

The recommended approach for this semantics change 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 setting the DBMS parameter "ambig_replace_64compat" to ON in Configuration-By-Forms.

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.

Typically, this does not matter, as Ingres does appropriate type conversions. However, 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 recommended approach 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, as follows:

```
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 applications to set II_NUMERIC_LITERAL in their environment.

Greater Sensitivity to BYREF Errors

Ingres 6.4 4GL programs are 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 the wrong type. In some cases, this can result in programs aborting and segmentation violations. The cure is to ensure that the called and calling routines return values of compatible length and type.

An as interim fix, an environment variable can be set to cause the 4GL runtime system to pass parameters the way 6.4 did: all integers forced to 4-byte, all floats forced to 8-byte. Character string passing is not affected. The environment variable setting is:

```
setenv II_PARAM_PASSING FORCEMAX
```

Journaling On by Default

In Ingres 6.4, if a database was journaled, a 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.

You can turn default journaling off by changing the Configuration-By-Forms parameter "default_journaling." Alternative options are to issue a SET NOJOURNALING statement at the beginning of an application, create temporary tables WITH NOJOURNALING, or use session tables.

Greater Sensitivity to Arithmetic Errors

Ingres 6.4 ignores a number of arithmetic error conditions (such as floating point overflow and 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 must 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))
```

Some 6.4 releases of 4GL had problems with variables of type TABLE_KEY. If you were doing type conversions to avoid the use of TABLE_KEY variables, consider removing the conversion altogether and using the TABLE_KEY type directly.

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. For details, see the *Object Management Extension User* Guide.

Application Preparation Summary

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, you do not have 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 enhance performance, there is plenty of time to add performance enhancements *after* the upgrade.

System Preparation

Take the following steps to prepare your system:

- Change customized start and stop shell scripts to reflect new commands
- Change shell scripts that use ingprenv1 environment variable
- Carry forward any changes to archiver exit script
- Change transaction log parameter settings

Ingres Startup and Shutdown

Ingres uses new commands for startup and shutdown: ingstart and ingstop instead of iistartup and iishutdown. If you have customized shell scripts that start and stop Ingres, you must change them. Verify the changes in the development Ingres installation and have the revised scripts ready for the production environment at time of upgrade.

If you are running multiple DBMS servers with Ingres 6.4, you should be able to simplify your startup and shutdown procedures. Ingres supports multiple DBMS servers directly from the Ingres configuration.

ingprenv Replaces ingprenv1

In Ingres, the ingprenv command replaces the Ingres 6.4 ingprenv1 command, which displayed one Ingres environmental variable. Shell scripts that use ingprenv1 must 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
```

Archiver Exit Shellscript

Ingres has a sample Archiver exit script, acpexit.def. If the Ingres 6.4 acpexit script was customized, you must carry over these changes to the Ingres installation.

For information about the acpexit script, see the *System Administrator Guide*.

Transaction Log Size

Generally, Ingres uses less transaction log file space than Ingres 6.4. A few operations may use more (for example, MODIFY TO MERGE). To allow for its improved logging algorithms, Ingres reserves transaction log space that it may not actually write.

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 barely large enough, it may be advisable to increase the size before or during the upgrade.

Unload/Reload Procedure for Upgrading from 6.4

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 clean databases, but requires more time and disk space than does the upgradedb method.

Note: Databases using the system-maintained logical key feature are best upgraded using upgradedb. Tables that contain SYSTEM_MAINTAINED table_key or object_key columns cannot be safely unloaded and reloaded without additional work. The reload step generates all new logical key values. If other tables reference the logical key columns, the new values must be manually propagated to those tables.

Unload/Reload Upgrade Types

You must choose one of the following variations of the unload/reload procedure:

- In-place upgrade, which replaces the 6.4 installation with the new Ingres installation. The master database (iidbdb) is upgraded with upgradedb, even though other databases are unloaded and reloaded. Because the iidbdb remains, all your locations, users, groups, and roles still exist in the new installation.

- Clean install upgrade, which leaves the 6.4 installation alone. Ingres is installed into a completely new installation. (The new installation may even be on a different machine.) When performing a clean install upgrade, you must take extra steps to recreate locations and move users, groups, and roles from the 6.4 installation to the new one.

Front-end Catalogs and the Upgradefe Program

The hardest part of the unload/reload upgrade is dealing with the front-end catalogs. These catalogs 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.

How You Upgrade from Ingres 6.4 Using Unload/Reload

To upgrade from Ingres 6.4 using the unload/reload procedure, follow this process.

Note: 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 created in Step 1, and perform this step." If using Ingres Star, include the coordinator database in the list of databases. Steps that apply to a particular upgrade type only (that is, in-place upgrade or clean install upgrade) are marked accordingly.

1. [Each DB including iidbdb] Create Unload Directory (see page 57).

2. [Each DB] Run Unloaddb (see page 58).

3. [Each DB] Check for Obsolete Users (see page 97).

4. [Each DB including iidbdb] (Optional) Checkpoint the Database (see page 59).

5. Disable User Access (see page 44).

6. Shut Down Ingres and Back Up System (see page 46).

7. [Each DB] Unload the Database (see page 60).

8. [Each DB] (Optional) Print Optimizer Statistics (see page 61).

9. [Each DB] Record Database Information (see page 47).

10. Record Database Privileges (see page 98).

11. Save Users, Groups, and Roles (see page 99).

12. [Each DB] Destroy the Database (see page 63).

13. Clean iidbdb Database (see page 100).

14. Record Ingres Configuration (see page 100).

15. Shut Down Ingres (see page 101).

16. Disable Ingres Startup (see page 65).

17. Preserve Site Modifications (see page 48).

18. Fix Logins (see page 101).

19. Save Ingres Settings (see page 101).

20. Clean Up Ingres 6.4 (see page 102).

21. Create Work Location (see page 102).

22. Install Ingres (see page 66).

23. Create imadb Database (see page 51).

24. Restore Site Modifications (see page 69).

25. Configure Ingres (see page 103).

26. Set Up Ingres Net (see page 69).

27. Start Ingres (see page 52).

28. Recreate Users, Groups, and Roles (see page 104).

29. Recreate Locations (see page 71).

30. [Each DB] Recreate the Database (see page 71).

31. [Each DB] Extend the Database (see page 71).

32. Recreate Database Privileges (see page 105).

33. [Each DB] Fix FE Reload Script (see page 106).

34. [Each DB] Reload the Database (see page 73).

35. [Each DB] Upgrade Front-End Catalogs (see page 74).

36. [Each DB] Reapply Optimizer Statistics (see page 74).

37. [Each DB including iidbdb] Checkpoint the Database (see page 53).

38. Install Upgraded Applications (see page 53).

The sections that follow provide details on steps that differ from those described previously in this guide.

Check for Obsolete Users

Old databases may have objects created by users who no longer exist. Check for obsolete users for each database.

To check for obsolete users:

1. Examine the scripts created by unloaddb earlier in the upgrade procedure.

   Each script contains one line for each user who owns a database object.

2. Make sure that all users listed are valid.

3. If obsolete users are found, delete the relevant lines from the scripts.

4. Delete the cp{user}.in and cp{user}.out files.

5. Go into the database and clean out these unwanted objects.

Record Database Privileges

To record database privileges

1. As the installation owner, change directories to the unload directory for iidbdb created in Step 1 of this upgrade procedure.

2. Run the following SQL to save private database access lists and user database privileges:

```
sql iidbdb
\script dbaccess.out
select dbname, usrname
from iidbaccess
order by dbname, usrname
\go
\script
\script dbprivs.out
select *
from iidbprivileges
where database_name <> ''
order by database_name,grantee_name
\go
\script
\quit
```

This procedure creates two files, dbaccess.out and dbprivs.out.

Save Users, Groups, and Roles

Note: This step is required only for a clean-install upgrade.

To save users, groups,and roles

1. As the installation owner, change directory to the iidbdb unload directory created in Step 1 of the upgrade procedure.

2. Run the following SQL to save users, groups, and roles:

```
sql iidbdb
create table unload_tmp as
select name,status,default_group
from iiuser
where name not in ('ingres','$ingres','root')
\go
copy unload_tmp (
    name=c0comma,status=c0comma,default_group=c0nl
) into 'users.out'
\go
drop unload_tmp;commit
\go

 copy iiusergroup (
  groupid=c0comma,groupmem=c0nl
) into 'groups.out'
\go

 copy iirole(
  roleid=c0nl
) into 'roles.out'
\go
 \quit
```

Clean iidbdb Database

Note: This step is required only for an in-place upgrade.

To clean the iidbdb database, become the installation owner and run the following steps against the master database iidbdb.

Note: 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

ckpdb -s -j  iidbdb
```

The verifydb command may issue the following messages, which you can ignore:

```
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.
```

You can also ignore the "patch warning" message that warns of the loss of user tables in "runinteractive" mode. This mode will not be used.

Record Ingres Configuration

To record Ingres configuration:

1. As the installation owner, execute the "showrcp" command.
2. Record the contents of the rundbms.opt file found in $II_SYSTEM/ingres/files.

You will use this information later as a guide for configuring Ingres. The Ingres installation procedure does not preserve the Version 6.4 parameter settings. During installation, the ingres/files directory is deleted, so save the information.

Shut Down Ingres

Shut down Ingres with the iishutdown command.

Fix Logins

To fix logins:

1. If necessary, make sure that the login for the installation owner sets LD_LIBRARY_PATH or the platform equivalent.

2. Make sure that the login for the user does not use ingprenv1, or install your ingprenv1 substitute, as described in ingprenv Replaces ingprenv1 (see page 93).

3. Check all your database owner (DBA) logins to ensure that they are properly set up for Ingres, with LD_LIBRARY_PATH or equivalent, and no use of ingprenv1.

4. Define LD_LIBRARY_PATH or equivalent for the installation owner user session that you will use to install and upgrade Ingres.

5. If you are doing a clean-install upgrade on a different machine, make sure that your login fixes are applied to the new machine, not to the old one.

Save Ingres Settings

Note: This step is required only for an in-place upgrade.

The upgrade runs more smoothly if the Ingres 6.4 executables, control files, and environment variables are deleted. However, you do not want to lose your installation ID and default locations. These are kept in a file named symbol.tbl.

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

Clean Up Ingres 6.4

Note: This step is required only for an in-place upgrade.

To guarantee a clean environment for Ingres:

1. Invoke the following commands:

   ```
   cd $II_SYSTEM/ingres

   rm -rf bin files lib utility dbtmplt version.rel admin

   mkdir files
   ```

2. Copy your saved symbol.tbl back into the $II_SYSTEM/ingres/files directory**.**

Create Work Location

Note: This step is required only for an in-place upgrade.

The Ingres installation procedure asks for a location for temporary files and sorting, and creates the directories if they do not exist. However, you should create this location manually because some versions of the installation procedure may not properly set the protections for the directories, which can cause upgradedb to fail when upgrading the iidbdb database.

For information on placement of your default work location, see the *Database Administrator Guide*.

As the installation owner, 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
```

Restore Site Modifications

Restore any site-specific files that you copied in the step Preserve 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 with Ingres, as the file format has changed. This means that you must recreate the changes using the Ingres 6.4 cktmpl.def as a guide. See the Ingres 6.4 *Database Administrator's Guide*.

If the archiver exit script acpexit was changed in Ingres 6.4, you must make the changes to the Ingres template (acpexit.def), and then move that file to $II_SYSTEM/ingres/files/acpexit.

Configure Ingres

Run Configuration-By-Forms (CBF) and initially configure 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 Guide*.

Information on the correlation between 6.4 and Ingres parameter names, is described in Corresponding Parameter Names (see page 114).

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

Ingres versions from 2.0 through 2.6 may calculate very large default lock and resource limits parameters. Check the lock_limit and resource_limit settings, and 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 may not need as much qsf_memory as did Ingres 6.4. OS-thread platforms should not reduce quantum_size, as it does not improve performance on those platforms.

Recreate Users, Groups, and Roles

Note: This step is required only for a clean-installation upgrade.

If your 6.4 installation has only a few Ingres users defined, you should use the accessdb utility or the CREATE USER SQL statement to recreate those users in the Ingres installation. As a guide, use the file users.out or refer to the 6.4 installation.

If you have many users, the following procedure recreates them in mass.

As the installation owner, change directory to your iidbdb unloaddb directory where you stored the files from the step Save Users, Groups, and Roles.

Run this SQL:

```
sql '-u$ingres' iidbdb
copy iiuser(name=c0comma,status=c0comma,default_group=c0nl)
from 'users.out'
\go
update iiuser
set default_priv = status, user_priv = status,
  flags_mask = case when default_group <> ' '  then 28 else 24 end
where user_priv = 0 and flags_mask = 0;
\go
copy iiusergroup(groupid=c0comma,groupmem=c0nl)
from 'groups.out'
\go
commit
\go
\quit
```

Windows: Omit the quotes from the sql command line.

Ingres has new user privileges that do not exist in 6.4. If you recreate users using the above bulk load procedure, you should review the added users with accessdb to make sure that all user privileges are set the way you want them. In particular, review the definitions for any 6.4 "superusers."

Ingres handles the "update system catalog" privilege differently than did 6.4. You must explicitly grant this privilege to the Ingres user after you recreate it, with a grant statement, as follows:

```
grant update_syscat on current installation to user-name
```

If your 6.4 installation had roles defined, recreate them with the ADD ROLE SQL statement. Use the file roles.out as a guide. Roles cannot be reliably bulk-loaded from the 6.4 installation, so you must recreate them by hand. After you recreate each role, issue the following SQL statement:

```
grant rolename to public; commit
```

This allows the role to be used in the same manner as in 6.4.

Recreate Database Privileges

To recreate database privileges:

1. As the installation owner, change to the iidbdb unloaddb directory.

2. Refer to the file dbaccess.out created in the step Record Database Privileges.

3. Start an iidbdb Terminal Monitor session:

   ```
   sql iidbdb
   ```

4. For each database and user combination listed in dbaccess.out, issue the statement:

   ```
   grant access on database database-name to username; commit
   ```

5. Review the file dbprivs.out created in the step Record Database Privilege.

   Each row describes one or more database privileges given to the user *grantee-name*. A Y or N in a privilege column indicates the specific privilege. (A U in a column means Unchanged.)

6. For each row, issue the statement

   ```
   grant privilege on database database-name to grantee-name;commit
   ```

   If the privilege column is N, grant *noprivilege* instead of *privilege*.

7. When finished, use **\quit** to exit the iidbdb session.

If you have defined many privileges, or recreated many users, groups, or roles, you should run sysmod on the iidbdb, which will accelerate query processing. Issue the sysmod command, as follows:

```
sysmod iidbdb
```

Fix FE Reload Script

Because the new database was not created with front-end catalogs, it is not necessary to drop them.

To fix the front-end reload script, for each database:

1. Open the file cp_ingres.in.

2. Delete the following lines:

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

   Note: The directory path may differ.

3. Save the file.

Alternate Upgradeb Procedure

The upgradedb utility in Ingres 2006 is enhanced to allow 6.4 databases to be upgraded using the standard procedure in the chapter "Upgrading Using Upgradedb." Due to the many enhancements made since Ingres 6.4, the upgradedb utility performs an intricate task when upgrading a 6.4 database.

If you encounter upgradedb problems in your testing, or if you prefer a safer (but more complex) procedure, use the alternate upgradedb procedure in this section. This modified procedure is designed so that the upgradedb utility has to perform as little work as possible, so that it will correctly handle the upgrade tasks.

In the procedure, 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 directs you to cut and paste the output of unloaddb to generate SQL that recreates database objects and storage structures. If procedures already exist to recreate database objects and storage structures, you can use these instead. Make sure, however, that the procedures recreate all the relevant objects. If users or applications dynamically create database objects, it may be safer to cut and paste from unloaddb.

The alternate 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 the installation owner (default user ID is ingres), and use the -u{user} flag to pretend to be that user whenever you must run an Ingres command.

How You Upgrade from Ingres 6.4 Using Upgradedb (Alternate)

To upgrade from Ingres 6.4 using the alternate upgradedb procedure, follow this process.

Note: 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 created in Step 1, and perform this step." Do not include the iidbdb or Ingres Star databases unless instructed. If using Ingres Star, remember to include the coordinator database in the list of databases.

1. [Each DB including Ingres Star DDBs] Create Unload Directory (see page 108).

2. [Each DB including Ingres Star DDBs] Run Unloaddb (see page 109).

    Note: You can omit Steps 2 through 4 if procedures already exist to recreate all database objects and storage structures. However, it will be necessary to make the appropriate changes to the oi_prep.sh script-see the step Remove Non-table Objects-for re-modifying all tables.

3. [Each DB including Ingres Star DDBs] Check for Obsolete Users (see page 97).

4. [Each DB] Edit the Unloaddb Output (see page 110).

5. [Each DB including iidbdb] (Optional) Checkpoint the Database (see page 59).

6. Disable User Access (see page 44).

7. Shut Down Ingres and Back Up System (see page 46).

8. [Each DB] (Optional) Print Optimizer Statistics (see page 61).

9. [Each DB] Remove Non-table Objects (see page 111).

10. [Each DB] Record Database Information (see page 47).

11. Clean iidbdb Database (see page 100).

12. [Each DB including iidbdb] Checkpoint and Turn Off Journaling (see page 112).

13. Record Ingres Configuration (see page 100).

14. Shut Down Ingres (see page 101).

15. Disable Ingres Startup (see page 65).

16. Preserve Site Modifications (see page 48)

17. Fix Logins (see page 101).

18. Save Ingres Settings (see page 113).

19. Clean Up Ingres 6.4 (see page 102).

20. Create Work Location (see page 102).

21. Install Ingres (see page 66).

22. Create imadb Database (see page 51).

23. Restore Site Modifications (see page 69).

24. Start Ingres (see page 52).

25. Run Upgradedb Utility (see page 52).

26. Configure Ingres (see page 103).

27. Set Up Ingres Net (see page 69).

28. [Each DB] Recreate Objects (see page 113).

29. [Each DB] Reapply Storage Structures (see page 113).

30. [Each DB] Reapply Optimizer Statistics (see page 74).

31. [Each DB including iidbdb] Checkpoint the Database (see page 53).

32. Install Upgraded Applications (see page 53).

The sections that follow provide details on steps that differ from those described previously in this guide.

Create Unload Directory

You must create a directory to hold scripts, but no data. Make the directory writable by anyone. The disk space needed is a maximum of 1 MB per directory.

To create a directory, issue the following commands for each database, including the Ingres Star databases:

UNIX:

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

Windows:

```
mkdir d:\someplace\dbname
```

Run Unloaddb

Note on Steps 2 through 4: You can omit Steps 2 through 4 if procedures already exist to recreate all database objects and storage structures. However, it will be necessary to make the appropriate changes to the oi_prep.sh script-see the step Remove Non-table Objects-for re-modifying all tables.

Run unloaddb against each database. The unloaddb command does not unload the database; it simply creates scripts. You can edit these scripts to produce a collection of scripts that recreate various database objects and storage structures.

For Ingres Star databases, unload the CDB in the same way as for a local database. For a DDB, use unloaddb/star.

For a regular DB or CDB, issue this command:

unloaddb *dbname*

For an Ingres Star DDB, issue this command:

unloaddb *ddbname*/star

Edit the Unloaddb Output

The unloaddb output must be modified for recreating just the database objects and storage structures.

To edit the unloaddb output, manually edit each cp{user}.in file that unloaddb created to extract the following statements:

- Create rule statements into a file named {user}_rule.sql

- Create procedure related statements into {user}_dbp.sql

- Create dbevent related statements into {user}_event.sql

- Modify statements into {user}_modify.sql

- Modify and create index statements into {user}_modindex.sql

- All other non-base-table related statements into {user}_grantview.sql. This file will contain grants, QUEL permits, QUEL integrities, and view definitions.

For UNIX, the extract_unloaddb.sh shellscript is available that extracts one user's object definitions. The script is available on the Ingres Technical Support web site.

Note: The $ingres user should not own any non-catalog objects, so do not process the cp_ingre.in file that unloaddb creates.

As a result of this step, SQL scripts are created that can recreate any database object or storage structure owned by any user in any database.

Remove Non-table Objects

The purpose of removing non-table objects is to reduce the database to base tables.

Some database objects such as procedures and views can be very complicated, and some past versions of upgradedb did not always process them successfully. Additionally, processing of some objects (grants in particular) is slow and expensive. Dropping the grants and later recreating them avoids any possible failure due to lack of transaction log space.

Note: Do not process Ingres Star distributed databases.

To remove non-table objects

1. Drop all non-table objects from the database including:

   - Optimizer statistics

   - Views

   - Rules

   - Database procedures

   - Database events

   - Secondary indexes

   - Grants and QUEL permits

   - QUEL integrities

2. Modify all tables to heap.

UNIX:

To perform this step automatically

1. Use the shell script oi_prep.sh. The script is available from Ingres Technical Support.

   Using the C shell, issue this command:

   ```
   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 drop views in reverse dependency order); ignore the "drop" errors

2. Run verifydb checks against the database.

   The verifydb -odbms command may output the following messages, which you can ignore:

   ```
   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.

Also ignore the "patch warning" message that warns of the loss of user tables in "runinteractive" mode. This mode will not be used.

3. If some databases produce a "verifydb failed" message and then abort, run the Terminal Monitor with the update system catalogs flag, as follows:

   `sql +U ` *dbname*

   `SELECT * FROM iistatistics;\go`

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

4. If there are rows, delete them, as follows:

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

5. Rerun the verifydb command as shown at the end of the oi_prep.sh.

6. If error messages are returned from verifydb, correct the problems before continuing. Contact Ingres Technical Support for help, if necessary.

Checkpoint and Turn Off Journaling

For each database, including the iidbdb, checkpoint each database and turn off journaling. Then save the configuration file.

The upgradedb process turns off journaling, so it is best to turn if off now. If upgradedb fails, you can use this checkpoint to recover and try again.

To checkpoint and turn off journaling

1. Checkpoint each database, using the ckpdb command with -j option to turn off journaling. Issue the following command:

   `ckpdb -d -j ` *dbname*

   Note: For the iidbdb, use the ckpdb -s option. The iidbdb database does not have an "unload" directory.

2. Save the configuration file stored in the dump area after each checkpoint. The configuration file is small. Issue the following command:

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

Save Ingres Settings

Save your installation ID and default locations, which are kept in a file named symbol.tbl. Copy $II_SYSTEM/ingres/files/symbol.tbl to a safe area not in the Ingres directory tree.

Recreate Objects

Using the scripts generated by the step Edit the Unloaddb Output, recreate the views and other database objects.

Recreate objects in the following sequence:

1. Views, QUEL integrities, and grants:

   ```
   sql -uuser dbname <user_grantview.sql
   ```

2. Dbevents:

   ```
   sql -uuser dbname <user_event.sql
   ```

3. Database procedures:

   ```
   sql -uuser dbname <user_dbsp.sql
   ```

4. Rules:

   ```
   sql -uuser dbname <user_rule.sql
   ```

Remember to run all four scripts for each user who owns objects in each database.

If your application system has its own scripts to recreate database objects, you may use them instead of the unloaddb-generated scripts.

Reapply Storage Structures

For each *user*_modindex.sql script generated by the step Edit the Unloaddb Output, reapply storage structures and indexes:

```
sql -uuser dbname <user_modindex.sql
```

If your application system has its own scripts to reapply storage structures and create indexes, you may use them instead of the unloaddb-generated scripts.

Corresponding Parameter Names

The configuration system in Ingres 6.4 differs from that of subsequent releases, so you need to know how the Ingres 6.4 server parameters correspond to the new Ingres parameters.

All corresponding Ingres parameters listed here are found in the DBMS Server component. Ingres parameters that do not have any corresponding Ingres 6.4 parameters are not listed.

Parameters in 6.4 rundbms.opt File

The following table lists Ingres 6.4 parameters and their corresponding parameter names in the new Ingres.

Note: Parameters of type Cache are repeated for each cache page size.

Ingres 6.4 Parameter	Ingres Parameter	Type
active_sessions	active_limit	Derived
cache_name	cache_name	
connected_sessions	connect_limit	
cpu_statistics	cpu_statistics	Derived
cursors_per_session	cursor_limit	
database_count	database_limit	Derived
dblist	database_list	Databases
define	define_address	Derived
dmf.cache_size	dmf_cache_size	Cache, Derived
dmf.count_read_ahead	dmf_group_count	Cache, Derived
dmf.dbcache_size	dmf_db_cache_size	
dmf.flimit	dmf_free_limit	Cache, Derived
dmf.memory	dmf_memory	Cache, Derived
dmf.mlimit	dmf_modify_limit	Cache, Derived
dmf.scanfactor	dmf_scan_factor	Cache
dmf.size_read_ahead	dmf_group_size	Cache
dmf.tblcache_size	dmf_tbl_cache_size	
dmf.tcb_hash	dmf_hash_size	

Ingres 6.4 Parameter	Ingres Parameter	Type
dmf.wbend	dmf_wb_end	Cache, Derived
dmf.wbstart	dmf_wb_start	Cache, Derived
events	event_limit	
fast_commit	fast_commit	Derived
flatten	query_flattening (ON)	
image	image_name	
maximum_working_set	unix_maximum_working_set	
names	name_service (ON)	
noflatten	query_flattening (OFF)	
nonames	name_service (OFF)	
opf.active	opf_active_limit	Derived
opf.aggregate_flatten	qflatten_aggregate (ON)	Derived
opf.complete	opf_complete (ON)	
opf.cpufactor	opf_cpu_factor	
opf.exactkey	opf_exact_key	
opf.memory	opf_memory	Derived
opf.noaggregate_flatten	qflatten_aggregate (ON)	Derived
opf.nocomplete	opf_complete (OFF)	
opf.nonkey	opf_non_key	
opf.rangekey	opf_range_key	
opf.repeatfactor	opf_repeat_factor	
opf.sortmax	opf_sort_max	
opf.timeoutfactor	opf_timeout_factor	
priority	unix_priority	
psf.memory	psf_memory	Derived
qef.qep_size	qef_qep_mem	
qef.sort_size	qef_sort_mem	
qsf.pool_size	qsf_memory	Derived
quantum	quantum_size	
rdf.max_tbls	rdf_max_tbls	

Ingres 6.4 Parameter	Ingres Parameter	Type
rdf.memory	rdf_memory	Derived
rdf.tbl_cols	rdf_tbl_cols	
rdf.tbl_idxs	rdf_tbl_idxs	
rule_depth	rule_depth	
scf.row_estimate	scf_rows	
server_class	server_class	
session_accounting	session_accounting	
shared_cache	cache_sharing (ON)	
sole_cache	cache_sharing (OFF)	
sole_server	sole_server	Derived
stack_size	stack_size	
write_behind	dmf_write_behind (see notes)	Cache

Notes on Specific DBMS Server Parameters

Note the following:

- The 6.4 QEF sorting algorithm is unsuited to large qef_sort_mem settings. All recent Ingres versions use a different sort that does not degrade with large qef_sort_mem settings. The 6.4 standard setting is much smaller than the Ingres default.

- Recent Ingres versions typically require significantly less qsf_memory than Ingres 6.4 does, perhaps as little as half. After upgrading, start with the same qsf_memory setting as Ingres 6.4, but monitor QSF memory usage with trace point QS501 and tune qsf_memory appropriately.

- The quantum_size parameter in an internal threads (slaves) installation is often set to a small number (50 to 100) to improve responsiveness. Quantum_size has a different meaning in an OS threads installation, where it should not be set to less than 300, or excessive polling of the session communications channel will occur. A quantum_size of 1000 is usually appropriate when OS threads are in use.

- The 6.4 write_behind parameter is a thread count. Starting with 2.5, the dmf_write_behind parameter is simply ON or OFF, and the server dynamically allocates threads. Prior to Ingres II 2.5, the write_behind parameter means the same as it did in 6.4.

- A stack_size of 64 KB is typical with 6.4. Recent Ingres versions use more stack, so the stack size should be set to 128 KB (or more, if sporadic session failures occur).

- 6.4 VMS installations can have a few additional non-UNIX parameters, which have the same or almost the same names in recent Ingres versions.

Locking and Logging System Parameters

These parameters are set with iistartup -init, or rcpconfig, in Ingres 6.4.

Ingres 6.4 Parameter	Ingres Parameter	Type
Log buffers in memory	buffer_count	Log
Transactions in the logging system	tx_limit	Log, Derived
Databases in the logging system	database_limit	Log, Derived
Maximum C.P. interval for invoking the archiver	archiver_interval	Log, Derived
Block size of the log file	block_size	Log
Log-full limit	full_limit	Log
Percentage of log for consistency point	cp_interval	Log, Derived
Force-abort limit	force_abort_limit	Log, Derived
Size of the lock hash table	hash_size	Lock, Derived
Size of the resource hash table	resource_hash	Lock, Derived
Maximum number of locks in the locking system	lock_limit	Lock, Derived
Maximum number of lock lists	list_limit	Lock, Derived
Maximum number of locks per transaction	per_tx_limit	Lock

Notes on Specific Logging and Locking Parameters

Note the following:

- There is no 6.4 equivalent to the Ingres log_writer parameter. Although the log_writer default is 1, it is usually advantageous to start your Ingres installation with log_writer set to 4 or 5. If you are using dual logging, double the setting.

- The default rule for computing lock_limit (and the new parameter resource_limit) tend to compute very high numbers-hundreds of thousands, or more. You can allow more locks than you did in 6.4. As an initial setting, a doubling of lock_limit is usually more than sufficient.

- Ingres 2006 requires at least 35 log buffers. The install or upgrade will raise the log buffer count to a minimum of 35.

- Configurator may compute a different archiver_interval than you used in 6.4. Carry over the 6.4 setting.

Appendix B: Keywords

This section contains the following topics:

Keywords in This Appendix

This appendix lists Ingres keywords and the contexts in which they are reserved. You can use the lists in this appendix to avoid assigning object names that conflict with reserved words.

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

Table Key

In the tables in this appendix, the column headings have the following meanings:

Non 6.4

Keywords not included in Ingres 6.4 keyword reserved lists

ISQL (Interactive SQL)

Keywords reserved by the DBMS

ESQL (Embedded SQL)

Keywords reserved by the SQL preprocessors

IQUEL (Interactive QUEL)

Keywords reserved by the DBMS

EQUEL (Embedded QUEL)

Keywords reserved by the QUEL preprocessors

4GL

Keywords reserved in the context of SQL or QUEL in Ingres 4GL routines

Reserved Single Keywords

The following single keywords are reserved.

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

| Keyword | Non 6.4 | SQL | | | QUEL | | |
		ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
abort		*	*	*	*	*	*
activate			*			*	
add		*	*	*			
addform			*			*	
after	*			*			*
all		*	*		*	*	
alter		*		*			
and		*	*		*	*	
any		*	*	*	*	*	
append					*	*	*
array	*			*			
as		*	*		*	*	*
asc		*		*			
asymmetric	*	*	*				
at		*	*	*	*	*	*
authorization		*	*				
avg		*	*	*	*	*	
avgu			*		*	*	
before				*			*
begin		*	*	*	*		*
between		*	*	*			
breakdisplay			*			*	
by		*	*	*	*	*	*
byref	*	*		*			*
call			*	*		*	*

Keyword	Non 6.4	SQL			QUEL		
		ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
callframe	*			*			*
callproc	*	*		*			*
cascade	*	*	*				
case	*	*	*	*			
cast	*	*					
check		*	*	*			
clear			*	*		*	*
clearrow			*	*		*	*
close		*	*		*		
coalesce	*	*					
collate	*	*	*				
column		*	*	*		*	
command			*			*	
comment	*			*			
commit		*	*	*			
committed	*	*	*	*			
connect			*				
constraint	*	*	*	*			
continue		*	*				
copy		*	*	*	*	*	*
copy_from	*	*					
copy_into	*	*					
count		*	*	*	*	*	
countu			*		*	*	
create		*	*	*	*	*	*
current		*	*				
current_user	*	*	*				
currval	*	*	*	*			
cursor		*	*				

| Keyword | Non 6.4 | SQL | | | QUEL | | |
		ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
cycle	*	*	*	*			
datahandler	*		*				
dbms_password	*		*	*			
declare		*	*	*			*
default	*	*	*	*			*
define	*	*			*		*
delete	*	*	*	*	*	*	*
deleterow			*	*		*	*
desc				*			
describe	*	*	*				
descriptor			*				
destroy					*	*	*
direct	*			*			*
disconnect			*				
display			*	*		*	*
distinct		*	*	*			
distribute	*				*		
do		*		*			*
down			*			*	
drop		*	*	*			
else		*		*			*
elseif		*		*			*
enable	*			*			
end		*	*	*	*	*	*
end-exec	*		*				
enddata			*			*	
enddisplay			*			*	
endfor		*	*	*			
endforms			*			*	

Keyword	Non 6.4	SQL			QUEL		
		ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
endif		*		*			*
endloop		*	*	*		*	*
endrepeat		*	*	*			
endretrieve						*	
endselect			*				
endwhile		*		*			*
escape		*	*				
except	*	*					
exclude	*				*		
excluding	*	*	*		*		
execute		*	*	*	*		
exists		*	*	*			
exit				*		*	*
fetch		*	*				
field			*			*	
finalize			*			*	
first		*	*	*			
for		*	*	*	*	*	
foreign	*	*	*				
formdata			*			*	
forminit			*			*	
forms			*			*	
from		*	*	*	*	*	*
full	*	*	*	*			
get	*			*			
getform			*			*	
getoper			*			*	
getrow			*			*	
global	*	*	*	*			

Keyword	Non 6.4	ISQL	SQL ESQL	4GL	IQUEL	QUEL EQUEL	4GL
goto			*				
grant		*	*	*			
granted	*		*	*			
group		*	*	*			
having		*	*	*			
help			*		*	*	
help_forms	*			*			*
help_frs			*			*	
helpfile			*	*		*	*
identified			*	*			
if		*		*			*
iimessage	*		*			*	
iiprintf	*		*			*	
iiprompt	*		*			*	
iistatement	*					*	
immediate		*	*	*			*
import	*	*					
in		*	*	*	*	*	
include			*		*		
increment	*	*	*	*			
index		*	*	*	*	*	*
indicator			*				
ingres						*	
initial_user	*	*	*				
initialize			*	*		*	*
inittable			*	*		*	*
inquire_equel						*	
inquire_forms	*			*			*
inquire_frs			*			*	

Keyword	Non 6.4	SQL			QUEL		
		ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
inquire_ingres	*		*	*		*	*
inquire_sql			*	*			
insert		*	*	*			
insertrow			*	*		*	*
integrity		*	*		*		*
intersect	*	*					
into		*	*	*	*	*	*
is		*	*	*	*	*	*
isolation	*	*	*	*			
join	*	*	*				
key	*		*	*			*
leave		*	*	*			
left	*		*	*			
level	*	*	*		*	*	
like		*	*				
loadtable			*	*		*	*
local	*	*					
max		*	*	*	*	*	
maxvalue	*	*	*	*			
menuitem			*			*	
message		*	*	*		*	*
min		*	*	*	*	*	
minvalue	*	*	*	*			
mode	*			*			*
modify		*	*	*	*	*	*
module	*	*					
move	*				*		
natural	*	*	*				
next		*	*			*	

Keyword	Non 6.4	SQL			QUEL		
		ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
nextval	*	*	*	*			
nocache	*	*	*	*			
nocycle	*	*	*	*			
noecho	*			*			*
nomaxvalue	*	*	*	*			
nominvalue	*	*	*	*			
noorder	*	*	*	*			
not		*	*		*	*	
notrim			*			*	
null		*	*	*		*	*
nullif	*	*					
of		*	*	*	*	*	*
offset	*	*	*	*			
on		*	*		*	*	*
only	*	*	*	*	*		*
open		*	*		*		
option		*					
or		*	*		*	*	
order		*	*	*	*	*	*
out			*			*	
outer	*	*	*	*			
param						*	
partition	*		*				
permit		*	*		*		*
prepare		*	*				
preserve	*	*	*				
primary	*		*	*			
print			*		*	*	
printscreen			*	*		*	*

Keyword	Non 6.4	SQL			QUEL		
		ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
privileges		*					
procedure		*	*	*			*
prompt			*	*		*	*
public		*	*				
purgetable	*		*	*		*	*
putform			*			*	
putoper			*			*	
putrow			*			*	
qualification	*			*			*
raise	*	*		*			
range					*	*	*
rawpct	*	*	*	*			
read	*	*	*		*		
redisplay			*	*		*	*
references	*	*	*	*			
referencing		*		*			
register		*	*	*	*	*	*
relocate		*	*	*	*	*	*
remove		*	*	*		*	*
rename	*				*		
repeat		*	*	*		*	*
repeatable	*	*	*				
repeated			*	*			
replace					*	*	*
replicate	*				*		
restart	*	*	*	*			
restrict	*	*	*				
result	*		*				
resume			*	*		*	*

Keyword	Non 6.4	SQL			QUEL		
		ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
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			*	*			
sleep			*	*		*	*

Keyword	Non 6.4	SQL			QUEL		
		ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
some		*	*	*			
sort					*	*	*
sql		*					
start	*	*	*	*			
stop			*				
submenu			*			*	
substring		*	*				
sum		*	*	*	*	*	
sumu			*		*	*	
symmetric	*	*	*				
system	*			*			*
system_ maintained	*	*	*		*	*	
system_user	*	*	*				
table		*	*				
tabledata			*			*	
temporary	*	*	*				
then		*	*	*			*
to		*	*		*	*	*
type	*			*			
uncommitted	*	*	*	*			
union		*	*	*			
unique		*	*	*	*	*	*
unloadtable			*	*		*	*
until		*	*	*	*	*	*
up			*			*	
update		*	*	*	*		
user		*	*	*			
using		*	*				
validate			*	*		*	*

| Keyword | Non 6.4 | SQL | | | QUEL | | |
		ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
validrow			*	*		*	*
values		*	*	*			
view		*	*		*		*
when	*	*	*				
whenever			*				
where		*	*	*	*	*	*
while		*					*
with		*	*	*	*	*	*
work		*		*			
write	*	*	*	*			

Reserved Double Keywords

The following double keywords are reserved.

| Keyword | Non 6.4 | SQL | | | QUEL | | |
		ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
add privileges	*			*			
after field	*			*			*
alter default	*		*	*			
alter group		*	*	*			
alter location	*	*	*	*			
alter profile	*	*					
alter role		*	*	*			
alter security_audit	*	*	*	*			
alter_sequence	*	*	*	*			
alter table	*		*	*			

Keyword	Non 6.4	SQL			QUEL		
		ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
alter user	*	*	*	*			
array of	*			*			
base table structure		*					
before field	*			*			*
begin declare	*		*				
begin exclude	*		*				
begin transaction		*	*	*	*	*	*
by group	*			*			
by role	*	*		*			
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	*	*	*	*			

| Keyword | Non 6.4 | SQL | | | QUEL | | |
		ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
create permit	*	*		*			
create procedure	*			*			
create profile	*	*	*	*			
create role		*	*	*			
create rule		*	*	*			
create security_alarm	*	*	*	*			
create sequence	*	*	*	*			
create synonym	*	*	*	*			
create user	*	*	*	*			
create view	*	*		*			
cross join	*	*	*	*			
curr value	*	*					
current installation	*			*			
current value	*	*	*	*			
define cursor					*		
declare cursor						*	
define integrity					*	*	*
define link						*	
define location					*		
define permit					*	*	*
define qry		*			*		*
define query		*			*		
define view					*	*	*
delete cursor					*	*	

Keyword	Non 6.4	SQL			QUEL		
		ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
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	*			*			
drop profile	*	*	*	*			
drop role		*	*	*			
drop rule		*	*	*			

Keyword	Non 6.4	SQL			QUEL		
		ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
drop security_alarm	*	*	*	*			
drop sequence	*	*	*	*			
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	*	*		*			

Keyword	Non 6.4	SQL			QUEL		
		ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
full outer	*	*		*			
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	*			*			
next value	*	*	*	*			
no cache	*	*	*	*			
no cycle	*	*	*	*			
no maxvalue	*	*	*	*			
no minvalue	*	*	*	*			

| Keyword | Non 6.4 | SQL | | | QUEL | | |
		ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
no order	*	*		*			
not like	*	*		*			*
not null	*				*		
on commit	*	*	*	*			
on current	*	*					
on database		*		*			
on dbevent		*		*			*
on location	*	*		*			
on procedure	*	*					
on sequence	*	*					
only where					*		
open cursor			*		*	*	
order by					*		
primary key	*	*		*			
procedure returning	*			*			*
put data	*		*				
raise dbevent		*	*	*			
raise error		*					
read only	*		*				
read write	*		*				
register dbevent		*	*	*			
register table	*						*
register view	*			*			*
remote system_password	*		*				

Keyword	Non 6.4	SQL			QUEL		
		ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
remote system_user	*		*				
remove dbevent		*	*	*			
remove table	*						*
remove view	*			*			*
replace cursor			*		*	*	*
result row	*	*	*	*			
resume entry	*			*			*
resume menu	*			*			*
resume next	*			*			*
resume nextfield	*			*			*
resume previousfield	*			*			*
retrieve cursor			*		*	*	
right join	*	*		*			
right outer	*	*		*			
run submenu	*			*			*
send userevent	*			*			
session group	*			*			
session role	*			*			
session user	*			*			
set aggregate	*	*			*		
set attribute	*		*				
set autocommit	*	*			*		
set connection	*		*	*			*
set cpufactor	*	*			*		

Keyword	Non 6.4	SQL			QUEL		
		ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
set date_format	*	*			*		
set ddl_concurrency	*	*					
set decimal	*	*			*		
set flatten	*	*			*		
set global	*		*				
set hash	*	*			*		
set io_trace	*	*			*		
set jcpufactor	*				*		
set joinop	*	*			*		
set journaling	*	*			*		
set lock_trace	*	*			*		
set lockmode	*	*			*		
set log_trace	*	*			*		
set logdbevents	*	*					
set logging	*	*			*		
set maxconnect	*	*			*		
set maxcost	*	*			*		
set maxcpu	*	*			*		
set maxidle		*			*		
set maxio		*			*		
set maxpage	*	*			*		
set maxquery	*	*			*		
set maxrow		*			*		
set money_format	*	*			*		

Keyword	Non 6.4	SQL			QUEL		
		ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
set money_prec	*	*			*		
set noflatten	*	*			*		
set nohash	*	*					
set noio_trace	*	*			*		
set nojoinop	*	*			*		
set nojournaling	*	*			*		
set nolock_trace	*	*			*		
set nolog_trace	*	*			*		
set nologdbevents	*	*					
set nologging	*	*			*		
set nomaxconnect	*	*			*		
set nomaxcost	*	*			*		
set nomaxcpu	*	*			*		
set nomaxidle	*	*			*		
set nomaxio	*	*			*		
set nomaxpage	*	*			*		
set nomaxquery	*	*			*		
set nomaxrow	*	*			*		
set noojflatten	*	*					
set nooptimizeonly	*	*			*		
set noparallel	*	*					
set noprintdbevents	*	*					
set noprintqry	*	*			*		
set noprintrules	*	*					

Keyword	Non 6.4	SQL			QUEL		
		ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
set noqep	*	*			*		
set norules	*	*					
set nosql	*				*		
set nostatistics	*	*			*		
set notrace	*	*			*		
set nounicode_substitution	*	*					
set ojflatten	*	*					
set optimizeonly	*	*			*		
set parallel	*	*					
set printdbevents	*	*					
set printqry	*	*			*		
set printrules	*	*					
set qep	*	*			*		
set random_seed	*	*			*		
set result_structure	*	*			*		
set ret_into	*				*		
set role	*	*					
set rules		*					
set session	*	*			*		
set sql	*				*		
set statistics	*	*			*		
set trace	*	*			*		
set transaction	*	*					
set unicode_substitution	*	*					
set update_rowcount	*	*			*		
set work	*	*					

Keyword	Non 6.4	SQL			QUEL		
		ISQL	ESQL	4GL	IQUEL	EQUEL	4GL
system user	*			*			
to group		*		*			
to role		*		*			
to user		*	*	*			
user authorization	*			*			
with null					*		
with short_remark	*	*					

Other Reserved Keywords

The following reserved keywords are only in the context of a WITH PARTITION= clause.

automatic	partition
hash	range
list	to
null	values
on	with

Appendix C: Features Introduced in Ingres 9.2

This section contains the following topics:

DBMS Server Enhancements

Scrollable Cursors

An ANSI/ISO SQL-92 feature, scrollable cursors are database query result sets that are maintained in the database server as long as the cursor is open and that allow the user to retrieve rows of a result set in any sequence. The number of elements of the result set is easily determined by the application that is using the cursor.

Scrollable cursors greatly ease the display of information, for example, in scrolling web applications.

Scrollable cursors allow an application to move backward and forward through the query results faster. Because scrollable cursors require more overhead than non-scrollable cursors, you should deploy scrollable cursors only where your application requires it.

The scrollable cursor can be declared as either a static or keyset type.

Static scrollable cursors are read only. The complete result set is stored in an internal temporary table.

Keyset scrollable cursors are updatable. Only the key columns and tids of the result set are stored in an internal temporary table

Scrollable cursors are supported in Ingres OpenAPI and JDBC.

For more information, see the *OpenAPI User Guide*, and the *Connectivity Guide*.

LOB Locators

Ingres supports LOB locators in compliance with the ANSI SQL 2003 Standard. LOB locators, which are passed between the client and server, are pointers to the LOB data, which can be stored in or outside a table. They permit the client to access LOB data outside the scope of row data retrieval, so that the client can control the processing and storage requirement of the LOB data.

LOB locators are supported in Ingres JDBC. For more information, see the *Connectivity Guide* and the *OpenAPI User Guide*.

UTF8 Character Set

Ingres supports the UTF8 character set, which lets you store multi-byte UTF-8 encoded Unicode characters into char, varchar, and long varchar strings. The UTF8 character set can be selected during installation.

Support for the UTF8 character set provides compatibility and portability with other database architectures.

Clients installed with the UTF8 character set can connect only to a DBMS Server that uses the UTF8 character set. If UTF8 is the character set for the server, then all clients connecting to this server must also use the UTF8 character set.

If the server character set is UTF8, then by default any database that is created on the server is created Unicode-enabled with Normalization Form C (NFC) with default UNICODE collation, even if it is not explicitly defined. Thus char, varchar, and long varchar columns (as well as nchar, nvarchar, and long nvarchar) use the UNICODE collation by default.

If the database you are connecting to is Unicode-enabled, the UNICODE collation is loaded.

The collation sequence UNICODE_FRENCH is added to support French Unicode collation.

Only char, varchar, and long varchar columns support UTF-8. Ingres character based tools (such as the terminal monitor and ABF) show the data in UTF-8.

Note: When creating a table in an installation set to the UTF8 character set, the column specification for char and varchar columns is in number of bytes (not number of characters).

String functions—such as length(), substring(), and position()—operate similarly on UTF-8 strings.

Coercion is supported between different string types and between string types and other Ingres data types (numeric, datetime, binary, and so on) in a UTF-8 database.

More information can be found in the *Installation Guide*, in the *SQL Reference Guide* under storage formats, and in the *Command Reference Guide* under the createdb and unloaddb command descriptions.

Improved Out of the Box Defaults

Ingres has new defaults for the following configuration parameters, set during installation.

The default page size (default_page_size) for tables is changed from 2K to 8K.

The default buffer cache size (cache_guideline) is configured as medium for all page sizes, resulting in increased size for the following page sizes:

- For 8K page size, increased from 16 MB to 48 MB.
- For 32K page size, increased from 12 MB to 48 MB.
- For 64K page size, increased from 11 MB to 48 MB.

DMF Cache 2K and 8K are enabled by default.

The default transaction log file size (II_LOG_FILE_SIZE_MB) is changed from 32 MB to 256 MB. The required minimum log file size is increased from 16 MB to 32 MB.

Automatic Storage Structure for New Tables

The storage structure of a base table, when created, is automatically determined based on the syntax used for the CREATE TABLE statement. If the CREATE TABLE statement includes at least a primary key, unique constraint, or referential (foreign key) constraint, the base table structure is set to B-tree and the usual secondary index is not built.

If the table definition includes more than one constraint, it chooses the primary key constraint over a unique constraint, and the first unique constraint over any referential constraint. For primary key or unique constraints, it also adds the UNIQUE_SCOPE=STATEMENT attribute to the base table structure. A dependency is added between the constraint and the base table structure so that the constraint must be explicitly dropped and re-added if the base table structure is modified.

This feature improves the initial performance of the table. Previously, all new tables were created with a heap structure, by default.

This feature is enabled or disabled in a given DBMS Server by setting the configuration parameter table_auto_structure to ON or OFF. The default is ON.

Additional SQL Functions

New SQL functions ease application migration.

New numeric scalar functions include:

- round - Returns a numeric value, rounded to the specified length or precision
- ceiling - Returns smallest integer greater than or equal to the argument
- floor - Returns largest integer less than or equal to the argument
- truncate - Truncates x to y decimal places
- atan2 - Arctangent of angle defined by coordinate pair (x, y)
- acos(n) - Arccosine of cosine value n
- asin - Arcsine value of sine value n
- tan - Tangent value of angle n
- pi - Value of pi (ratio of the circumference of a circle to its diameter)
- sign - Returns -1 if n < 0, 0 if n = 0, +1 if n > 0

For further details, see the section Numeric Functions in the *SQL Reference Guide*.

New string scalar functions include:

- chr - Converts integer into corresponding ASCII code.
- ltrim - Returns a character expression with leading blanks removed
- rtrim - Returns a character string with trailing blanks removed
- lpad - Returns specified character string of specified length left-padded by blanks or copies of the second expression
- rpad - Returns specified character string of specified length right-padded by blanks or copies of the second expression
- replace - Replaces all occurrences of a specified string value with another string value
- byteextract - Returns the *n*th byte of the specified string.

  Note: Byteextract can replace charextract, which now handles multi-byte characters and whose returned data type has changed from char(1) to varchar(4).

For further details, see the section String Functions Supported in the *SQL Reference Guide*.

Incremental Rollforwarddb

The incremental rollforwarddb feature allows the journals from a database to be incrementally applied, as they are generated, to a backup copy of the database.

This feature can be used to minimize downtime in the event that the backup database is needed for disaster recovery.

For this feature, two new options are added to the rollforwarddb command:

- -incremental
- -norollback

To apply journals incrementally

1. Start the incremental rollforwarddb by issuing the following command:

   ```
   rollforwarddb dbname +c -j -incremental
   ```

2. Discover and apply new journals by issuing the following command:

   ```
   rollforwarddb dbname -c +j -norollback -incremental
   ```

   The database remains inconsistent and readonly. There may be open transactions.

3. Discover and apply new journals and roll back open transactions by issuing the following command:

   ```
   rollforwarddb dbname -c +j -rollback -incremental
   ```

   The -rollback flag ends the incremental rollforwarddb, and the database is marked consistent and updatable.

For details on the flags, see the rollforwarddb command description in the *Command Reference Guide*.

Improved Exception Handling

Exception handling in the DBMS Server is improved, resulting in better responsiveness when a user interrupts a process, removes a session, or aborts a query using iimonitor.

Increased Precision for Decimal Data Type

The maximum precision for the decimal type is increased from 31 digits to 39 digits. The resulting byte size of decimal columns is increased from a maximum of 16 bytes to a maximum of 20 bytes.

Improved Performance of String Comparisons

The performance of comparisons involving char and varchar values is improved.

Unicode Uppercase and Lowercase

Ingres supports uppercase and lowercase operations on Unicode data types nchar, nvarchar, and long nvarchar.

Fetch First n and Offset n

The OFFSET clause and FETCH FIRST clause on the SELECT statement can be used to return a subset of rows from a result set.

Users can code the "n" value of a FETCH FIRST n or OFFSET n specification in a SELECT statement as a host language variable in embedded SQL applications, or as a parameter or local variable in a database procedure.

This feature is useful in Web-style applications that page results back to the user, as in the results from using a web search engine.

For example, the following query returns rows starting from the 25th row of the result set:

