

Ingres[®] 9.2 for OpenVMS

RMS Gateway User Guide

INGRES

ING-92-RMS-01

This Documentation is for the end user's informational purposes only and may be subject to change or withdrawal by Ingres Corporation ("Ingres") at any time. This Documentation is the proprietary information of Ingres and is protected by the copyright laws of the United States and international treaties. It is not distributed under a GPL license. You may make printed or electronic copies of this Documentation provided that such copies are for your own internal use and all Ingres copyright notices and legends are affixed to each reproduced copy.

You may publish or distribute this document, in whole or in part, so long as the document remains unchanged and is disseminated with the applicable Ingres software. Any such publication or distribution must be in the same manner and medium as that used by Ingres, e.g., electronic download via website with the software or on a CD-ROM. Any other use, such as any dissemination of printed copies or use of this documentation, in whole or in part, in another publication, requires the prior written consent from an authorized representative of Ingres.

To the extent permitted by applicable law, INGRES PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IN NO EVENT WILL INGRES BE LIABLE TO THE END USER OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USER OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF INGRES IS EXPRESSLY ADVISED OF SUCH LOSS OR DAMAGE.

The manufacturer of this Documentation is Ingres Corporation.

For government users, the Documentation is delivered with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227-7013 or applicable successor provisions.

Copyright © 2008 Ingres Corporation. All Rights Reserved.

Ingres, OpenROAD, and EDBC are registered trademarks of Ingres Corporation. All other trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Contents

Chapter 1: Introducing the RMS Gateway

RMS Gateway	1-1
How the RMS Gateway Works	1-2
Gateway Query Processing	1-3
In This Guide	1-4
Audience	1-4
Terminology	1-5
Query Languages	1-5

Chapter 2: Preparing to Use the RMS Gateway

RMS Gateway and Its Environment	2-1
Ingres Data Manager	2-2
Ingres Components	2-2
Ingres Net	2-3
Ingres Star	2-4
Accessing Data Through the RMS Gateway: A Summary	2-5
Using OpenSQL For Portability	2-6

Chapter 3: Creating RMS Gateway Databases

What Is an RMS Gateway Database?	3-1
Creating RMS Gateway Databases	3-2
Naming RMS Gateway Databases and Database Objects	3-2
Including RMS Gateway Data in a Distributed Database Using Ingres Star	3-3
Destroying Gateway Databases	3-4

Chapter 4: Registering RMS Data to the Gateway

Defining RMS Data to the Gateway: A Summary	4-1
Invoking the Terminal Monitor	4-2
Registering Sharable Tables	4-2
Registering RMS Files	4-3
RMS Tuple Size versus RMS Record Size	4-6
Registering a File as Multiple Tables	4-7
Specifying RMS File Names	4-7
Registering Primary Keys	4-7
Specifying the Key Clause	4-8
Specifying the Row Count	4-9

Mapping Ingres and RMS Data Types	4-9
Removing Table Registrations	4-16
Examples of Register Table Statements	4-17
Registering RMS Indexes	4-18
Removing Index Registrations	4-20
Example of a Register Index Statement	4-20

Chapter 5: Accessing RMS Data Through the RMS Gateway-1

RMS File Access	5-1
Table Ownership and Access	5-2
Connecting to a Database	5-2
Issuing the Embedded SQL Connect Statement	5-3
Connecting to an RMS Gateway Database with an Ingres Component	5-3
Accessing RMS Data in a Distributed Database Using Ingres Star	5-4
Connecting to a Database on a Remote Node	5-4
Ingres Components Supported by the Gateway	5-5
Supported Ingres Statements	5-5
SQL Statements Not Supported by the RMS Gateway	5-6
SQL Statements That Modify RMS Files	5-6
Repeat Queries	5-7
Specifying Integrity Constraints on RMS Gateway Tables	5-7
Using the Ingres Optional Product Extension	5-8
Object Management Extension	5-8
RMS Gateway Transaction Handling	5-8
Locking RMS Data Files	5-8
Setting Timeout Values	5-9
Protecting Updates to RMS Files	5-10
Performance Tuning Your Applications	5-10

Chapter 6: Access to Repeating Group Data in the RMS Gateway

Repeating Group Access	6-1
Usage Guidelines	6-2
Registering a Table with Repeating Group	6-3
Read Data from Repeating Group	6-4
Characteristics of Repeating Group Access	6-4
Share Open Files and Multistream Record Access	6-5
RMS Descending Key	6-6
Read-Regardless-Lock Option	6-6
Read-Only Server Option	6-7
Buffering Record File Address (RFA) Option	6-8
RMS Gateway Database Commands	6-9

Limitations of Optimizedb	6-9
Using Register and Remove Commands with Stored Statistics	6-10
Data Types of Columns	6-10
When to Run the Utilities	6-11
Suitability to RMS Gateway Tables with Repeating Groups	6-11
Suggested Command Sequence	6-12

Chapter 7: Managing RMS Gateway Databases

Granting Privileges for RMS Gateway Databases	7-1
Using Ingres Database Commands and Utilities with RMS Gateway Databases	7-2
Backup and Recovery of RMS Gateway Databases	7-4
RMS Gateway Error Handling	7-5
Standard Ingres Errors	7-5
RMS Gateway User Errors	7-6
Debugging Complex Query Errors	7-8

Appendix A: Generic Errors

Generic Error Codes	A-1
---------------------------	-----

Appendix B: RMS Gateway Catalogs

RMS Gateway Catalogs	B-1
RMS Gateway Extended Catalogs	B-2
Catalog Entries for Registered RMS Files	B-3
Catalog Entries for Registered RMS Indexes	B-4

Appendix C: Installation and Configuration Notes

Installation Requirements	C-1
How the RMS Gateway Is Installed	C-1
Changing Log File Size	C-1
Configuring Logging and Locking System Values	C-2
Initializing the RMS Gateway	C-2
Configuring the RMS Gateway Data Manager	C-3
Specifying Server Parameter Values During Gateway Configuration	C-3
Configuration Differences for RMS Gateway Data Manager	C-4
Starting and Stopping the RMS Gateway	C-5

Index

Chapter 1: Introducing the RMS Gateway

The RMS gateway allows Ingres® users transparent access from various client applications to data stored in OpenVMS RMS files.

Ingres users on a client can develop database applications on an OpenVMS platform, which can be executed against RMS data without requiring additional programming.

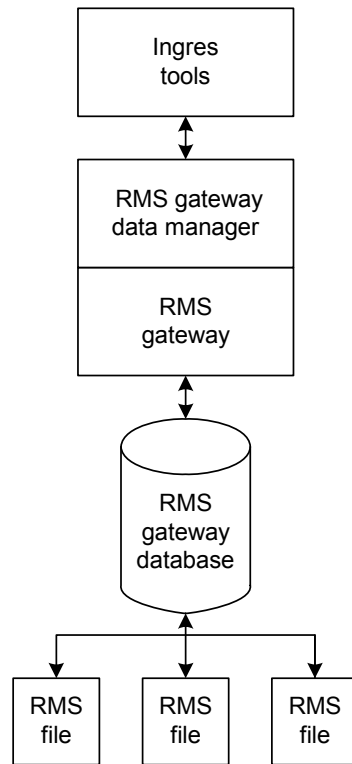
RMS Gateway

The RMS gateway lets you use the Ingres tools to query, update, and create data stored in RMS files. The gateway provides full read and write capabilities.

The RMS gateway is part of an RMS gateway data manager—similar to the standard Ingres data manager—that lets Ingres manipulate data in RMS files as relational data in Ingres tables. The RMS gateway performs all supported Ingres data manipulation functions, such as retrieving, updating or inserting data.

To use the RMS gateway, you must install an RMS gateway data manager, even if your site already has a standard Ingres data manager (that is, a data manager that lets you access data stored in Ingres databases).

The following figure illustrates the basic structure of the RMS gateway:



How the RMS Gateway Works

The RMS gateway lets you represent data records in RMS files as rows in Ingres tables, using the full relational capabilities of Ingres. The RMS gateway resides in the RMS gateway data manager process as part of the Generic Gateway Facility (GWF). The GWF, which is an extension of the Data Manipulation Facility (DMF) component of the data manager, allows physical access to non-Ingres data files.

To access RMS data through the gateway, you must define the data to Ingres by creating an RMS gateway database in which you register your RMS data as Ingres tables and indexes. This database has the extension `/rms` so that the data manager knows that it is an RMS gateway database.

An RMS gateway database does not contain any RMS data. It holds only the catalog entries for the underlying RMS data—that is, the RMS data files and indexes that you have registered in the database. These catalog entries tell the gateway:

- Where the underlying data is located
- What the data looks like
- How data is mapped to Ingres databases and tables

This guide describes how to register RMS data and access the data through the RMS gateway.