```
SELECT * FROM MYTABLE ORDER BY COL1 OFFSET 25
```

For example, the following query fetches only the first 10 rows of the result set:

```
SELECT * FROM MYTABLE ORDER BY COL1 FETCH FIRST 10 ROWS ONLY
```

A query can use any combination of the ORDER BY, OFFSET, and FETCH FIRST clauses, but in that order only.

The OFFSET and FETCH FIRST clauses can be used only once per query, and cannot be used in unions or view definitions. They cannot be used in subselects, except a subselect in a CREATE TABLE statement or an INSERT statement.

The FETCH FIRST clause cannot be used in the same SELECT statement as SELECT FIRST rowcount.

The syntax for each clause is OFFSET n or FETCH FIRST n, where *n* is a positive integer, a host variable, or a procedure parameter or local variable.

In the FETCH FIRST clause, the keywords FIRST and NEXT, and the keywords ROWS and ROW are interchangeable. Because you can offset and fetch first in the same query, NEXT is an alternative for readability. For example:

```
OFFSET 10 FETCH NEXT 25 ROWS ONLY
```

Cached Dynamic Cursor Query Plans

Query plans for cursors defined with dynamic SELECT statements can be cached and reused, rather than recompiled every time they are prepared. When a subsequent PREPARE is executed on the identical syntax, the query plan in cache is used and the query is not re-optimized.

This feature provides significant performance improvement for complex queries.

A dynamic query associated with a cursor can be cached by using the keyword REPEAT or REPEATED in the prepared select. For example:

PREPARE statement_name FROM REPEATED SELECT…

The configuration parameter, cache_dynamic (available in CBF or Configuration Manager), enables or disables this feature at the server level at startup. When set to ON, all query plans for cursors defined with dynamic SQL will be cached, removing the need to explicitly code the REPEAT or REPEATED keyword in applications or database procedures. The default setting is OFF.

The server-level default setting can be overridden by using a SET statement of the form:

SET [NO]CACHE_DYNAMIC

The server-level setting can be overridden at the session level by using a SET SESSION statement of the form:

SET SESSION [NO]CACHE_DYNAMIC

The DBMSINFO function returns the current setting for the session, indicating whether caching is on or off. For example:

SELECT DBMSINFO ('CACHE_DYNAMIC')

Prior to this feature, the REPEAT or REPEATED keyword could be used on INSERT, SELECT, DELETE, and UPDATE statements coded directly in an embedded program (that is, without PREPARE or EXECUTE IMMEDIATE). This new feature allows REPEATED SELECT statements to be prepared if they use cursors. The REPEATED keyword results in the caching of the query plan.

Changes to ANSI Date Feature

The configuration parameter date_alias (which was introduced in Ingres 2006 Release 2 and replaces date_type_alias parameter) controls whether the keyword DATE used to define the column data type refers to the INGRESDATE (default) or ANSIDATE data type.

In this release (Ingres 9.2), the value of date_alias is communicated from the client to the server. If the client and server have different values for date_alias, then the server uses the date_alias value set in the config.dat file on the client. This ensures that the definition of date_alias intended at the client is preserved in communication to the server.

To change the value of date_alias in the config.dat file, use the iisetres command. For details on the iisetres command, see the *Command Reference Guide*.

Supportability Enhancements

Logging of Verifydb, Chkpdb, and Rollforwarddb

Recovery tools ckpdb, rollforwarddb, and verifydb now log messages when they are executed, stating the database affected and other information. This supportability enhancement makes it easier to verify that recovery tools have been used. For example, it is now easier to tell that verifydb has been used to force a database consistent.

Ability to View Cursor Definition Text for an Executing Fetch

When a cursor FETCH is executing for a user session, the text of the cursor as defined by DECLARE CURSOR is visible using iimonitor or the Interactive Performance Monitor (ipm). For example:

```
Query: open  ~Q  cursor for select a, b, c from t1 for readonly
```

This feature makes the query string being executed available; such information can be useful for performance tuning or problem resolution.

Server Type Reported for Terminated Programs

When a program attached to the Ingres shared memory segment terminates unexpectedly, a message is written to the log to identify the running program and, if it is a command line program, the program arguments. For example:

```
Process (000033B4) died with info 'auditdb -a tstdb'.
```

When database servers and database utilities such as auditdb are executing, they attach themselves to the Ingres shared memory segment. In previous releases, when an error occurred, it was difficult to determine which memory-attached program terminated and, thus, whether the failure was critical or merely a benign error that could, for example, be resolved by re-running the utility program that failed or was canceled.

The new information logged makes it easier to debug problems associated with the shared memory segment.

Note: Because the messages appear in the log when Ingres notices the abnormal termination, their writing may be asynchronous with the program termination itself.

Connectivity Enhancements

LOB Locator Support in JDBC and OpenAPI

JDBC supports an abstraction of LOB locators through the BLOB and CLOB classes. LOB locators allow the client to request data from a specific offset in the LOB without having to retrieve the data in between. For details, see the section LOB Locators in the chapter "Understanding JDBC Connectivity" in the *Connectivity Guide*.

With OpenAPI, an application can request a reference to the long data, called a locator, by setting the IIAPI_QF_LOCATOR flag when calling IIapi_query(). Locators are 4-byte integer values that reference the long data where it resides in the database. For details, see the section LOB Locators in the chapter "Introduction" in the *OpenAPI User Guide*.

Scrollable Cursors in JDBC and OpenAPI

Scrollable cursors are supported in JDBC. ResultSet types TYPE_SCROLL_INSENSITIVE and TYPE_SCROLL_SENSITIVE and the ResultSet scrolling methods are supported. Ingres read-only (static) scrollable cursors are insensitive to changes, while updatable (keyset) scrollable cursors are sensitive to changes. For details, see the section Cursors and Result Set Characteristics in the chapter "Understanding JDBC Connectivity" of the *Connectivity Guide*.

Scrollable cursors are supported in OpenAPI. The application requests a scrollable cursor by setting the IIAPI_QF_SCROLL flag when opening the cursor using IIapi_query() with query type IIAPI_QT_OPEN. A scrollable cursor can be positioned prior to calling IIapi_getColumns() using either IIapi_scroll() or IIapi_position(). IIapi_getColumns() then returns rows starting with the row specified by IIapi_scroll() or IIapi_position(). For details, see the section Scrollable Cursors in the chapter "Introduction" in the *OpenAPI User Guide*.

Connection Pooling in ODBC CLI (UNIX and VMS)

The Ingres ODBC Call-level Interface (ODBC CLI) now supports ODBC connection pooling. Connection pooling allows connections to be shared in ODBC applications and improves performance, especially in multi-threaded applications and applications with a large number of connections.

Connection pooling is set on a per-process basis and is supported in the SQLSetEnvAttr() function. By default, ODBC connection pooling is disabled.

If the ODBC CLI detects that a pooled connection has remained connected past the defined time-out interval, the connection is terminated. A new screen in the Ingres ODBC Administrator utility (iiodbcadmin) allows users to specify the connection timeout value. The minimum timeout value is 1 second; the maximum is 2,147,483,647 seconds.

If a user upgrades his or her Ingres installation without running iiodbcinst, iisuodbc, or iiodbcadmin, the default timeout value is -1, which indicates no timeouts.

Note: ODBC connection pooling is already supported in Windows environments.

.NET Data Provider Enhancements

Enhancements to the .NET Data Provider include:

- New keywords in the connection string:

  dbms_user

    Specifies the user name to be associated with the DBMS session. This keyword is equivalent to the Ingres -u flag, which can require administrator privileges.

  dbms_password

    Specifies the DBMS password for the user. This flag is equivalent to the Ingres -P flag.

  character encoding

    Specifies the .NET character encoding used for conversions between Unicode in the .NET application and character data types in the database. Typically, the character encoding is determined automatically by the data provider from the Data Access Server installation character set. This keyword allows an alternate character encoding to be specified or a valid character encoding to be used if the data provider is unable to map the server's character set.

- Integration with Visual Studio 2005 and Visual Studio 2008 is enabled by additional properties to IngresConnectionStringBuilder. New edit boxes are added to the Connection String Editor dialog.

- The current Interval data type is split into two data types: IntervalDayToSecond and IntervalYearToMonth. The IntervalYearToMonth remains mapped to .NET String data type. The IntervalDayToSecond is mapped to .NET TimeSpan data type.

For details, see the sections Connection String Keywords, Connection String Editor (Data Adapter Configuration Wizard), and Mapping of Ingres Native Types to .NET Types in the chapter "Understanding .NET Data Provider Connectivity" in the *Connectivity Guide*.

Performance Improvements in Network Communications (UNIX and Windows)

Connections across Ingres Net using TCP/IP are set up more quickly, especially in heavily loaded or slow networks, such as wide-area networks (WANs).

Appendix D: Features Introduced in Ingres 2006 Release 2

This section contains the following topics:

New Features in the DBMS Server

Ingres 2006 Release 2 enhancements to the DBMS Server allow for better integration by Ingres partners. New features for application developers include:

- Additional flexibility in application design.

- Simplified migration of existing applications that run against non-Ingres database architectures.

Derived Tables

A *derived table* results when you code a SELECT in the FROM clause of a SELECT or UPDATE statement.

Derived tables let you create or simplify complex queries. Useful in data warehousing applications, they provide a way to isolate complex portions of query syntax from the rest of a query.

Some complex queries cannot be implemented without using either pre-defined views or derived tables. The derived table behaves like an inline view, but is more concise and avoids having to define persistent objects that may be used for a single query only.

For details on derived tables, see the *SQL Reference Guide*.

ANSI Date and Time Support

Ingres now supports the ANSI date and time data types DATE, TIME, TIMESTAMP, and INTERVAL. This enhancement makes it easier for applications to migrate to Ingres from a non-Ingres database.

Previously, Ingres supported one date data type that could store dates, times, intervals, and time stamps. The previous date type is renamed to INGRESDATE.

The configuration parameter date_alias controls whether the keyword DATE used for a column data type refers to INGRESDATE or to ANSIDATE. The date_alias parameter is set during installation and defaults to INGRESDATE.

In Ingres 9.2, the server interprets the date_alias according to the client's config.dat setting. (In contrast, in Ingres2006 Release 2, the client and server each interpret date_alias according to their own config.dat settings. We recommend, therefore, that in Ingres 2006 Release 2, both the client and server have the same value for date_alias.)

When migrating from an earlier version of Ingres, the existing date data in the database is not affected. The data is still a valid INGRESDATE data type.

If you set II_DATE_TYPE_ALIAS to the ANSIDATE format, existing scripts and database procedures that use the keyword date to imply old DATE column definitions, may need to be changed to explicitly use the INGRESDATE keywordtype.

Note: The date_alias parameter replaces the date_type_alias parameter, which is deprecated.

For details, see the *SQL Reference Guide*.

BEFORE Triggers

BEFORE triggers let an application call an Ingres database procedure before a triggering operation (INSERT, UPDATE, or DELETE) is executed. The procedure can change the values of columns in rows being inserted or updated, or can inhibit the deletion of rows, depending on their contents. In the CREATE RULE statement, the keyword BEFORE can be used in defining the table condition that triggers the rule.

In addition, in the CREATE PROCEDURE statement, you can optionally assign formal parameters a mode: IN, OUT, or INOUT. For OUT and INOUT parameters, modified parameter values can be passed back to the calling procedure or triggering operation.

For details, see the *SQL Reference Guide* and *Database Administrator Guide*.

SQL Language Enhancement—Describe Input Statement

To ease application migration, Ingres SQL supports the DESCRIBE INPUT statement. This statement obtains the number and type of input parameters of a prepared statement. Such information is necessary for products that support user-supplied queries.

For details, see the *SQL Reference Guide*.

Indexes on Temporary Tables

Ingres supports indexes on temporary tables. This feature can be used on global temporary tables for more efficient access.

For more information, see the *SQL Reference Guide*.

Syntax for Referencing Temporary Tables

This release introduces a new syntax for referencing global temporary tables. This enhancement facilitates porting applications to Ingres from other database architectures.

The new syntax drops the requirement of prefixing a temporary table name with the SESSION qualifier in a DECLARE GLOBAL TEMPORARY TABLE statement. If the SESSION schema qualifier is omitted in the declaration, all subsequent DML and DDL statements referencing the table can optionally omit it. When this syntax is used, creating temporary and permanent tables with the same name is not allowed, to avoid confusion in referencing tables.

Using the "SESSION." schema qualifier when referencing temporary tables is still required if the DECLARE GLOBAL TEMPORARY TABLE statement includes the SESSION qualifier in the table name. When this syntax is used, temporary and permanent tables can have the same name.

For more information, see the *Database Administrator Guide* and *SQL Reference Guide*.

Sequence Defaults

A column can be defined that automatically takes an increasing sequence value when one is not provided. This feature can be used to create surrogate keys.

For more information, see the *SQL Reference Guide*.

Automatic Coercion Between Integers and Strings

The DBMS Server performs automatic coercion between integer/float data types and char, varchar, nchar, or nvarchar data types. For example, an INSERT can place a quoted string value into an integer column or a database procedure can have an assignment statement that assigns a float value to a char variable—all without using explicit coercion functions such as CHAR() or INT4().

Date Functions

The following date functions extract the specified portion of a date or timestamp: year(), quarter(), month(), week(), week_iso(), day(), hour(), minute(), second(), and microsecond(). For example, year('2006-12-15') returns 2006.

For more information, see the *SQL Reference Guide*.

Ease of Use Enhancements

The following enhancements make Ingres easier to use and support:

- Name Server registration management
- Server management for GCF servers

Name Server Registration Management

The General Communications Facility is enhanced to improve the reliability, performance, and ease of use of your Ingres system. The Name Server registration mechanism no longer erroneously de-registers servers nor requires you to recycle the installation to recover servers. These problems occasionally occurred under heavy DBMS connectivity loads.

The enhancement also solves the problem with manually registering servers. Now when the iinamu utility is used to manually register a server, all necessary information is restored, clients are correctly validated, and attempts to connect with an installation password are not rejected.

As a result of this enhancement, the output of the SHOW SERVERS command in the iinamu utility is now identical in format to other SHOW *class* commands.

Server Management for GCF Servers

The iimonitor utility is an Ingres command line utility used to monitor DBMS servers, Recovery servers, and now GCF servers. The GCF servers support iimonitor commands. This feature provides additional capability to the Ingres system administrator to monitor, diagnose, and control the Name Server, Communications Server, and Data Access Server.

New iimonitor commands that are specific to GCF servers are as follows:

- set trace

- register server

- remove tickets

- remove pooled sessions

For more information, see the *Command Reference Guide*.

Connectivity Enhancements

The connectivity enhancements ease migration of applications from non-Ingres database architectures and improve interoperability.

ODBC Enhancements

The Ingres ODBC 3.5 driver is enhanced to support most of the ODBC 3.x specifications. The enhancements offer these benefits:

- Better integration with Microsoft products such as Access and Excel, and with other third-party products that use ODBC

- Easier migration from other databases, such as Oracle and SQL Server

- Improved performance

The new Ingres ODBC driver includes several new features, including the following ODBC 3.x functions:

- SQLBrowseConnect()

- SQKGetInfo()

- SQLGetTypeInfo()

- SQLDescribeParam()

- SQLColumnPrivileges()

  Note: Not supported on gateways except RMS.

- SQLTablePrivileges()

  Note: Not supported on gateways except IMS, VSAM, and RMS.

The following functions are supported through the ODBC escape sequence syntax:

- CONVERT

- INTERVAL scalar

The following features in Ingres 2006 Release 2 are supported:

- The DESCRIBE INPUT query

- New data types for ISO dates and time intervals

For more information, see the *Connectivity Guide*.

JDBC Enhancements

The following JDBC enhancements ease application migration and performance:

- The Ingres JDBC driver supports the Java 2 Platform Standard Edition 5.0 (J2SE 5.0) specification, excluding RowSets.

- The JDBC cursor default is changed to CURSOR=READONLY. This cursor mode setting typically generates improved JDBC performance, especially for selects returning many rows. A configuration parameter lets you retain the old default, if preferred.

- The JDBC ParameterMetaData interface, which is related to the DESCRIBE INPUT functionality in the DBMS, is supported.

- Enhanced support is provided for XA transactions in J2EE environments using the IngresXADataSource.

For more information on JDBC, see the *Connectivity Guide*.

Ingres .NET Data Provider 2.0

The Ingres .NET Data Provider is enhanced to support the Microsoft .NET 2.0 Framework and MS Visual Studio 2005.

The Ingres .NET Data Provider 2.0 takes advantage of the .NET 2.0 features for greater usability, interoperability, and flexibility of Ingres data access in a Microsoft .NET environment. It supports the new base classes introduced into the data provider class hierarchy by Microsoft. For Visual Studio 2005, the data provider supports the new Server Explorer window, making it easier to manage data source definitions.

For more information, see the *Connectivity Guide*.

Note: The new Ingres.Client assembly of Ingres2006 Release 2 replaces the old Ca.Ingres.Client assembly of Ingres2006. Old applications that want to use the new Ingres.Client assembly require source application changes.

PHP 5 Support

The Ingres PHP driver is updated to support the features of PHP 5.

For more information, see the download page of the Ingres web site.

Support for IPv6 Networks

Ingres supports Internet Protocol version 6 (IPv6) networks. IPv6 provides more addresses for networked devices and more efficient processing than does IPv4.

Use of IPv6 by Ingres is transparent. Ingres immediately begins using IPv6, wherever possible, as the network is migrated to IPv6.

Ingres 2006 Release 2 supports both IPv6 and IPv4 addresses. IPv4 must be used to access versions of Ingres prior to Ingres 2006 Release 2.

For details, see the *Connectivity Guide*.

Supportability Enhancement

This release includes enhanced support for debugging database procedures and for dealing with runtime errors. New trace point QE131 enables the display of the line number within a procedure at which an error condition is detected. In addition, the HELP PROCEDURE command has been enhanced for user written procedures; the procedure text is displayed with line numbers to the left of the procedure source code.

Usability Enhancements

This release contains the following usability enhancements:

- Redesigned installation wizards on Linux and Windows for easier installation.

- Installation wizard for Ingres .NET Data Provider.

- Demonstration database that can be created and populated at the end of the installation process. This database supports the demonstration application, which shows you how to code Ingres applications.

- *Quick Start Guide* describes how to begin using Ingres and connect to Ingres from various application development environments.

Removed or Deprecated Features

JDBC Server Removed

As of Ingres 2006 Release 2, the JDBC Server is removed. It has been replaced by the Data Access Server.

Ingres ICE Deprecated

As of Ingres 2006 Release 2, Ingres ICE (also known as the Web Deployment Option) is deprecated. It will be removed in a future release.

Appendix E: Features Introduced in Ingres 2006

This section contains the following topics:

What Is Ingres 2006?

Ingres 2006 is an enterprise-class open source database. Ingres 2006 was previously released by Computer Associates as Ingres r3.

Features Included in Open Source

The following Ingres components are contributed to the open source community:

- Ingres DBMS and associated database administration tools

- Embedded SQL precompilers

- Character-based querying, reporting, and application development tools

- Connectivity components, including ODBC, JDBC, and the .Net Data Provider

- Ingres Star (formerly Ingres Distributed Option)

- Ingres Replicator Option

- Ingres Web Deployment Option

- TP monitors, including CICS, Tuxedo, and Encina

Features Not Included in Open Source

Features not included in the open source edition are as follows:

- Support for spatial objects
- B1 security

The spatial object library is available for download from www.ingres.com if you have a valid technical support contract with Ingres Corporation.

While the source for the Visual DBA suite is not contributed to the open source community, the suite is included in the Ingres for Windows download.

The following members of the Ingres product family are not contributed to open source and continue to be available for purchase from Ingres Corporation:

- OpenROAD
- Enterprise Access
- EDBC products

New Features for Database Administrators

The new features for database administrators make it easier to administer an Ingres database and allow database administrators to deploy Ingres in a more scalable environment.

Parallel Query

As a multi-threaded server, Ingres has long supported symmetric multi-processing (SMP) systems by creating individual threads to handle user queries, and executing these queries in parallel across all available CPUs. Ingres 2006 introduces the ability to execute individual queries in parallel across all available CPUs in the system, which greatly improves performance. For more information, see the *SQL Reference Guide*.

Key Range Table Partitioning

With the functionality of key range table partitioning, the data in a database can be partitioned based upon the value of a given key, which significantly improves the performance of queries that require a full table scan. For more information, see the *SQL Reference Guide*.

Online Modify

The online modify functionality enables users to modify tables while working online. The DBMS performs the modify processing while allowing concurrent updates to the table. For more information, see the *SQL Reference Guide*.

Ingres High Availability Option

Ingres 2006 provides automatic failover support for Ingres clusters on Sun Solaris and Windows. For more information, see the *System Administrator Guide*.

Unextenddb Utility

Ingres 2006 introduces an unextenddb utility, which provides the ability to unextend a database location. For more information, see the *Database Administrator Guide* and *Command Reference Guide*.

Killing Queries

Ingres 2006 provides the ability to kill a query in another session while leaving the session in place. For more information, see the *Command Reference Guide*.

Numeric Overflow Support in Report-Writer

Ingres 2006 provides support for numeric overflow in Report-Writer. For more information, see the *Command Reference Guide*.

Collation Specification at the Column Level

This feature allows the specification of a collation sequence at the column level that differs from the database default collation sequence. A new optional COLLATE clause is added to the column specification on the CREATE TABLE statement. The COLLATE clause lets you specify a case-insensitive collation for columns that contain Unicode data.

In previous releases of Ingres, the DBA had the option when creating a database of defining a collation sequence to be used for non-Unicode text columns. The Unicode standard default collation sequence was provided for Unicode text columns. These collation sequences were in effect for all columns in the database and could not be changed without recreating the database.

Note: As part of this feature, the data descriptor used throughout the Ingres system changed. This data descriptor is also compiled into imaged ABF applications. After upgrading to Ingres 2006, all ABF applications should be re-imaged. Delete the contents of the ABF object directory, $ING_ABFDIR/*database-name*/*app-name*, and then re-image.

For details on this new feature, see the *SQL Reference Guide*.

System-wide Setting for Default Lock Level

This feature adds a system configuration parameter that allows the Ingres administrator to define the default lock level for the entire Ingres instance.

In previous releases of Ingres, to alter the default lock level the application programmer had to use the SET LOCKMODE statement in the application, which affected only the current session.

The new parameter, system_lock_level, is available to the administrator through Configuration-By-Forms. Valid values are DEFAULT, ROW, PAGE, and TABLE. The DEFAULT value is the default and allows the system to decide the lock level. DEFAULT is the assumed value if the parameter is not present.

Note: Each of the default lock levels is subject to escalation, as in previous releases.

For more information on the default locking level, see the *Database Administrator Guide*.

New Features for Application Developers

The new features for application developers provide additional flexibility in application design and make it easier to migrate existing applications that run against non-Ingres database architectures.

Automatic Sequence Number Generation

Ingres has a sequence facility that provides the ability, through SQL, to create a column in a table that contains a sequentially incremented number for each row.

Users can define a named sequence generator by using a CREATE SEQUENCE statement. The sequence generator can produce values in any context that requires a scalar value using the phrase next value for <sequence name> or <sequence name>.nextval. The sequence generator can be used in the values list of an INSERT statement, in the select list of a query, or anywhere that a scalar numeric value is required.

The sequence facility allows sequences to be defined as integer or decimal values. Ingres permits sequences to be decimal(31), which supports a range of +/- 10^{32}.

Use the ALTER SEQUENCE statement to change the parameter settings for a sequence generator and the DROP SEQUENCE statement to delete a sequence generator.

For more information, see the *SQL Reference Guide*.

No Wait for Lock Requests

Ingres uses a value of NO WAIT for the timeout parameter in the SET LOCKMODE statement to indicate that when a lock request that is made that cannot be granted without incurring a wait, control is immediately returned to the application that issued the request. NO WAIT applies to any lock in a transaction or a lock on one or more specific tables. For more information, see the *SQL Reference Guide* and *Database Administrator Guide*.

Support for New Data Types

Ingres 2006 supports the bigint and tinyint data types.

bigint

The bigint numeric data type stores 64-bit integers. The bigint data type is an implementation of the ANSI standard bigint. Integer8, Int8, and i8 are synonyms for this data type.

The int8 function converts the specified expression, which can be a c, char, varchar, nchar, nvarchar, text, float, money, decimal, integer1, smallint, to a 64-bit integer. Decimal and floating-point values are truncated. Numeric overflow occurs if the integer portion of a floating-point or decimal value is too large to be returned in the requested format.

tinyint

The tinyint numeric data type is a synonym for i1. Supported values are from -128 to +127.

For more information on new data types, see the *SQL Reference Guide*.

Additions to the Visual DBA Suite

Three new Visual DBA (VDBA) Suite tools have been added:

- Visual Database Objects Differences Analyzer
- Visual Configuration Differences Analyzer
- Export Assistant

Visual Database Objects Differences Analyzer

The new Visual Database Objects Differences Analyzer (VDDA) tool allows you to compare groups or individual Ingres database objects, either in the current installation or saved into a snapshot file. It also allows you to visualize the differences in the database objects.

The following options are supported:

- Performing the comparison either at the installation level (that is, comparing any database objects present in an installation) or at the schema level (that is, comparing objects owned by a given user within a database). In both cases, you can limit the comparison to certain object types.

- Saving such groups of database object definitions into snapshot files for later comparison with current or other saved database object definitions. This allows comparisons within an installation over time.

The list of differences generated by VDDA includes one line for each difference in a property of the given database object. The bottom status line indicates the number of objects with differences and the number of differences found in the comparison.

For more information, see online help for VDDA.

Visual Configuration Differences Analyzer

The new Visual Configuration Differences Analyzer (VCDA) tool allows you to compare the configuration information for two Ingres installations. VCDA enables you to take snapshots of the installation and compare these snapshots with either a snapshot taken at some point in the past or with a configuration snapshot taken on another machine.

For example, after you install Ingres and have tuned the configuration to meet your needs, take a snapshot of the configuration. If you encounter problems with the installation later on, take another snapshot of the configuration and compare it to the earlier snapshot to determine if any configuration changes have contributed to the problem. Keep an on-going record of configuration changes by taking a snapshot of the installation each time you change its configuration.

VCDA snapshots contain information taken from the config.dat file, the symbol table, and the vnode database, as well as environment variables set at the system and user level.

Specifically, VCDA allows you to:

- Save a snapshot of the current installation configuration into a file

- Compare two snapshot files, or the current installation with a saved snapshot

- Restore selective groups of configuration parameters from a saved snapshot

VCDA lists the differences with an associated icon that distinguishes between those parameters that are different and those that exist in one snapshot but not in the other.

VCDA uses information from the config.dat file, which typically contains parameters that apply to the local host name only. However, if you want to concatenate the contents of all your config.dat files into a single config.dat file to distribute across multiple environments, VCDA manages this situation as follows:

- The host name under which a snapshot is saved becomes part of the snapshot information.

- If VCDA detects that host names other than the snapshot host name are managed within the config.dat information of the snapshot, VCDA displays them.

- A host name mapping option is available in that situation, so that VCDA can compare the additional host names' configuration parameters. If this option is not used, the parameters are compared, including their host names (that is, only parameters that are identical for the same host name in the two snapshots are considered identical).

For more information, see online help for VCDA.

Export Assistant

The new Export Assistant complements the existing Import Assistant in both design and function. The Export Assistant is a wizard designed to simplify the task of exporting Ingres (or Enterprise Access) data into external files. Specifically, you can export data into the following file formats:

- .csv (and other delimiter formats)

- .xml

- .dbf

- fixed widths

The Export Assistant is accessible from the Start menu and from the Ingres Visual Manager and Visual DBA tools. You can also invoke the Export Assistant from the command line.

Connectivity Enhancements

The following new features provide connectivity enhancements.

Support for JDBC 3.0 API

Support for the JDBC 3.0 API includes three components: an installation server, a Java client driver, and an information utility. For more information, see the *Connectivity Guide*.

Data Access Server

The Data Access Server (DAS) runs as part of a standard Ingres installation. The DAS translates JDBC requests from the Ingres JDBC Driver into Ingres internal format and forwards the request to the appropriate DBMS Server. The DAS supports the same network protocols and port designations as the Communications Server.

The DAS also supports the new Ingres .NET Data Provider component that enables high-performance native .NET access to Ingres data sources and delivers Ingres data to the Microsoft .NET Framework.

Through the DAS, a JDBC client and Ingres .NET Data Provider have full access to Ingres, Enterprise Access, and EDBC databases. The DAS can also access database servers on remote machines using Ingres Net.

For more information, see the *Connectivity Guide*.

JDBC Driver

The Ingres JDBC Driver is a pure Java implementation of the JDBC 3.0 API released with the Sun Java 2 SDK, version 1.4. The driver supports application, applet, and servlet access to Ingres data sources through the Data Access Server.

The JDBC driver provided in Ingres 2.6 continues to be supported in Ingres 2006. For migration instructions related to the JDBC driver, see the *Migration Guide*.

The Ingres JDBC Driver with the DAS supports the following JDBC 3.0 features:

- Boolean data type (similar to Bit)
- Savepoints
- Named procedure parameters
- Auto-generated keys
- Connection Pool Configuration

The Ingres JDBC Driver is delivered as a single Java archive file, iijdbc.jar, located in the library directory (lib) of the Ingres installation. Access to the driver can require, depending on the Java environment used, adding the Java archive to the CLASSPATH environment setting or as a resource in the appropriate utility. For browser/applet access, the Java archive must be copied to the Web Server directories.

JDBC Information Utility

The JDBC 3.0 API support includes a JDBC information utility, JdbcInfo. This utility displays the Ingres JDBC Driver internal release information. The class files for the JdbcInfo utility are located in the library directory (lib) of the Ingres installation.

For more information, see the *Connectivity Guide*.

Updatable Result Sets in JDBC

The Ingres JDBC driver supports updatable result set features of the JDBC 2.1 API. Updatable result sets permit an application to update or delete the current row of the result set, or insert rows into the associated table using methods provided by the JDBC ResultSet class. A new class, RsltUpdt, has been added as an extension to the cursor result set class, RsltCurs, to support updatable result sets. The result set methods associated with the RsltUpdt class are listed below.

The ability to update a result set is determined by calling the following method:

ResultSet.getConcurrency()

The current row of a result set can be deleted using the following method:

ResultSet.deleteRow()

Columns values of the current row can be set using the following methods:

ResultSet.updateAsciiStream()

ResultSet.updateBigDecimal()

ResultSet.updateBinaryStream()

ResultSet.updateBoolean()

ResultSet.updateByte()

ResultSet.updateBytes()

ResultSet.updateCharacterStream()

ResultSet.updateDate()

ResultSet.updateDouble()

ResultSet.updateFloat()

ResultSet.updateInt()

ResultSet.updateLong()

ResultSet.updateNull()

ResultSet.updateObject()

ResultSet.updateShort()

ResultSet.updateString()

ResultSet.updateTime()

ResultSet.updateTimestamp()

Once column values have been set, the changes can be saved or dropped using the following methods:

ResultSet.updateRow()

ResultSet.cancelRowUpdates()

To insert a row, the result set current position must be moved to a special reserved row. The following methods control the positioning of the result set and the insertion of rows:

ResultSet.moveToInsertRow()

ResultSet.moveToCurrentRow()

ResultSet.insertRow()

The following methods can be used to determine the status of a result set row:

ResultSet.rowDeleted()

ResultSet.rowInserted()

ResultSet.rowUpdated()

For more information, see the *Connectivity Guide*.

.NET Data Provider and Visual Studio .NET Integration

This release of Ingres introduces support for the Microsoft .NET Framework and Visual Studio .NET application development tools.

Ingres .NET Data Provider

The Ingres .NET Data Provider is a .NET component that enables high-performance native .NET access to Ingres data sources and delivers Ingres data to the Microsoft .NET Framework.

The Ingres .NET Data Provider offers a series of .NET types to describe the user's data, .NET provider classes to manipulate the data, and connection pooling to efficiently manage data connections.

The design and naming conventions of the Ingres .NET Data Provider's data types, classes, properties, and methods follow the same pattern as the Microsoft .NET Data Providers. Consequently, developers who are familiar with the Microsoft providers can easily develop or convert existing code from Microsoft databases to Ingres databases.

All Ingres .NET Data Provider modules are written in C#, a managed .NET language with full access to every .NET Framework capability. Even though the data provider is written in C#, any managed language such as VB.NET or J# can use the data provider because of .NET's language interoperability feature.

For more information, see the *Connectivity Guide*.

Visual Studio .NET Integration

The .NET Framework was written with design-time support. Integration with the Visual Studio .NET visual tools allows programmers to drag-and-drop the Ingres .NET Data Provider design component onto a design surface such as a the Windows Form Control (WinForm). Integration also includes the following design components:

- Data Adapter Configuration Wizard: Enables programmers to specify the design properties of the Ingres DataAdapter object.

- Query Builder: Enables programmers to build SQL statements that the Ingres .NET Data Provider uses to retrieve and modify database information.

- Parameter Collection Editor: Enables programmers to add parameters to the Command component.

For more information, see the *Connectivity Guide*.

Ingres ODBC Administrator

Prior to this release, ODBC users on non-Windows platforms were required to manually edit the odbc.ini configuration file to define a data source. This release introduces an Ingres ODBC Administrator utility for non-Windows platforms. Supported platforms include UNIX, Linux, and VMS. The new ODBC Administrator enables users to:

- Create, edit, and delete data source definitions

- View configuration details about a particular data source

- Display a list of installed drivers and view configuration details for a selected driver

- Define an alternate path for accessing driver definitions

- Define an alternate path for accessing data source definitions

- Turn ODBC tracing on or off

- Test a data source connection

For more information, see the *Connectivity Guide*.

WinSock 2.2 API TCP/IP Protocol Driver for Windows

This release provides a new Windows TCP/IP protocol driver that takes advantage of the latest Windows Winsock 2.2 API. The new driver (tcp_ip) removes an architectural limitation of the previous implementation (wintcp), which resulted in a performance problem when a client application used many INSERT statements or a large number of single SELECT statements. Also, connection attempts sometimes failed when multiple connects were attempted simultaneously.

To allow a smooth migration from the existing protocol driver to the new one, the existing wintcp protocol driver is included in this release, but will be removed in the future. We recommend that you use the tcp_ip protocol driver.

For more information, see the *Connectivity Guide*.

ODBC Call-level Interface

The Ingres ODBC Call-level Interface (CLI) provides access to the ODBC application environment without the need to use third-party software. It is installed when you install the Ingres ODBC Driver and is supported on all platforms on which Ingres runs.

The Ingres ODBC CLI performs the following functions:

- Optionally determines driver characteristics from ODBC configuration files

- Loads and unloads the ODBC driver into and from application memory

- Maps the driver manager API to the driver API

- Performs basic error checking

- Provides thread safety

- Provides ODBC tracing

- Provides function templates, type definitions, and constant definitions for ODBC applications

For more information, see the *Connectivity Guide* and the *System Administrator Guide*.

New Features for Linux

The following features are new for the Linux platform.

RPM Packaging

Ingres 2006 for Linux is packaged using the Red Hat Package Manager.

KDE/GNOME Desktop Integration

When Ingres 2006 is installed on Linux, it installs the Ingres for Linux *Getting Started* guide on the desktop. It also creates two program groups, one for the Visual DBA suite and the second for the complete set of Ingres documentation. The user has the information needed to start and use Ingres within minutes.

Changes to Existing Features

This section describes changes made in Ingres 2006 to existing features.

Enhanced Unicode Support

Unicode is a standard method of storing character data for multinational situations. Many applications require the ability to store and retrieve Unicode data. Ingres 2006 supports the second phase of a phased implementation of full Unicode support and extends support to the UTF-16 encoding scheme.

For details on this feature, see the *SQL Reference Guide*, the *OpenSQL Reference Guide*, and the *Command Reference Guide*.

Unicode Coercion

Ingres coerces between Unicode data types and non-Unicode data types, including nchar/nvarchar and char/varchar types. Unicode coercion includes the following features:

- Unicode and non-Unicode columns can be joined together in a query.

- The Ingres COPY statement (and by inference, COPYDB and UNLOADDB) can be executed to generate or load non-binary data files where a table contains at least one Unicode column.

- Query strings containing non-Unicode literals that reference Unicode columns are coerced to Unicode before the DBMS executes the query.

Collation Sensitive Support for Wildcard Searching

Ingres performs wildcard searches involving Unicode strings by making use of a case insensitive collation table.

Complex Query Optimization

The Ingres query optimizer has been enhanced to handle queries that reference large numbers of tables and indexes with shorter response time. For more information, see the *SQL Reference Guide*.

Increased Range Table Limit

The limit for the number of table references in a query has increased from 30 to 126.

This limit refers to the sum of explicit table references (even if the same table appears several times in a query), explicit view references, and tables and views included by the expansion of a view definition. It also refers to the sum of such references in all subselects and unioned selects of a query.

For more information, see the *SQL Reference Guide*.

JDBC User ID Enhancements

In previous releases of Ingres, the JDBC driver required a user ID and password that was valid in the context of the DBMS platform, even for local connections where the local user ID is sufficient. The JDBC driver also failed to use virtual node (vnode) login information when making remote connections, and was not able to access private vnode definitions.

In Ingres 2006, the JDBC driver no longer requires a user ID and password when the Data Access Server is running on the same platform as the Java client. Instead, for local connections, the local user ID is sufficient to establish the DBMS Server connection.

When the Data Access Server and DBMS Server are on separate platforms, a vnode is required in the target database specification that defines how the connection between the Data Access Server and DBMS Server is to be established. The vnode provides the connection information while the JDBC user ID and password are used to access the remote DBMS Server.

A new driver connection property/URL attribute allows the JDBC application to control the context (Data Access Server (local) or DBMS Server (remote)) in which the user ID is used.

When used in the Data Access Server context, the user ID and password allow access to the private vnode information for the user ID provided, and both the login and connection information from the vnode is used to access the remote DBMS Server.

When used in the DBMS Server context, global vnode definitions are used for (nonsensitive) connection information and the provided user ID and password are used to access the remote DBMS Server.

For more information, see the *Connectivity Guide*.

Increased Column Limit

The limit on the number of columns per table has increased from 300 to 1024. This feature is upwardly compatible; however, programs written to take advantage of the new limit cannot be used with earlier Ingres releases that included the 300-column limit. For more information, see the *SQL Reference Guide*.

VDBA Enhancements

The following VDBA enhancements have been implemented for Ingres 2006. These enhancements are available only on Windows, Solaris, HP-UX, and AIX platforms.

VDBA Architecture Split

Two VDBA utilities, SQL/Test and Performance Monitor, have been split from the VDBA architecture and are now called Visual SQL and Visual Performance Monitor. By isolating these components as smaller, stand-alone executables, users benefit from increased response time because they no longer need to launch the whole VDBA executable.

When launched in VDBA, these new components also help avoid VDBA locking itself. This can happen, for example, when there is an uncommitted query on a table in an SQL Test window, and an operation is performed on the same table in a DOM window opened in the same VDBA instance.

The SQL Assistant has also been split into a stand-alone DOM component. The SQL Assistant can now be accessed from the Ingres Export Assistant, in addition to Ingres Network Utility and VDBA, to help build SQL queries.

Visual SQL and Visual Performance Monitor are accessible from the Start menu, Ingres Visual Manager, Network Utility, VDBA, and the command line.

Visual SQL

The new stand-alone executable (vdbasql) for Visual SQL displays the following window.
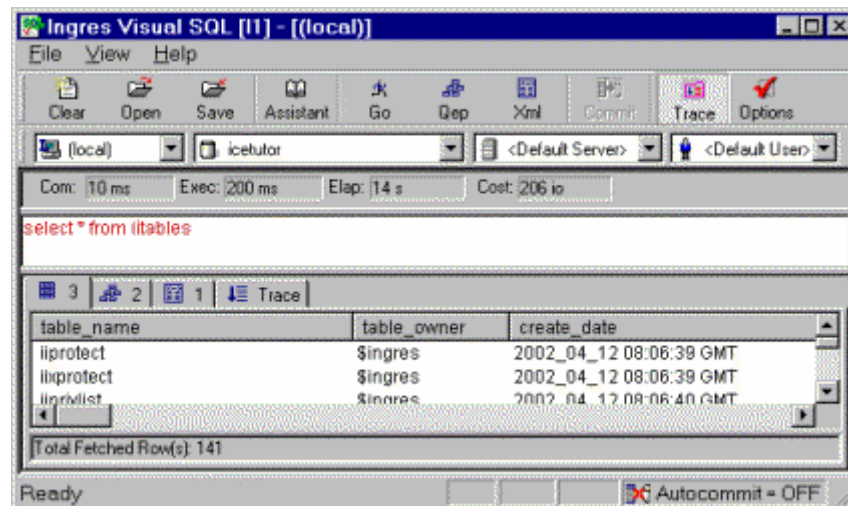
This window allows all the functionality of the previous SQL/Test window in VDBA, but has the following additional toolbar controls:

- New combo-boxes for choosing a node, server class, and impersonated user. (The equivalent function is provided in VDBA through the branch chosen in the nodes tree when launching an SQL/Test window.)

- A Commit button that allows you to directly commit the current transaction without the need to type commit in the query area.

- An Options button that provides access to the properties of the SQL Test control. (This is equivalent to the SQL/Test Preferences in previous VDBA releases, plus the session timeout parameter that was global to VDBA.)

In addition, the status bar now provides the autocommit state of the transaction.

Visual Performance Monitor

The new stand-alone executable (vdbamon) for Visual Performance Monitor displays the following window.
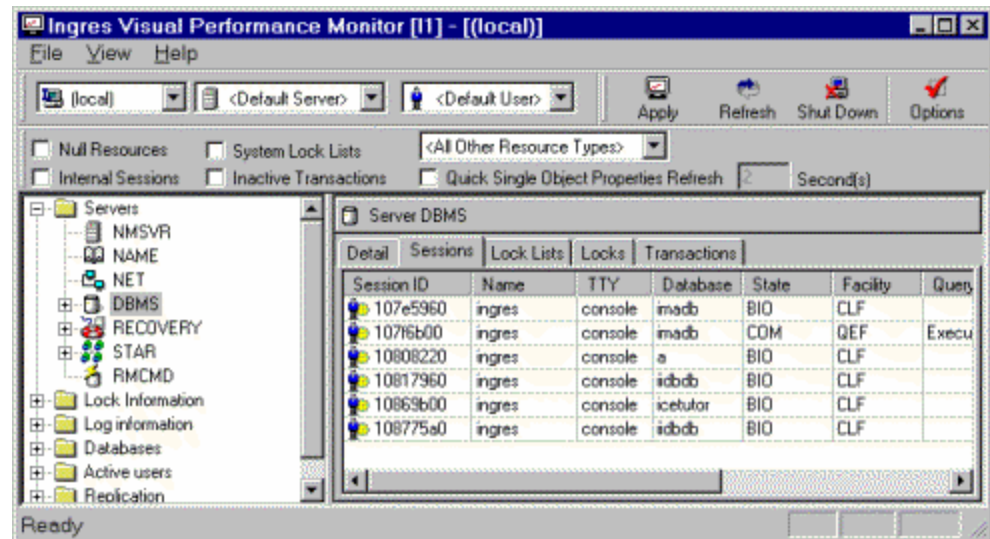
This window allows all the functionality of the previous Performance Monitor window in VDBA, but has the following additional toolbar controls:

- New combo-boxes for choosing a node, server class, and impersonated user. (The equivalent function is provided in VDBA through the branch chosen in the nodes tree when launching a Performance Monitor window.)

- An Apply button that displays the monitor information corresponding to the chosen node (and, if applicable, server class and impersonated user).

- An Options button that provides access to the properties of the Performance Monitor control. (This is equivalent to the Monitor Preferences in previous VDBA releases, plus the background refresh preference, the Grid In List option, and the session timeout and max number of sessions parameters that were global to VDBA.)

Additional Properties

The Properties dialog is accessible from the new Visual SQL and Visual Performance Monitor executables, as well as from VDBA. (This was previously called the Preferences dialog.)

The SQL/Test Priorities dialog now includes a session timeout parameter that was previously global to all VDBA sessions.

The Performance Monitor Properties dialog includes the following additional parameters:

- Background refresh for monitor windows (includes the choice of having background refresh active or not)

- Session timeout and the maximum number of sessions used within the Performance Monitor control's internal session cache

- Grid In List option that displays a grid in the Detail Information pane that allows for easier viewing of data

In addition, Visual SQL and Visual Performance Monitor provide a new menu option, Save Preferences As Default. If this option is selected, which is the installation default, the properties are permanently stored and used every time the utility is executed. If this option is unselected, the selected properties are only used for the utility's current instance.

Notification of Metadata Changes

Notification of changes to Ingres metadata includes the following features in VDBA:

- VDBA is notified by the DBMS of any metadata changes and automatically refreshes itself.

- The refresh preferences are no longer available.

- A new preference (called other servers) provides background refresh for those servers that do not provide metadata change notification (for example, Ingres DBMS with releases older than Ingres 2006 and gateway servers).

- The background refresh of monitor data and node definitions are placed in the Monitor and Nodes preferences.

For more information, see the online help for VDBA.

Ingres Visual Manager Enhancements

The Ingres Visual Manager contains the following enhancements:

- Direct access to all Ingres tools

- Can register alerts in the OS event log

- Message explanations

Note: These enhancements are only available on Windows, Solaris, HP-UX, and AIX platforms.

Direct Access to All Ingres Tools

Ingres Visual Manager provides direct access to all Ingres visual tools. The only exception is the Ingres Service Manager; its full functionality is already provided by Ingres Visual Manager. You can access these tools in the following ways:

- Right-click the tray toolbar to display a menu:

The Ingres 2006 documentation is accessible from the menu.

- Select a tool from the IVM toolbar.

Only the most frequently used tools are accessible from this toolbar. Use the above menu to access all other tools.

Registering Alerts in the OS Event Log

When an Alert message is written to the Ingres errlog.log file, IVM indicates the alert through a special icon change in the tray toolbar and in the IVM window tree. IVM also lets you set additional preferences for alert notification using the Preferences dialog. These additional preferences include:

- **Sound** – When selected, you are alerted of an event by a sound (beep).

- **Message Box** – When selected, you are alerted of an event by a message box if there are unread Alert messages.

A new preference has been added to the Preferences dialog:

- **OS Event** (for new Events only) – When selected, the full text of all Alert messages resulting from new events are logged in the Operating System [Application] Event log in addition to the Ingres errlog.log file. This new preference allows third-party tools to be signaled by Ingres Alert messages.

In addition, if the operating system supports it (as with Windows), the message category (class) and number is stored as information belonging to the event. External applications that are monitoring error messages for the Ingres installation are given the error category and number without having to parse the text of the message.

If the OS Event option is selected, you can also set preferences for generating the following specific operating system events:

- One event is generated whenever the OS Event option checkbox has been selected or deselected and the Preferences dialog has been validated.

- One or more events are generated when the Alert States have been changed (for example, from "alert" to "non-alert") in the Messages Categories and Notification Levels dialog.

- One event is generated if IVM detects that the errlog.log file has been manually changed.

Viewing Message Explanations

To help you respond to Ingres messages, IVM now displays an explanation for any message in the errlog.log file.

Select the desired message on any page or window in IVM that allows message selections and view its explanation by clicking the Message Explanation toolbar button.

Configuration Rules System Enhancements

The Configuration Rules File System has been modified to handle negative values and decimal values.

Shadow Copy of the Symbol Table

When setting, unsetting, or changing environment variables, a backup of the symbol.tbl file (symbol.bak) is maintained. If the original symbol.tbl file becomes corrupted, you can use the backup symbol table file to restore it. A history of updates to the symbol.tbl file is maintained in the symbol.log file. The symbol.bak and symbol.log files are located in the same location as the original symbol.tbl file. For more information, see the *System Administrator Guide*.

Additional Join Functionality

In addition to inner, left, right, and full joins, users can request cross joins (effectively inner joins without an ON clause) and natural joins. Also, the ON clause can be replaced by a USING clause that contains a list of columns, each of which appears in both tables being joined. Instead of the explicitly coded join qualification of the ON clause, the USING clause applies one equijoin predicate for each column pair in the list of columns. For example, "… a left join b using c1, c2 …" is identical to coding "… a left join b on a.c1 = b.c1 and a.c2 = b.c2 …." For more information, see the *SQL Reference Guide*.

Improved Out-of-the-Box Configuration Defaults

Ingres configuration defaults have been updated to reflect the current hardware environments.

Improved IMA Support

Ingres 2006 introduces an additional Ingres Management Architecture (IMA) component, IMP.

IMA provides the framework for accessing system data for monitoring and managing installations through SQL without affecting the underlying operation of the product. For more information, see the *System Administrator Guide*.

CREATEDB Enhancements

Ingres 2006 adds a flag to CREATEDB, which allows the specification of a non-default page size for catalogs. For more information, see the *Command Reference Guide*.

ALTERDB Enhancements

Ingres 2006 adds three new flags to the ALTERDB utility, which allow for the deletion of invalid checkpoints, the deletion of a specific checkpoint, and a non-Unicode enabled database to be Unicode enabled. For more information, see the *Command Reference Guide* and the *Database Administrator Guide*.

Terminal Monitor Enhancements

Ingres 2006 adds support for the –p flag to the terminal monitor (that allows a password to be specified) and support for the –r flag (that allows a role name and optional role password to be added).

Also, command completion and command history recall are supported for the Linux operating system.

For more information, see the *SQL Reference Guide* and the *Command Reference Guide.*

Enhancements for Log Full

Ingres 2006 allows the user to dictate behavior when a log full situation occurs. The options provided are COMMIT, ABORT, or CONTINUE. For more information, see the *SQL Reference Guide*.

Extended B-tree Limits

Ingres 2006 extends the 440-byte limit on the maximum width of a B-tree key.

Installer Enhancements

Ingres on Windows environments is delivered in Microsoft Windows Installer format. Microsoft Windows Installer version 2.0 is required on the machine on which you are installing Ingres components.

The installer on Windows presents a setup wizard to guide you through the installation process. The setup wizard lets you select a Complete or Custom install. The Custom installation lets you choose individual components for installation.

Installation as a User Other Than ingres

With Ingres 2006, the product can be installed as a user other than ingres. Ingres security is retained. When installing Ingres, you can specify a user ID (and its associated group ID) that owns the Ingres installation. During installation, this system administrator ID is automatically created and given the required permissions. If they are not previously defined, the user ID and group ID are added to your UNIX or Windows systems.

Supportability Enhancements

Product enhancements that improve the supportability of Ingres include dumping queries to the error log file upon certain error conditions (for example, an optimizer time-out or an exhaustion of resources), and modifying the ingstart process so that the exact version and patch information is written to the error log file on startup.

Help System Enhancements

Each Ingres visual tool now has its own Help system that is independent of other visual tools. This separation of Help systems makes it easier and quicker for users to navigate through the list of Help system topics. In addition, search results now display only those topics that are applicable to the visual tool in use. Standalone Help systems are now provided for the following visual tools:

- VDBA
- Configuration Manager
- Ingres Visual Manager
- Import Assistant
- Export Assistant
- Journal Analyzer
- Network Utility
- Visual SQL
- Visual Performance Monitor
- Visual Database Objects Differences Analyzer
- Visual Configuration Differences Analyzer

Appendix F: Features Introduced in Ingres 2.6

This section contains the following topics:

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.

SUBSTRING Function

The ANSI compliant SUBSTRING function has been added to the SQL syntax. The SUBSTRING function is often easier to use than combinations of LEFT and SHIFT functions that Ingres traditionally supported. The syntax is:

substring(string-expr from start-column for length)

and the **for length** clause is optional.

New Aggregate Functions

Additional aggregate functions for statistical analysis have been added:

- The function **corr** computes a correlation coefficient
- Functions **covar_samp** and **covar_pop** compute covariance
- A collection of **regr_xx** functions generate regression analysis results

For details, see the *SQL Reference Guide*.

Increased 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 is increased to 32000 bytes, the maximum row size supported in Ingres 2.6.

User-Visible DBA Enhancements

Enhancements have been made to 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 modifies 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 Journal Analyzer are included:

- 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.

For the syntax of the auditdb command, see the *Command Reference Guide*.

Copydb Utility

The copydb utility is 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 in which 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 initial support for raw data locations on UNIX platforms. Raw data locations provide dramatic performance improvements over cooked locations. In this release, only one table may occupy any given raw location. Many raw locations can exist on a single raw disk slice.

GatherWrite Threads

A new internal 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. 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 Configuration-By-Forms. 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. For the syntax of the XML import/export utility, xmlimport, see the *Command Reference Guide*.

Journal Analyzer

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

Import Assistant

The 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 Import Assistant utility, see the online help.

Automated Creation of Location Directories

Before Version 2.6, the Ingres DBA had to manually create the directories for alternate locations as prescribed in the *Database Administrator 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 created, altered, or extended.

This process is clarified and simplified in Ingres 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 the section Create an Area in UNIX in the *Database Administrator Guide*, 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

Notes:

- 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.

Remote Command Server Enhancements

Users commonly encounter problems running utilities that require exclusive access to the iidbdb database because the Remote Command Server process (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 ODBC 3.5 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 aggregate handling allows Ingres to better support data-mining products such as CleverPath OLAP, which make extensive use of data aggregation.

Previous versions required a sort before doing grouping and aggregation. Ingres 2.6 now does grouping with hash bucketing instead of sorting. Hash grouping is usually faster than sorting. Other internal refinements streamline the calculation of common aggregates, reducing the amount of CPU time needed.

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 locking system to eliminate or minimize bottlenecks identified when running various performances tests.

Preallocated RSB/LKBs

Each resource RSB now has an embedded 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 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 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 Ingres 2.6 through integration with an existing Web application development environment.

ICE Development Environment

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

ODBC Enhancements

The ODBC driver has been updated to Version 3.5.

Functions Supported by ODBC Driver

The Ingres ODBC driver supports all level one functions, as well as the following level two functions:

- SQLExtendedFetch (through Microsoft Cursor Library only)
- SQLForeignKeys
- SQLMoreResults
- SQLNumParam
- SQLPrimaryKeys
- SQLProcedureColumns
- SQLProcedures
- SQLSetPos (through Microsoft Cursor Library only)

Unavailable Features in the ODBC Driver

The Ingres ODBC driver does not provide the following features as of Release 2.6:

- Executing functions asynchronously
- Translation DLL
- Support for Ingres SQL COPY TABLE command
- Support for Ingres SQL SAVEPOINT command

JDBC Enhancements

The following JDBC 2.0 extensions have been added to Ingres:

- Compatibility with GA release (protocol levels)
- Execution in JDK 1.1 environment due to a new driver
- Batch processing
- javax—two-phase commit
- javax—client connection pooling
- Updatable 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 in future releases.

In this release, the DBMS supports three new data types:

- nchar
- nvarchar
- 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 can be used in indexes and database statistics.

The native two-byte (UCS2) format is supported and maintained through the entire 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.

OpenAPI version 3 was added to indicate support for these new data types.

VDBA also supports the 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 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 to Support Euro Currency Symbol

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

To set the money format to the Euro currency symbol you must issue the following command:

```
ingsetenv II_MONEY_FORMAT L:€
```

Alternatively, you can set this value 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 includes the Euro sign. Users wishing to use the Euro symbol in a Windows GUI environment 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 either issue the following command:

```
ingsetenv TERM_INGRES IBMPCD
```

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

UNIX:

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 character set, be aware that this is not typically supported because the new character set could display existing characters in your databases incorrectly. 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 with caron
A8	¨	Diaeresis	š	Latin small letter 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

Differences Between ISO 8859-1 and ISO 8859-15 Character Sets

Appendix G: Features Introduced in Ingres II 2.5

This section contains the following topics:

Sort Enhancements

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

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:

- 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 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 or 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 to 10 percent 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 II 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 they were in previous releases.

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 to 50 percent in testing). Even single CPU machines benefit somewhat, because sort I/O and sort computation can be overlapped. 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 or 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 in DMF Sort Enhancements (see page 209) 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.

Ingres II 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.

Ingres II 2.5 also introduces the ANSI/ISO notion of referential actions for the definition of referential (foreign key) constraints. In releases prior to Ingres II 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.

Ingres II 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 II 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 II 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. Delays caused by this operation could show up in unexpected situations, such as the use of statement level rules.

In Ingres II 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 II 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 II 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 II 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 log file in Ingres II 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. Ingres system administration is made simpler, as the transaction log file can be expanded by adding another partition, instead of resizing an existing file or partition. The full log file paths are now defined through Configuration Manager or CBF rather than through the symbol table.

Optimizer and Optimizedb Enhancements

A variety of enhancements have been made to optimizedb.

The 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.

The query compiler has been enhanced to compile more efficient strategies for complex queries involving aggregate views and unions.

Read-only Database Support

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

Example: Create a Read-only Database

For example, to create a read-only database called mydatabase on UNIX:

1. Log in as the installation owner.

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. Run ingstop to cleanly shut down the Ingres installation. This will ensure a consistent copy of the database.

   ```
   ingstop
   ```

7. Copy the database directory and its subdirectories to the new area:

   ```
   cp -r mydatabase /stagingarea/ingres/data/default/
   ```

8. Restart Ingres:

   ```
   ingstart
   ```

9. Change location to the staging area:

   ```
   cd /stagingarea
   ```

10. Copy the directory structure from the staging area to the CD-ROM or other device:

    ```
    cp -r ingres /cdrom
    ```

11. Create a new database location in the target installation using the create location statement or using the accessdb utility:

    ```
    create location cdromloc with area='/cdrom', usage=(database);
    ```

12. Use the createdb command to access the read-only database in the target installation:

    ```
    createdb -rcdromloc mydatabase
    ```

New SQL Functionality

New SQL operations have been added, bringing Ingres SQL closer to the SQL standards. Enhancements have been made to the internal performance that concern bit-wise operator support and miscellaneous functions.

Order By/Group By Expression

The ORDER BY and GROUP BY statements now allow an expression instead of being limited to column names. ORDER BY can also reference a column name or expression that is not part of the select result list.

CASE Expression

An ANSI SQL '92 compliant CASE expression has been added. The CASE expression allows if-then-else testing anywhere that an expression is allowed.

There are two syntax forms. The most general CASE expression is:

```
CASE WHEN boolean-expression THEN expression
        WHEN boolean-expression THEN expression
        ...
        ELSE otherwise-expression
END
```

Each boolean-expression is evaluated in turn, and if TRUE, the corresponding then expression is the CASE result. If all the boolean-expressions are of the form expr1 = expr2, a shorthand form can be used:

```
CASE expr1 WHEN expr2 THEN expression
        WHEN expr3 THEN expression
        ...
        ELSE otherwise-expression
END
```

Parallel Index Creation

A new variation of the CREATE INDEX statement allows the user to create multiple secondary indexes with a single pass through the base table. After the required base table columns are extracted, the indexes are created in parallel, each one using an independent worker thread. For additional performance, any necessary sorting is performed using the new parallel sort capability.

The new syntax is:

```
CREATE INDEX (index-spec), (index-spec), ...
```

where an index-spec looks similar to the original CREATE INDEX statement:

```
(index-name ON base-table (column-list) WITH with-clause)
```

SELECT Enhancement

The SELECT statement now allows the FIRST n clause in the result list. This clause limits the result returned to the user to the first N rows.

Bit-wise Operator Support

The following functions have been added to Ingres II 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.

Aggregate Functions

New aggregate functions have been added:

- stddev_samp

- stddev_pop

- var_samp

- var_pop

The _pop forms divide by group size, the _samp forms divide by group size minus one.

Miscellaneous Functions

The following miscellaneous functions have been added to Ingres II 2.5:

intextract

Extracts the number at the given location.

ii_ipaddr

Converts an IP address to a byte 4.

random, randomf

Generates random integer or float8 values

power, ln

Are ANSI compliant synonyms for ** and log functions.

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).

Extended Date Support

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

Large File Support

A major enhancement to Ingres II 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, system catalogs can use pages larger than 2 KB. As a result, the user does not have to configure a 2 KB cache size in the DBMS for system catalogs.

Row Locking for System Catalogs

For improved concurrency, the Ingres II 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 Configuration-By-Forms or 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 II 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 II 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 II 2.5, the DBMS uses a value locking protocol for serializable transactions with an equal predicate, thereby allowing better concurrency.

Query Optimization and Execution Enhancements

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

Ingres Star Features

In Ingres II 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 Net Features

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

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

Ingres ICE Security Enhancements

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.

Ingres ICE Session Management Enhancements

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 HTML template files. In Ingres II 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 II 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 II 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

Replicator Enhancements

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

Generic Replicator Server

Ingres II 2.5 introduces a generic Replicator Server that is functionally identical to the custom repserver built by the user in previous 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, 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.

OpenAPI Enhancements

Environment handles were added to OpenAPI. Environment handles allow greater application control over configuration of the OpenAPI runtime environment. OpenAPI version 2 was added to indicate support for environment handles.

Extensions were made to the following OpenAPI functions to support environment handles:

- IIapi_initialize() to allocate environment handles

- IIapi_connect() to open connections using environment settings

Three new environment handle associated functions were added:

- IIapi_setEnvParam() for setting environment configuration parameters

- IIapi_formatData() for converting data values relative to environment settings

- IIapi_releaseEnv() to release environment handle resources

Also added is the function IIapi_abort() to forcefully shut down connections under error conditions.

Index

features in 2.5 • 219
startup • 52, 93
    disable • 65
statdump command • 61
storage structure • 146
    reapply • 113
supportability • 162, 190
symbol table • 188
syntax, Report-Writer • 26
system
    backup • 35, 46
    practice upgrade • 41
    preparation • 35
    restore • 35
    testing • 39
system monitoring shellscripts • 36
system_maintained column name • 34

T

table
    default storage structure • 146
    references • 180
    storage structure • 146
    temporary, indexes • 157
    temporary, referencing • 157
TCP/IP • 162, 178
Terminal Monitor • 189
terminated programs • 152
testing • 34
thread implementation on Linux in 2.6 • 200
transaction
    log size • 94

U

unextenddb • 167
Unicode • 149, 168, 180, 203
UNIX kernel parameters • 38
unload database • 60
unload directory, create • 57, 108
unload upgrade • 55, 94
unload/reload procedure • 57, 94
    when to use • 15
unloaddb command • 31, 33, 58, 60, 106, 109
    output • 110
update mode locking • 218
UPDATE…FROM semantics change • 90
upgrade
    applications • 21

hardware issues • 18
planning • 14
using unload/reload procedure • 55, 94
using upgradedb procedure • 44
upgradedb procedure • 43, 107
    when to use • 15
upgradedb utility • 52
user • 190
user access, disable • 44
user check • 58, 97
usermod utility • 194

V

value locking protocol • 219
Visual DBA • 170, 171, 181, 182, 222
Visual Studio • 177
visual tools • 171, 183, 185, 186
VMSinstal, running • 82

W

work location, create • 102

X

xml import/export utility • 196