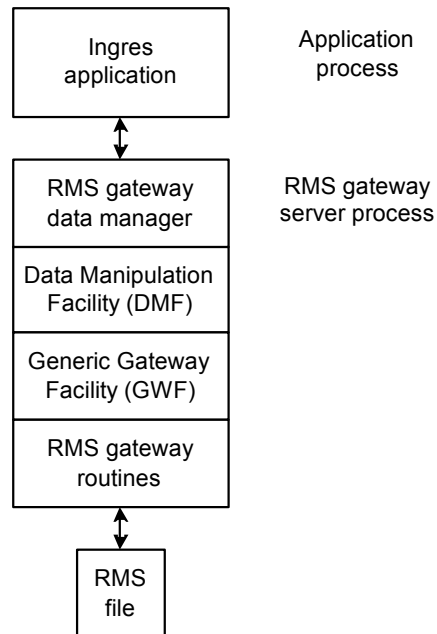
Gateway Query Processing

When you submit a query to the RMS gateway server process, most components of the data manager treat the query as though it were accessing Ingres data. However, the data manager knows that you are connected to a gateway database because of the `/rms` extension.

In this way, the data manager's Data Manipulation Facility (DMF) can make the proper calls to the gateway facility. For example, when a table in a gateway database is opened, the DMF *open* routine calls the gateway routine that uses the RMS function to open the file. Similarly, when data is requested, the DMF *read* routine calls the gateway routine to obtain the data from the RMS file.

After the gateway exit routine is completed successfully, the data is converted to Ingres data types and looks as though it comes from an Ingres table.

The following figure illustrates how the RMS gateway processes a query to an RMS file:



In This Guide

This guide describes features and special characteristics of the gateway, and provides information about how to:

- Define RMS data to the gateway
- Register RMS data files as Ingres tables in an RMS gateway database
- Access your RMS gateway data
- Maintain RMS gateway databases

Audience

This guide is designed for:

- The system administrator who installs and maintains the Ingres products that provide access to RMS
- The OpenVMS system administrator who supports the installation and operation of these Ingres products

The guide assumes that the reader is familiar with the basic operations of OpenVMS/RMS and Ingres products.

Terminology

In this guide, the following terminology is used:

- A *command* is an operation that you execute at the operating system level. An extended operation invoked by a command is often referred to as a *utility*.
- A *statement* is an operation that you embed within a program or database procedure, or execute interactively (for example, using the Terminal Monitor).

A statement can be written in Ingres/4GL, a host programming language (such as C), or a database query language (such as SQL or QUEL).

In this guide, the term *OpenSQL* is synonymous with *Ingres OpenSQL*.

Query Languages

The industry standard query language, SQL, ISO Entry SQL92, is used as the standard query language throughout this guide. For details about the settings required to operate in compliance with ISO Entry SQL92, see the online *OpenSQL Reference Guide*.

Chapter 2: Preparing to Use the RMS Gateway

This chapter helps you prepare to use the RMS gateway. The chapter includes:

- An overview of the RMS gateway and its environment
- A summary of the steps involved in accessing data through the RMS gateway
- A section about using OpenSQL to develop applications that are portable to other database management systems

You should read this chapter before you try to access data through the RMS gateway. The information it contains can help you use the RMS gateway more effectively.

RMS Gateway and Its Environment

The RMS gateway is part of the Ingres relational database management system that lets you access data across a wide range of hardware, software, and operating systems. This section gives an overview of Ingres and describes how the RMS gateway fits into an Ingres installation.

A full Ingres installation at an OpenVMS site can include various combinations of the following components:

- The Ingres data manager that coordinates requests between the Ingres tools and Ingres databases.
- Various Ingres components.
- The RMS gateway to access RMS files through an RMS gateway database; the gateway is part of the Ingres RMS gateway data manager, which is similar to the Ingres data manager.
- Ingres Net to access data when the Ingres tools and RMS gateway are on different nodes or computer systems.
- Ingres Star to allow simultaneous access to any combination of databases—Ingres, RMS gateway, and others.

The following sections describe these components and how they fit together.

Ingres Data Manager

The Ingres data manager is a process that coordinates all requests between you and your data in Ingres databases. When you submit a query through an Ingres user interface, the data manager interprets your request and interacts with the data accordingly. The data manager can communicate with multiple interfaces at the same time and coordinate each of their requests.

(The data manager is also known as the *server*. In this guide, the term ``server`` generally refers to a specific data manager process.)

Ingres Components

The Ingres components include user interfaces and tools that let you access data. Ingres components include:

- Applications developed with Ingres 4GL (Fourth Generation Language) and OpenROAD®
- Host-language programs that include embedded SQL (Structured Query Language) commands
- Forms-based screens
- SQL commands submitted interactively

There are several types of Ingres user interfaces. Forms-based screens let you use tools—such as Query-By-Forms and Visual-Forms-Editor—that are highly structured and easy to use. However, these tools generally have a limited range of operations.

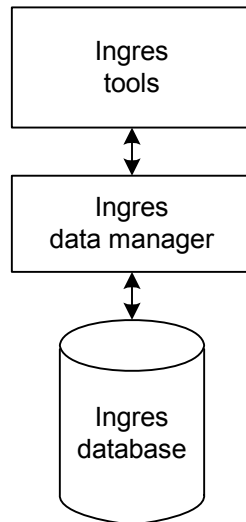
Other Ingres user interfaces such as SQL are more flexible and allow you a wider range of operations. However, these interfaces require that you learn specific tools in greater detail to make full use of their capabilities.

The following user interfaces are available with the RMS gateway:

- Application development tools, such as Applications-By-Forms and Ingres 4GL, Ingres Vision and OpenROAD
- Forms-based user tools, such as Query-By-Forms, Report-By-Forms, and Visual-Forms-Editor
- Ingres 3GL (Third Generation Language) preprocessors for including embedded SQL statements in host language programs
- Ingres Terminal Monitor for submitting SQL commands interactively

For more information about using these components, see Ingres Components Supported by the Gateway in the chapter “Accessing RMS Data Through the RMS Gateway.”

The following figure shows the structure of a basic Ingres installation that includes the Ingres tools, the Ingres data manager, and the Ingres database:



Ingres Net

You can access data in either of two modes in an Ingres installation:

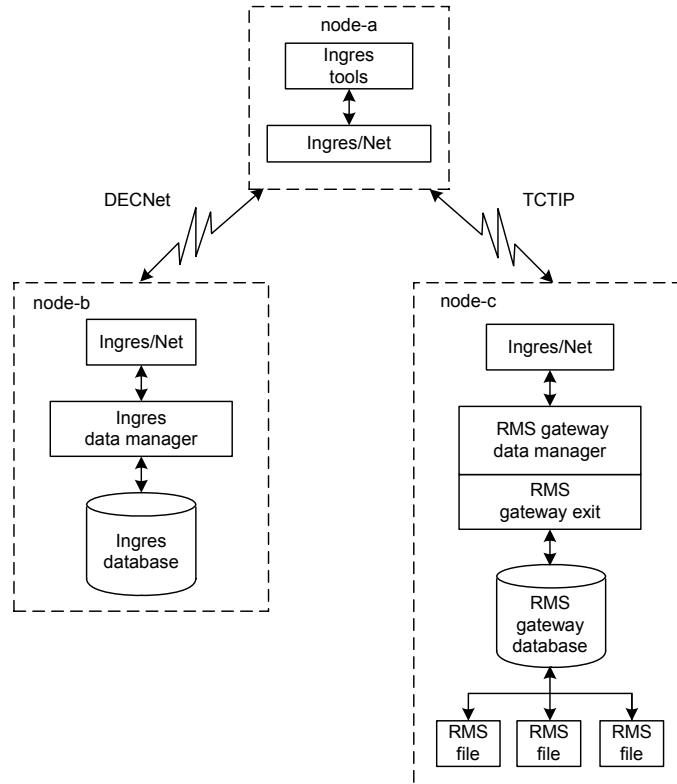
- In *local* mode, all components reside on your local computer system
- In *network* mode, the various components reside on different computers linked to others through a communications protocol, such as DECnet

In network mode, each computer becomes a network *node*. Your computer is the *local* node; all others are considered *remote* nodes.

You must install Ingres Net to use the Ingres components in network mode. With Ingres Net and the RMS gateway on your system, applications on remote nodes—in another building, another city, even another country—can access local RMS files.

Ingres Net is not itself a network but an interface to your existing network setup and protocols. In a network with Ingres Net on each node, you can access remote data through Ingres user interfaces on your local node and a data manager on a remote node. The remote data appears as though it were on your local node; remote data access is handled completely by Ingres Net.

The following figure shows an example of an Ingres installation that includes the RMS gateway and Ingres Net. See the *System Administrator Guide* for more information about using Ingres Net.



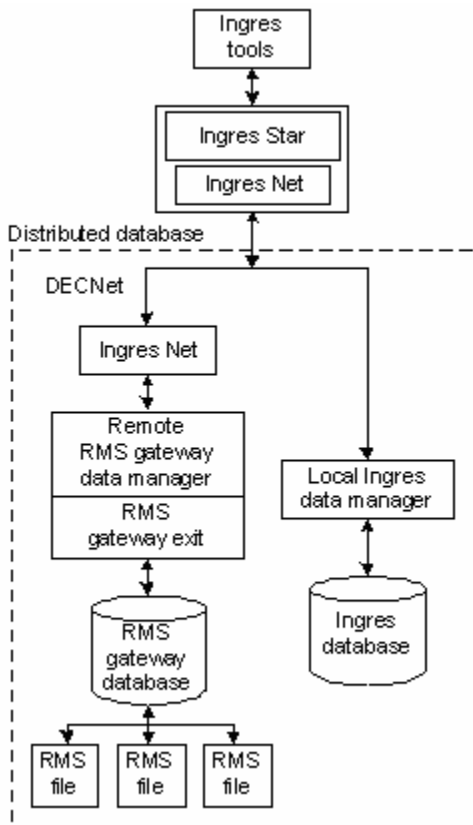
Ingres Star

In the configuration described above, you still can access only one database at a time. However, when you add Ingres Star to your Ingres installation, you can access multiple databases simultaneously. You do this by creating a *distributed database* containing data from any combination of these sources:

- Ingres databases
- RMS gateway databases
- Data from other database management systems that are supported by an Ingres gateway (for example, Oracle or Rdb)

You then can use an Ingres tool to access your distributed database as you would any other database. The various types of data appear to you and to Ingres as residing in a single database.

The following figure illustrates an Ingres installation that contains a distributed database containing Ingres and RMS data. For more information about Ingres Star and distributed databases, see the *Ingres Star User Guide*.



Accessing Data Through the RMS Gateway: A Summary

To access RMS data through the RMS gateway, you first must define the data to Ingres. This process includes:

- Creating an RMS gateway database to hold your data definitions

To create a gateway database, use the `createdb` command. Include the extension `/rms` after the database name. The chapter "Creating RMS Gateway Databases" discusses how to create and destroy RMS gateway databases.

An RMS gateway database stores the Ingres definitions of your RMS data files rather than the data itself. You can modify these definitions—by registering and dropping tables and indexes, or even destroying the RMS gateway database—without affecting the underlying data, which always remains in the RMS files.

- Registering your RMS data files and indexes as Ingres tables and indexes in the RMS gateway database

You do this by issuing the `register table` and `register index` statements from your application. These statements create catalog entries in the gateway database to describe the RMS data as Ingres tables and indexes.

- Converting RMS data types to Ingres data types

When you register your RMS files as Ingres tables, you must specify your RMS data in terms of the standard data types that Ingres uses. You can designate the default data type conversions or specific conversions for any columns.

The chapter “Registering RMS Data to the Gateway” discusses how to register RMS files and indexes and how to convert data types.

After registering your RMS data and indexes, call the Ingres user tools to access the tables in the RMS gateway database. You must include the `/rms` database extension, which informs the Ingres RMS gateway data manager to use the underlying RMS files that you have registered.

The chapter “Accessing RMS Data Through the RMS Gateway” discusses how to access RMS data through the RMS gateway.

Using OpenSQL For Portability

The RMS gateway requires that you use Structured Query Language (SQL) for database queries. In Ingres, there are two versions of SQL:

- Full SQL contains all data definition and manipulation statements understood by the data manager.
- OpenSQL, a subset of full SQL, contains only those statements and data types that you can use with an Ingres gateway to a relational database management system (RDBMS).

OpenSQL lets you develop applications that are portable to any RDBMS—for example, Oracle or Rdb.

Because the RMS gateway uses the Ingres RMS gateway data manager to process queries, it can accept the full set of SQL statements. If you intend to use your applications with an Ingres gateway to any RDBMS, however, you should limit them to OpenSQL. See the *OpenSQL Reference Guide* to learn which statements and data types are not available in OpenSQL.

Chapter 3: Creating RMS Gateway Databases

The first step in accessing RMS data through the RMS gateway is to create an RMS gateway database to hold the Ingres definitions of your RMS data files and indexes. This database does not store any RMS data. Rather, it holds entries in the Ingres catalogs that tell the gateway where and how to locate your RMS data.

This chapter describes RMS gateway databases and how to:

- Create an RMS gateway database.
- Include RMS data in a distributed database using Ingres Star.
- Destroy a gateway database.

What Is an RMS Gateway Database?

Ingres always requires that you be connected to the database that contains the data you want to access. Because RMS stores data in files rather than databases, you must create a database through which the RMS gateway can access the RMS data.

A database is simply a collection of related data. An RMS gateway database can correspond to a number of RMS data files. For example, you can create a gateway database called "personnel" in which you register several RMS files containing employee information. See the Registering RMS Files section in the chapter "Registering RMS Data to the Gateway" for more information on how to register RMS files.

An RMS gateway database stores the catalogs that hold Ingres definitions of the RMS files and indexes. These catalogs tell the RMS gateway data manager where the data is located. When you use an Ingres tool, the data manager refers to these catalog entries to access the appropriate RMS data.

The RMS gateway-specific catalogs are described in the appendix "RMS Gateway Catalogs." For a description of the Ingres system catalogs, see the *Database Administrator Guide*.

Because an RMS gateway database does not contain any of the actual RMS data, you can create or destroy gateway databases without affecting the RMS files registered in them.

Creating RMS Gateway Databases

To create an RMS gateway database, enter the following command at the OpenVMS operating-system prompt:

```
$ createdb dbname/rms
```

where *dbname* is the name you want to give your RMS gateway database. This name must follow the standard Ingres naming conventions described in the Naming RMS Gateway Databases and Database Objects section.

Note: If you are unable to issue the `createdb` statement, consult your Ingres system administrator to make sure you have the necessary Ingres privileges to create databases.

For example, to create an RMS gateway database to hold the definitions of RMS files with personnel data, enter the following at the OpenVMS prompt:

```
$ createdb personnel/rms
```

You must include the extension `/rms` after the database name. This tells Ingres that you are creating an RMS gateway database. You use this extension as part of the database name when you register or access the RMS data.

When you create a database, you become the database administrator (DBA) for that database. You then can grant privileges to other users as discussed in the Granting Privileges for RMS Gateway Databases section in the chapter "Managing RMS Gateway Databases."

Naming RMS Gateway Databases and Database Objects

Observe the following rules when you choose a name for an RMS gateway object (such as a database, table, or column):

- The name can be no more than 32 characters in length (not including the `/rms` extension).
- The first character of the name must be alphabetic (a-z) or an underscore (`_`).
- You also can use the characters 0-9, `#`, `@`, or `$` in the name after the first character.
- You must give a unique name to each database in an Ingres installation, regardless of the database extension.

For example, if you have an Ingres database called "inventory," you cannot create an RMS gateway database called `inventory/rms`.

Unless you have the SQL 92 option turned on:

- Do not use uppercase and lowercase letters to differentiate between names because the RMS gateway converts any uppercase letters to lowercase.

For example, the RMS gateway considers “personnel” and “Personnel” to be the same name.

- You cannot use the RMS gateway reserved words as names.

See the *SQL Reference Guide* for a list of reserved words.

The names of the RMS gateway tables and indexes in your gateway databases must also follow these rules when you register your RMS files and indexes as described in the chapter “Registering RMS Data to the Gateway.”

Including RMS Gateway Data in a Distributed Database Using Ingres Star

If you have Ingres Star available on your system, you can include an RMS gateway database in a *distributed database*. You then can access the RMS gateway database at the same time as Ingres databases and databases that you access through other Ingres gateways, such as the Rdb gateway. Ingres Star lets you view and manipulate data from various sources as part of a single database.

To access RMS data as part of a distributed database:

1. Create an RMS gateway database as described previously.
2. Create a distributed database through Ingres Star.

Give this database the extension */d*. You must include this extension when you want to access the distributed database.

3. In the Ingres Star distributed database, register the following:
 - Tables in the RMS gateway that you want to access through Ingres Star
 - Any other databases you want to include in the distributed database

For details on creating and accessing distributed databases, see the *Ingres Star User Guide*.

Destroying Gateway Databases

Destroying an RMS gateway database destroys all the tables in the gateway database to which RMS files are registered. This process has no effect on the RMS data files themselves.

To destroy a gateway database, enter the following command at the OpenVMS operating system prompt:

```
$ destroydb dbname
```

Chapter 4: Registering RMS Data to the Gateway

This chapter describes how to register RMS data files as Ingres tables in an RMS gateway database. You can then access the RMS data with the Ingres user interfaces simply by referring to the Ingres tables in which the data is registered.

The tables in which you register your files do not contain any RMS data. Rather, they store Ingres catalog entries that tell the gateway where the RMS data is located and what it looks like.

If you have a secondary index on an RMS file, you can also register that index as described in this chapter. Registering your RMS indexes improves the performance of the gateway.

This chapter also discusses how to remove the registration of tables and indexes from a gateway database. You can change the definitions in your gateway databases however you desire—by registering or removing tables and indexes—without affecting the underlying RMS data files.

If you are not the database administrator—that is, the user who created the RMS gateway database in which you want to register a table—see the Registering Sharable Tables section before you register any RMS data.

Defining RMS Data to the Gateway: A Summary

You specify the definitions of your RMS files by issuing the SQL register table statement. This statement creates entries in Ingres tables that describe your RMS data. Using the register index statement, you can also register a secondary index that exists for an RMS file.

The register table and register index statements are discussed in detail in this chapter.

To register RMS files and indexes:

1. Invoke the Ingres Interactive SQL Terminal Monitor for the RMS gateway database in which you want the tables to reside.

The Terminal Monitor is the interactive Ingres tool that lets you enter SQL statements directly on the screen. The Invoking the Terminal Monitor section describes how to invoke the Terminal Monitor.

2. Register the file with the register table statement.

The register table statement updates the Ingres system catalogs with information describing your RMS files. See the appendix “RMS Gateway Catalogs” for descriptions of these catalog entries.

The RMS file need not exist until you are ready to access it. This is because the catalog entries contain only data definitions, not the underlying data itself, which always remains in the RMS files.

3. If the RMS file has a secondary index, issue the register index statement to describe the index. The gateway makes the appropriate entries in the Ingres catalogs, as described in the appendix “RMS Gateway Catalogs.”

Issue a commit statement after each register table and register index statement to free system catalogs and avoid unnecessary deadlock situations.

Invoking the Terminal Monitor

You issue the SQL register table and register index statements on your terminal screen in the Ingres Interactive SQL Terminal Monitor. To invoke the Terminal Monitor, issue the following command at the OpenVMS operating system prompt:

```
$ isql dbname/rms
```

where *dbname* is the name of the RMS gateway database in which you want to register the RMS file.

You must include the */rms* extension after the database name so that the Ingres data manager knows that you want to connect to an RMS gateway database.

You invoke the Terminal Monitor in the same way to remove the registration of a table or index, as described later in this chapter. If you are not familiar with using the Interactive Terminal Monitor, see the discussion of Interactive SQL in the *Character-based Querying and Reporting Tools User Guide*.

Registering Sharable Tables

When you register an RMS file as an Ingres table, Ingres recognizes you as the owner of the table. Only tables owned by the database administrator (DBA) for a database—that is, the user who creates the database—can be shared with other users.

If you are not the DBA and you want to register tables that other users can access, invoke the Terminal Monitor as follows:

```
$ isql -udba dbname/rms
```


where:

- The `-u` flag tells Ingres that you want to assume the identity of another user. (The `-u` flag is in effect only for the duration of your Terminal Monitor session.)

Note: To use the `-u` command flag, you must have been given Ingres superuser privileges by your Ingres system administrator or other Ingres superuser.

- *dba* represents the Ingres username of the DBA

When you call the Terminal Monitor in this way, Ingres treats the tables that you register as though the DBA registered them. The DBA then can grant other users access privileges to the tables (see the Granting Privileges for RMS Gateway Databases section in the chapter "Managing RMS Gateway Databases").

An alternative to the above procedure is to have the DBA for the database register your tables for you.

Registering RMS Files

After you invoke the Terminal Monitor, you can issue the SQL register table statement. This statement maps an RMS data file to an Ingres table name.

When you register an RMS file, you describe it as an Ingres table by providing the following information:

- A name for the table
- A name for each column in the table
These columns correspond to the fields of the RMS file record.
- The Ingres data type for each column in the table
- The RMS file name and directory path
The gateway uses structure to assign a storage structure type to the Ingres table.
- Any primary key fields in the RMS file
- The number of rows in the table
- Whether the table can accept duplicate rows
- Whether users can update rows in the table through the gateway

The full syntax of the register table statement is:

```
register table tablename
(col_name format [is 'ext_format']
{,col_name format [is 'ext_format']})
as import
from 'source'
with
dbms = rms
[,structure = none|[unique]sortkeyed]
[,key = (col_name [asc] {,col_name })]
[,rows = nnnnnnnn]
[,[no]duplicates]
[,[no]update]
;
```

See the Examples of Register Table Statements section in this chapter for examples of table registrations.

The following table describes each of the parameters included in the register table statement. You are referred to the appropriate sections in this chapter for more detail about some parameters:

Parameter	Description	Required or Optional
tablename	The Ingres tablename that you give to the RMS file. This name must follow the standard Ingres naming conventions listed in the chapter "Creating RMS Gateway Databases."	Required
col_name	The name that you choose for a column of the Ingres table. This name must follow the standard Ingres naming conventions listed in the chapter "Creating RMS Gateway Databases." Each column name corresponds to a field in the RMS file. Specify table columns in the order of the corresponding fields in the RMS file. To indicate where you have omitted fields in the RMS file, see the Specifying Extended Formats section.	Required

Parameter	Description	Required or Optional
format	The Ingres data type for the column. See the Mapping Ingres and RMS Data Types section for more information about specifying data types.	Required
is 'ext_format'	Specifies either: <ul style="list-style-type: none">■ A non-default data type mappingor■ An offset for an omitted or non-contiguous field You must enclose the value of <i>ext_format</i> in single quotes. See the Specifying Extended Formats section for more information.	Optional
as import	Indicates that you are registering an RMS gateway table.	Required
from 'source'	The OpenVMS file specification for the RMS file that you are registering. You must enclose the value of <i>source</i> in single quotes. See the Specifying RMS File Names section for more information.	Required
with	Introduces additional information about the file being imported.	Required
dbms = rms	Indicates that you are registering an RMS file.	Required
structure =	Indicates the primary key for the RMS file. See the Registering Primary Keys section for more information. If you do not include the structure clause, Ingres assumes that the RMS file is not keyed.	Optional
key =	Specifies the column name(s) to use as keys. See the Registering Primary Keys section for more information.	Required if you specify sortkeyed as the key type

Parameter	Description	Required or Optional
rows = nnnnnnnn	Specifies the approximate number of rows in the table; if you do not indicate a value, the default is 1,000. See the Specifying the Row Count section for more information.	Optional
[no]duplicates	Indicates whether the table can contain duplicate rows; if you do not include this clause, the default is to allow duplicates.	Optional
[no]update	Specifies whether the table can be updated; if you do not include this clause, the gateway assumes the default of noupdate (read-only).	Optional

It is important that you define your RMS files properly so that you can retrieve data correctly through the RMS gateway because Ingres cannot validate your data definitions. The RMS gateway detects errors in registration at runtime. See the RMS Gateway Error Handling section in the chapter “Managing RMS Gateway Databases” for more information about registration errors.

The following sections provide rules and guidelines to help you register your RMS files correctly.

RMS Tuple Size versus RMS Record Size

In earlier versions of RMS gateway, a table could have a tuple size (that is, row size, record size) of up to 2008 bytes. The maximum record size for RMS files is 32 KB.

The maximum tuple size of RMS gateway has been increased to 32660 bytes. The default tuple size remains at 2008 bytes. Use Configuration-By-Forms (CBF) to increase the values for MAX_TUPLE_SIZE, DEFAULT_PAGE_SIZE, and CACHE configuration as needed. If the parameter MAX_TUPLE_LENGTH is set to 32660 bytes, the parameter DEFAULT_PAGE_SIZE should be set to 32768 (32 KB) only if your record size will always be 32 KB—with the parameter DMF Cache 32K under CACHE configuration set to ON.

Your environment usually consists of varying tuple sizes and page sizes. The default page size parameter (DEFAULT_PAGE_SIZE) should be set to the most common page size value allowed in your configuration. The configuration parameters specified under the CACHE option in the RMS gateway configuration menu determine the allowable page sizes. DMF Cache 2K must always be set to ON. To use 4K, 8K, 16K, or 32K tuple sizes and page sizes, the appropriate CACHE value must be set to ON.

For a detailed explanation of these parameters, start CBF, select the parameter and press the Help key.

Registering a File as Multiple Tables

Use at least one register table command for each RMS file. You can register a file as more than one table—for example, if you need to access only a portion of an RMS file.

If you divide an RMS file into more than one table, you must use an extended format (see the Specifying Extended Formats section) in the register table statement for the second and any subsequent tables. The extended format indicates the column offset—that is, the position of the field in the RMS file that corresponds to the first column of the table. Because RMS gateway does data conversion on every column, you can improve performance by escaping data in a large RMS record using extended format and registering multiple tables on a file.

Specifying RMS File Names

For each RMS file that you register:

- Specify a full directory path and file name in the *from source* clause of the register table statement.
- Enclose in single quotes the entire string that you specify as *source*.
- Define any logicals in the pathname as follows so that the gateway can translate them at runtime:
 - For production installations, define logicals at the SYSTEM level.
 - For test installations, define logicals at the SYSTEM or GROUP level.

The RMS gateway server is a detached OpenVMS process; therefore, it cannot recognize any logicals defined at the JOB or PROCESS level.

- You can include DECnet node names in RMS file specifications; however, this can result in file protection violations on remote RMS files because the username of the RMS gateway server process is used in DECnet proxy logins.

Registering Primary Keys

To register a primary key for an RMS file, you should include in the register table statement:

- A structure clause with a value of sortkeyed
- A key clause to describe the key

You cannot include a key clause if you specify none in the structure clause.

The structure clause of the register table statement accepts the following values:

Key Type	Description
[unique]sortkeyed	<p>The table is keyed; the RMS gateway returns records in sort order. The gateway treats the registered table as having a B-tree structure.</p> <p>If you specify sortkeyed, you can include the optional unique qualifier to indicate that each row of the table must have a unique key value.</p>
none	<p>The table is not keyed; the RMS gateway:</p> <ul style="list-style-type: none">■ Uses a full scan to locate records■ Inserts new records at the end of the file <p>The gateway treats the registered table as having a heap structure even if the RMS file is keyed.</p>

Specifying the Key Clause

If you specify in the structure clause that a key exists for the RMS file, use the key clause of the register table statement to describe the key. The key clause should contain a column name or list of column names corresponding to the field or fields that make up the primary key for the file. You must specify the columns in the same order as the primary key in the RMS file.

The gateway only supports keys whose columns all are ascending. Do not attempt to register keys with descending columns.

If you specify a list of columns in the key clause, be sure these columns are of character types (char, varchar, c, or text). Non-character typed columns in the key clause may result in incorrect responses because the RMS gateway takes the entire multi-column key as one type, STRING, and searches and collates accordingly.

Important! Because there is no way to predict whether a query will result in errors in returned values when the key clause contains non-character type columns in a multi-column key, be sure that a multi-column key contains only character types.

Specifying the Row Count

You can use the rows clause of the register table statement to specify as closely as possible the number of rows in the table. When you submit a query that accesses the table, the data manager uses this value in planning the most efficient way to execute the query.

If you do not specify a value, the RMS gateway uses a default value of 1000 rows.

Mapping Ingres and RMS Data Types

The RMS gateway is responsible for converting the data types of RMS data that pass through the gateway. Conversion occurs:

- From an OpenVMS data type to an Ingres data type when you retrieve records from an RMS file
- From the Ingres data type back to the OpenVMS data type when you insert or update records in the RMS file

When you register an RMS file as an Ingres table, you represent the OpenVMS data type for each field in the file as an Ingres data type for the corresponding column in the table. You can designate the Ingres data type for a column in either of two ways:

- Specify the *default* Ingres data type to which the RMS gateway maps the OpenVMS data type for the field

The default data type is the Ingres data type that most closely matches each OpenVMS data type. When you specify the default Ingres data type, you improve the performance of the gateway because little or no conversion is involved.

- Specify an Ingres data type other than the default mapping

When the default data type does not perform the conversion you want, you can indicate a different mapping—to convert an integer field as a float column, for example. Also, no default mapping is available for some OpenVMS data types.

You must specify an Ingres data type for each column that you include in a register table statement. To specify a non-default mapping, use an extended format as described in the Specifying Extended Formats section. However, never use extended format on a column that is part of a key.

The table below indicates for each OpenVMS data type:

- Default Ingres data type mapping
- Additional Ingres data type conversions that are available through an extended format specification

- Conversion issues

In deciding on a data type conversion, keep in mind that:

- Some conversions provide better performance than others—for example, conversion between integer and string types is more efficient than conversion from integer to float.
- Some conversions can result in a loss of precision or range—for example, converting from an OpenVMS `f_float(4)` data type to an Ingres `integer4` data type.

For descriptions of the Ingres data types, see the *SQL Reference Guide*. For more information about OpenVMS data types, see the OpenVMS documentation.

OpenVMS Data Type	Default Ingres Data Type	Additional Ingres Data Types Available	Conversion Issues
signed byte	integer1	integer2, integer4 float4, float8 money, decimal c, text, char, varchar	None
unsigned byte signed word	integer2	integer1	Possible errors in range values
		integer4 float4, float8 money, decimal c, text, char, varchar	None
unsigned word	integer4	integer1, integer2	Possible errors in range values
		float4, float8 money, decimal c, text, char, varchar	None
signed longword	integer4	integer1, integer2	Possible errors in range values
		float4	Possible loss of precision
		float8 money, decimal c, text, char, varchar	None
unsigned longword	float8	integer1, integer2, integer4	Possible errors in range values
		float4	Possible loss of precision

OpenVMS Data Type	Default Ingres Data Type	Additional Ingres Data Types Available	Conversion Issues
		money, decimal c, text, char, varchar	None
signed quadword unsigned quadword signed octaword unsigned octaword	float8 (possible loss of precision)	integer1, integer2, integer4	Possible errors in range values
		float4, money, decimal	Possible loss of precision
		c, text, char, varchar	None
f_floating s_floating (IEEE)	float4	integer1, integer2, integer4 money, decimal	Possible loss of precision
		float8 c, text, char, varchar	None
d_floating t_floating (IEEE)	float8	integer1, integer2, integer4 float4 money, decimal	Possible loss of precision
		c, text, char, varchar	None
g_floating h_floating	float8	integer1, integer2, integer4 float4 money, decimal	Possible errors in range values
		c, text, char, varchar	None
OpenVMS date	date	c, text, char, varchar	Must be an absolute OpenVMS date
blankpad fixed string	char	date	Must be a valid Ingres ASCII date
		c, text, char, varchar	None
		integer1, integer2, integer4 float4, float8 money, decimal	Possible syntax or range errors
blankpad varying string	varchar	date, decimal	Must be a valid Ingres ASCII date
		c, text, char, varchar	None
packed decimal	decimal	int1, iwt2 int4, f4, f8 c, text, char varchar, money	Possible loss of precision

OpenVMS Data Type	Default Ingres Data Type	Additional Ingres Data Types Available	Conversion Issues
packed decimal unsigned numeric leading numeric zoned numeric overpunched numeric trailing separate numeric (not a native OpenVMS data type; see your OpenVMS COBOL documentation for a definition)	none	integer1, integer2, integer4 float4, float8 money, decimal	Possible loss of precision
		c, text, char, varchar	None

Additional Notes About Data Type Conversions

Be aware of the following general restrictions and considerations when you convert OpenVMS data types to Ingres data types:

- The c, text, float4, integer1, and money Ingres data types are not valid in OpenSQL. Do not convert to these data types if you intend to use your applications with a gateway to a relational database management system (such as Oracle Rdb).
- You cannot declare columns in gateway tables as WITH NULL or WITH DEFAULT because RMS does not support these concepts. All columns are implicitly created as NOT NULL NOT DEFAULT.
- When you access an OpenVMS date field through the RMS gateway and through RMS simultaneously, you may get different results if the Ingres user interface or application is located in a time zone other than the one where the OpenVMS date was originally stored.

This is due to the Ingres Time Zone Differential that displays dates based on the difference between local time and Greenwich Mean Time.
- OpenVMS dates always include a time stamp, but Ingres dates sometimes do and sometimes do not. When you retrieve from an OpenVMS date, you get a time stamp. If you did not include a time stamp in your insert, you get the default time stamp, 00:00:00.

The following table describes issues related to specific conversions of OpenVMS data types to Ingres data types:

When Converting From...	To...	Be Aware:
An RMS string or numeric type	An Ingres character data type (c, text, char, varchar)	<p>The gateway may truncate data if the length of the RMS field is longer than the Ingres data type to which you convert.</p> <p>For example, if you convert a 20-character fixed string to a varchar(18), the gateway truncates the last two characters.</p> <p>Similarly, if you convert an RMS byte field to an Ingres varchar(2) column, a value of 127 in an RMS record is truncated to 12 in the Ingres column.</p>
An OpenVMS date	An Ingres character data type	<p>The gateway ignores the II_DATE_FORMAT setting.</p> <p>You cannot convert the formatted character string to an Ingres date without truncating the milliseconds portion.</p> <p>In general, you should convert OpenVMS dates to Ingres dates if you intend to use them in calculations.</p>
A numeric OpenVMS data type	An Ingres date	The value must be in the form of a valid Ingres numeric date specification; this is MMDDYY or DDMMYY, depending on the value of the II_DATE_FORMAT logical.
An RMS integer	An Ingres date	When adding or updating a record, the date must be an absolute date and the year must be between 1900 and 1999 inclusive.
RMS string types	An Ingres integer data type	<p>The string must be in integer format.</p> <p>The string could not contain a value of 1.234e5, for example.</p>
RMS string types	c or text data type	<p>The gateway changes non-printable characters (for c data type) or NULLs (for text data type) to blanks.</p> <p>This is an important consideration when updating an RMS record that contains unprintable characters or NULLs that you have converted to c or text. The gateway does not display a warning that it is changing the data in this way.</p>

When Converting From...	To...	Be Aware:
G_FLOATING or H_FLOATING	An Ingres character data type	<p>Ingres floating point formatting options have no effect.</p> <p>The gateway always returns these numbers using the "e" format with the maximum width (17 for G_FLOATING; 35 for H_FLOATING).</p> <p>Use this conversion method for G_FLOATING and H_FLOATING numbers that are too large to express as Ingres float8 columns. See the <i>SQL Reference Guide</i> for more details about Ingres data type formats.</p>
RMS string types	Ingres money or floating point data types	<p>The RMS field cannot be longer than 2000 characters, unless you use an extended format to specify an offset.</p> <p>See the Specifying Extended Formats section for more information.</p>

Specifying Extended Formats

Extended formats let you indicate additional information about an RMS field in a register table statement. Use an extended format to specify:

- An Ingres data type other than the default mapping, for example, to map an RMS integer field to an Ingres column with float data type
- An offset for a field; you must specify an offset in either of the following cases:
 - The field is not contiguous with the previous field
 - The first column of the table does not correspond to position 0 in the RMS file
- The position of an implied decimal point

When you include an extended format in the register table statement, use the following syntax:

```
col_name format is 'VMS_datatype (offset [,len [,scale]])'
```

For additional information, see the Examples of Register Table Statements section in this chapter. The following table describes the parameters of the extended format specification within the statement:

Parameter	Description
-----------	-------------

Parameter	Description
col_name	The name of the column in the Ingres table that corresponds to the RMS record field with the extended format.
format	The Ingres data type to which you are mapping the OpenVMS data type.
VMS_datatype	The OpenVMS data type of the field in the RMS file record. See the OpenVMS Data Type to Ingres Data Type Conversion table for permissible OpenVMS data types.
offset	Specifies the position (beginning with position 0) of the field within the RMS record.
length	Specifies the length, in bytes, of the field in the RMS file. Exceptions are the OpenVMS decimal data types, in which case <i>length</i> specifies the number of decimal digits in the packed decimal number. These data types include: <ul style="list-style-type: none"> ■ Additional Ingres data type conversions that are available through an extended format specification packed decimal ■ Additional Ingres data type conversions that are available through an extended format specification leading numeric ■ Additional Ingres data type conversions that are available through an extended format specification zoned numeric ■ Additional Ingres data type conversions that are available through an extended format specification overpunched numeric ■ Additional Ingres data type conversions that are available through an extended format specification unsigned numeric
scale	Specifies the position of the implied decimal point.

You must include the *VMS_datatype* and *offset* values for each extended format specification. You should use the *length* and *scale* parameters according to the following guidelines:

- Include a *length* specification for OpenVMS string data types, even if the length is the same as that of the Ingres data type to which you are converting

The OpenVMS string data types include blank or nullpad fixed or varying string.

- Do not specify *length* for OpenVMS data types that have a fixed length; these include:
 - byte, word, longword, quadword, octaword (signed or unsigned)
 - d_floating, f_floating, g_floating, h_floating
 - OpenVMS date
- Include values for both *length* and *scale* for the OpenVMS decimal and numeric string data types; these include:
 - packed decimal
 - unsigned numeric, leading numeric, zoned numeric, overpunched numeric

See the Examples of Register Table Statements section for sample extended format specifications.

Removing Table Registrations

Use the SQL remove table statement to remove the definition of an RMS file from an RMS gateway database. The remove table statement removes all Ingres catalog entries for a registered table, including any related views, integrities, and permits. The underlying RMS file is not affected.

The remove table statement has the syntax:

remove table *tablename*;

where *tablename* is the Ingres table name for a registered RMS file.

If you remove a table for which you have registered an RMS index, the registration for the index is also removed.

You should issue a commit statement immediately after a remove table statement to free system catalog locks and avoid unnecessary deadlocks.

Examples of Register Table Statements

The following register table statement registers the RMS file rmsf.t1 as the Ingres table ingtbl1. This example:

- Uses the default data type conversions
See the Mapping Ingres and RMS Data Types section for the possible data types of the RMS file fields.
- Creates columns in the Ingres table that correspond to contiguous fields in the RMS file
- Registers the RMS file without a primary key
- Assumes the defaults of noduplicates and nouupdates and 1000 rows because these clauses are not explicitly included

```
register table ingtbl1
(c1 integer,
 c2 integer2,
 c3 integer4,
 c4 float,
 c5 float8,
 c6 char(5),
 c7 varchar(15),
 c8 date
)
as import
from 'TE_USER:[USER1]rmsf.t1'
with
dbms=rms;
```

The following register table statement registers the RMS file rmsf.t2 as the Ingres table ingtbl2. In this example, an extended format is specified for each field:

```
register table ingtbl2
(c1 i4 is 'byte(0)',
 c2 i4 is 'byte(1)',
 c3 i4 is 'signed byte(2)',
 c4 i4 is 'unsigned byte(3)',
 c5 i4 is 'word(4)',
 c6 i4 is 'signed word(6)',
 c7 i4 is 'unsigned word (8)',
 c8 i4 is 'longword(10)',
 c9 i4 is 'signed longword(14)',
 c10 i4 is 'unsigned longword(18)'
 c11 f8 is 'quadword(22)',
 c12 f8 is 'signed quadword(30)',
 c13 f8 is 'unsigned quadword(38)',
 c14 f8 is 'octaword(46)',
 c15 f8 is 'signed octaword(62)',
 c16 f8 is 'unsigned octaword(80)',
 c17 money is 'packed decimal(96,10,2)',
 c18 char(20) is 'fixed string(101,20)',
 c19 char(20) is 'blankpad fixed string(121,20)',
 c20 varchar(20) is 'blankpad varying string(141,20)
)
as import
from 'TE_USER:[USER1]rmsf.t2'
with
dbms=rms;
```

Registering RMS Indexes

If an RMS file that you register has a secondary index, you can register the index as well as the file itself. You must register the file before you can register its index.

To register an index, use the register index statement. This statement does not create an Ingres or an RMS index; it simply informs the data manager that an RMS index exists, so that the gateway can use the index for faster query processing.

The register index statement has the syntax:

```
register [unique] index indexname on tablename  
(columnname  
{,columnname})  
as import  
from 'source'  
with structure = sortkeyed;
```

The following table describes the parameters of the register index statement. See the chapter “Introduction and Installation Overview” for a description of the conventions used to specify this syntax.

You can only use the register index statement for secondary indexes. Use the register table statement to specify the primary key, as described in the Registering Primary Keys section.

Parameter	Description	Required or Optional
unique	Indicates that there are no duplicate index records. Be sure that the index actually is unique before specifying unique, or queries may be processed unpredictably.	Optional
indexname	An Ingres indexname for the RMS index. This name must follow the naming conventions listed in the Naming Ingres Databases and Database Objects section in the chapter “Creating RMS Gateway Databases.”	Required
tablename	The name of the registered Ingres table that corresponds to the RMS file on which the index is defined.	Required

Parameter	Description	Required or Optional
columnname	<p>A column in the registered Ingres table; each column corresponds to a field of the RMS index.</p> <p>You must register columns in the same order in which they occur in the RMS secondary index.</p> <p>For a multi-column key, all columns must be of character types.</p> <p>Note: Non-character type columns in a multi-column key may result in errors in returned values.</p> <p>Because of RMS limitations, your index can be no more than:</p> <ul style="list-style-type: none"> ■ 255 KB in total length ■ Eight columns if OpenVMS string data types ■ One column if other OpenVMS data types 	Required
as import	Indicates that you are specifying an RMS gateway index.	Required
from 'source'	<p>The RMS key reference.</p> <p>This should be an integer between 1 and 254, entered as a quoted string.</p> <p>This value corresponds to the value that RMS stores in the RAB\$B_KRF field and to the FDL attribute CONNECT_KEY_OF_REFERENCE.</p>	Required
with structure =	<p>Indicates that the RMS file is sortkeyed.</p> <p>See the Registering RMS Files section for more information about specifying RMS keys.</p>	Required

Removing Index Registrations

Use the SQL remove index statement to delete the definition of an RMS index from an RMS gateway database. This statement removes all Ingres catalog entries for the index only.

When you issue the remove index statement, the underlying RMS index is not affected. Also, the registration for the index's base RMS file is unaffected.

The remove index statement has the syntax:

remove index *indexname*;

where *indexname* is an Ingres index name mapped to an RMS index.

Issue a commit statement immediately after a remove index statement to free system catalog locks and avoid unnecessary deadlocks.

Example of a Register Index Statement

The following register index statement registers the employee_number column as an index (called emp_idx) on the employee_tab table:

```
register index emp_idx on employee_tab
(employee_number)
as import
from '1'
with structure=sortkeyed;
commit;
```

Chapter 5: Accessing RMS Data Through the RMS Gateway

After you create your RMS gateway databases and register your RMS files and indexes, you can use Ingres tools to access the RMS data through the RMS gateway. This chapter discusses how to access your RMS data, including:

- Ownership and access of RMS gateway tables
- Connecting to a database
- Using Ingres tools with the RMS gateway
- SQL statements supported by the RMS gateway
- Transaction handling with the RMS gateway
- Locking RMS data files
- Tuning your applications to improve the performance of the RMS gateway
- Converting from earlier release RMS gateway applications

RMS File Access

The RMS gateway enforces the OpenVMS file protection system when you attempt to access an RMS file through the gateway. Access is governed by:

- File ownership
- File protection

To access RMS data through the gateway, you must have at least *read* access to a registered RMS file and intermediate directories in order to run queries against the data. If you cannot open the file, the RMS subsystem returns a runtime error message.

- Access Control Lists

An Access Control List (ACL) is a list of user identifiers (UIC) and general identifiers, which are arbitrary names assigned to collections of users.

- OpenVMS privileges

The RMS gateway uses the SYS\$CHECK_ACCESS OpenVMS system service to enforce RMS file security. Although it takes into account your authorized OpenVMS privileges, the gateway does not consider default privileges or the SETPRV privilege.

Table Ownership and Access

When you attempt to access a registered Ingres table in an RMS gateway database, the gateway checks your access rights as follows:

1. The gateway verifies your Ingres access rights to ensure that you meet one of the following conditions:
 - You own the table (that is, you are the user who registered the table)
 - If you are not the Database Administrator (DBA), then the DBA owns the table and has granted you access to it (see the Granting Privileges for RMS Gateway Databases section in the chapter “Managing RMS Gateway Databases”).
2. The gateway uses your login username to determine your access rights to the underlying RMS file.

If you are logging on through Ingres Net, your remote login ID determines your access privileges. See the *System Administrator Guide* for details.

When you access an RMS gateway database with an Ingres tool such as Query-By-Forms (QBF), you may be presented with the names of registered gateway tables to which you do not have access privileges. If you try to access such a table, the gateway returns a protection violation error.

Connecting to a Database

To access data in RMS files through the RMS gateway, you must be connected to the RMS gateway database that contains the tables in which you registered the files. The method you use to connect to the database depends upon how you want to access the data:

- In an embedded SQL program, use the connect statement
- With an Ingres tool, include the database name on the command line

Each method is described in detail in the following sections.

Issuing the Embedded SQL Connect Statement

If you are using embedded SQL statements in a host language program (such as C or COBOL), use the connect statement to indicate the gateway database you want to access. The connect statement must precede any other executable SQL statements that also access the database.

The connect statement has the syntax:

exec sql connect 'dbname/rms';

where:

- *dbname* is the RMS gateway database containing your registered RMS files
- */rms* extension tells the data manager to connect to an RMS gateway database

For example, the following statement connects you to the gateway database personnel:

exec sql connect 'personnel/rms';

See the *SQL Reference Guide* for more information about the connect statement.

Connecting to an RMS Gateway Database with an Ingres Component

You can use any of the available Ingres components listed in the Ingres Components Supported by the Gateway section in the chapter “Accessing RMS Data Through the RMS Gateway” to access RMS data through the RMS gateway. When you call one of the components, you must specify the gateway database in which you registered the RMS files you want to access.

You do this by including the gateway database name and */rms* extension on the operating system command line. The general command syntax to call an Ingres component is:

command **dbname/rms**

where:

- *command* is the command to call one of the Ingres components listed below
- *dbname* is the name of a gateway database
- */rms* tells the data manager to access an RMS gateway database

For example, to call Applications-By-Forms to access applications in an RMS gateway database called personnel, enter:

\$ abf personnel/rms

Accessing RMS Data in a Distributed Database Using Ingres Star

If you include an RMS gateway database as part of a distributed database using Ingres Star, you must specify the proper database extension to indicate how to access the RMS data:

- To access the RMS data only, use the `/rms` database extension
This tells the data manager to connect to your RMS gateway database.
- To access simultaneously the RMS data and any other data included as part of the distributed database, use the `/d` extension

This tells the data manager to connect to your Ingres Star distributed database that includes the RMS gateway database.

For example, assume you have registered the RMS gateway "personnel" database as part of the "corp_data" Ingres Star distributed database. You then can query the RMS data in either of the following ways:

- In the RMS gateway database only, by specifying `personnel/rms` when you call an Ingres tool such as Query-By-Forms
- Simultaneously in the RMS files and an Ingres database that is registered in the distributed database, by specifying `corp_data/d`

For more information about distributed databases, see the *Ingres Star User Guide*.

Connecting to a Database on a Remote Node

To connect to a database on a remote node through Ingres Net, include the virtual node (vnode) name for the database location. You must do this with both RMS gateway databases and Ingres Star distributed databases.

To connect to a database on a remote node, use the following syntax for the connect statement:

```
exec sql connect 'v_node::dbname/rms'  
or  
exec sql connect 'v_node::dbname/d'
```

To call an Ingres tool to access a database on a remote node, use the following syntax:

```
command v_node::dbname/rms  
or  
command v_node::dbname/d
```

You must include the two colons after the vnode name. For example, to use Query-By-Forms to access data in the parts gateway database on the remote philia node, type:

qbf phila::parts/rms

See the *System Administrator Guide* for more details about using Ingres Net.

Ingres Components Supported by the Gateway

You can access data in registered RMS files with any of the Ingres components listed in the following table. To call a component, enter the appropriate command, followed by the name of the gateway database in which the RMS data files are registered. Be sure to include the /rms extension to inform the data manager that you want to access an RMS gateway database.

For example to call Query-By-Forms to query the "inventory" RMS gateway database, enter the following command at the operating-system prompt:

```
qbf inventory/rms
```

The following table lists Ingres components available with the RMS gateway:

Component	Access Command
Applications-By-Forms (ABF)	abf
3GL Preprocessors	esqla (ADA), esqlb (BASIC), esqlc (C), esqlcbl (COBOL), esqlf (FORTRAN), esqlp (PASCAL)
Ingres Menu	ingmenu
Interactive SQL Terminal Monitor	isql
Query-By-Forms (QBF)	qbf or query
Report-By-Forms (RBF)	rbf
Report-Writer	report and sreport
VIFRED (Visual-Forms-Editor)	vifred
Vision	vision
OpenROAD	w4glrun

Supported Ingres Statements

The RMS gateway supports all Ingres SQL statements except for those listed in the following section. See the *SQL Reference Guide* for details about the syntax and usage of SQL statements.

SQL Statements Not Supported by the RMS Gateway

Do not use the create table and create index statements to create Ingres tables and indexes in your RMS gateway databases. These databases should contain only tables and indexes to which you have registered RMS data files and alternate keys—that is, those that you have created with the register table and register index statements.

To access Ingres data tables through the gateway:

1. Create the table in an Ingres database.
2. Create an Ingres Star distributed database. Include the Ingres table and any registered tables from an RMS gateway database.

Also, do not use the drop table or drop index statements to destroy tables and indexes in RMS gateway databases. To remove registered tables and indexes, use the remove table or remove index statement described in the chapter “Registering RMS Data to the Gateway.”

You cannot use the modify statement to change the physical storage structure of a table in an RMS gateway database. To make such changes, follow these steps:

1. Reorganize the registered RMS data file within RMS.
2. Use the remove statement to drop the table and index registrations from the gateway database.
3. Issue new register table and register index statements, specifying the new organization in the structure clause.

SQL Statements That Modify RMS Files

Observe the following restrictions when using insert, update, and delete statements to modify an RMS file through the RMS gateway:

- You must have included the update clause in the register table statement when you registered the file
- You cannot delete records from an RMS sequential file (RMS imposes this restriction)
- You cannot change a record’s length with an update statement to a fixed-length RMS file
- You cannot issue an insert or update statement that will result in exceeding the maximum record size of an RMS file

Repeat Queries

The RMS gateway supports the use of repeat queries, an Ingres feature that lets you retain query definitions for the duration of a session. You can use repeat queries with these SQL statements:

- delete (non-cursor)
- insert
- open|select (in a declare cursor statement)
- select (non-cursor)
- update (non-cursor)

To use a repeat query, preface the query statement with the keyword *repeated*, for example:

**repeated insert into TABLE1 (COL1, COL2)
values (:IVAR, :CVAR)**

Notice that you can use program variables with a repeat query statement just as you would with a regular query statement.

Specifying Integrity Constraints on RMS Gateway Tables

You can use the create integrity statement to specify an integrity constraint to use with updates performed through the RMS gateway. When you perform updates through the RMS gateway, Ingres enforces all integrity constraints on RMS files, including those applied through RMS itself.

If you perform an update directly through RMS, the gateway cannot enforce those constraints that you specify with the create integrity statement. Also, you can use the drop integrity statement to remove only those constraints that you create through Ingres; you must use RMS to remove those constraints that you create directly through RMS.

You must be the owner of the table on which you wish to specify a constraint. In an RMS gateway installation, this means that you are the user who registered the table.

See the *SQL Reference Guide* for more information about specifying constraints.

Ingres defines three types of referential integrities—rejections, nullifies, and cascades. The RMS gateway fully supports nullifies and cascades on registered RMS files. You cannot use rejections through the gateway because they depend on the ability to roll back a transaction.

Using the Ingres Optional Product Extension

This section discusses the use of the optional Ingres product extension, Object Management Extension, with the RMS gateway.

Object Management Extension

The features of the Object Management Extension currently are not available through the RMS gateway because the gateway itself uses the extension to manage all RMS data type conversions.

RMS Gateway Transaction Handling

The RMS gateway supports full multi-statement transaction handling for updates to Ingres data contained in RMS gateway databases. You can use all Ingres transaction statements, such as set autocommit and savepoint. See the *SQL Reference Guide* for more information about Ingres transaction statements.

Be aware that the RMS gateway does not manage RMS transactions. The gateway cannot journal changes to RMS files. If your RMS gateway database contains both Ingres tables and RMS files, transaction handling statements refer only to the Ingres data.

Therefore, when you develop your gateway applications, you must remember to provide a way to handle inconsistent RMS data in the event of:

- System failures during multi-statement transactions involving RMS records
- Overwrites to RMS data caused by rival simultaneous processes

Also, the RMS gateway makes no provision for logging or recovery of RMS data that you have committed. For more information about recovering data with the RMS gateway, see the Backup and Recovery of RMS Gateway Databases section in the chapter “Managing RMS Gateway Databases.”

Locking RMS Data Files

The RMS gateway implements standard Ingres locks on RMS data files when multiple users attempt concurrent updates through the gateway. These locks are held at the table level only. The RMS subsystem, rather than the gateway, is responsible for holding locks at the record level in an RMS data file.

See the *Database Administrator Guide* for more information about Ingres locks. See the appropriate RMS documentation for more information about RMS locking.

Simultaneous updates, inserts, or deletes of an RMS file through the RMS gateway can result in corrupted data. When designing applications to use with the gateway, you should ensure that modifications to an RMS file occur serially.

Setting Timeout Values

If you are accessing RMS files through the gateway and through RMS simultaneously, you should set a *timeout* value to prevent gateway applications from waiting for an indefinite period for RMS records that are locked by other processes. The timeout value can be between 0 and 255 seconds inclusive. Note that 0 seconds (the default value) causes the gateway to wait indefinitely.

You can use the set lockmode statement to set a timeout value for a specific application or for an entire gateway session. Each method is described in the following sections. Always issue a commit or rollback statement before you issue a set lockmode statement to make sure that your application is not in an active multi-statement application.

See the *SQL Reference Guide* for more information about the set lockmode statement.

Setting a Timeout for an Application

To set a timeout value for an application, include the following SQL statement:

```
set lockmode on tablename
where level = page,
timeout= n
```

where:

- *tablename* is the name of the RMS table as registered with the RMS gateway
- *n* is the number of seconds (up to 255) that the application should wait to access a record

Note that the page level lock in the above statement represents a lock on an RMS record.

Setting a Timeout for a Gateway Session

You can declare a single timeout value for all RMS files to be accessed during a single RMS gateway session by issuing the following SQL statement at the beginning of the session:

```
set lockmode session
where level=page,
timeout= n
```

where *n* is between 0 and 255 seconds.

Note that the page level lock in the above statement represents a lock on an RMS record.

Alternating Between Application-level and Session Timeouts

An application-level timeout value overrides a value that you have set for an entire gateway session. You can restore the session timeout value by issuing the following SQL command:

```
set lockmode...timeout=system
```

Protecting Updates to RMS Files

You can use the set lockmode statement to keep simultaneous RMS gateway users from overwriting updates to RMS files. To lock out rival gateway applications, issue the following statement:

```
set lockmode on tablename  
      where level=table,  
      readlock=shared|exclusive
```

where:

- *tablename* is the name of the RMS file as registered with the gateway
- **shared** gives read-only access to other gateway users
- **exclusive** prevents other gateway users from reading the RMS file as well as writing to it

Be aware that if Ingres applications through the RMS gateway and non-Ingres applications are accessing an RMS file simultaneously, RMS locks take precedence over Ingres locks.

The locking process:

- Begins when you issue your first query against the RMS file
- Ends when a commit or rollback is executed

Remember that for multi-statement transactions, the RMS gateway supports Ingres locks only. The gateway supports RMS locks only for individual queries. For more information, see the RMS Gateway Transaction Handling section in this chapter.

Performance Tuning Your Applications

When you develop applications to access data through the RMS gateway, you can increase performance in several ways:

- Use the default data type mappings when you register your RMS files.

The gateway can process queries more quickly because these mappings require little or no data type conversion. See the Mapping Ingres and RMS Data Types section in the chapter “Registering RMS Data to the Gateway” for a list of data type conversions.

- Register any secondary indexes that exist for your RMS data files.

Although you cannot create Ingres indexes for tables in your RMS gateway databases, you can use the register index statement to inform the RMS gateway about secondary indexes that already exist for your RMS files. The gateway then uses these indexes to execute your queries more efficiently.

See the Registering RMS Indexes section in the chapter “Registering RMS Data to the Gateway” for more information.

- If you are issuing a statement more than once, use repeat queries.

With a repeat query, the gateway only has to define the statement once for the entire application. See the Repeat Queries section in this chapter for more information.

- Register only those columns in an RMS file that you require for an application.

Because data type conversion is performed on all registered columns, you can improve performance by limiting the number of columns that you register. In some cases, you can optimize performance by registering an RMS file as several Ingres tables, with different columns appearing in each table.

Chapter 6: Access to Repeating Group Data in the RMS Gateway

This chapter describes how the RMS gateway provides read-only relational access to a repeating group. The following topics are discussed:

- Read-only relational access to repeating groups (array data)
- Multiple record stream access of open RMS files
- Reduction of RMS file open and close operations to improve performance
- Limited support of RMS descending key access to an RMS file
- A read-regardless-lock option that reads a record even if it is locked against read
- A startup option that sets the RMS gateway to be a read-only server so that no file will be open for write access
- How the RMS gateway optionally buffers RMS Record File Address (RFA), dynamically simulates the Ingres Tuple Identifier (TID), and fully supports any file size of an indexed RMS file when RFAs are buffered
- The Ingres `optimizedb`, `sysmod`, and `statdump` utilities generate statistics so that the query optimizer can produce optimal query plans

Repeating Group Access

A repeating group is a varying set of entries within a record. For example, a repeating group can be data captured in an array structure, the rank of which may vary from record to record. Intra-record data representations, such as repeating group, are inherently non-atomic, and therefore cannot be normalized without violating relational model constraints. The current relational model cannot handle any form of nested or hierarchically related groups of data within a record.

Often, a repeating group may be the best way to represent data. In most file systems, storing data in an array is simple. Moreover, an array may be the most natural data structure for the application. However, this poses a problem for database gateways to provide relational access to file systems that support and contain this kind of array data.

The RMS gateway provides a way to map data in RMS files to Ingres tables, thus allowing relational access of the data in the mapped RMS files. The RMS gateway supports read-only access of repeating groups. Features like multi-stream file access and sharing open files among threads or non-Ingres applications are necessary conditions for supporting read-only access to repeating groups.

Limited support of access to RMS records through descending RMS keys is also supported. The Ingres DBMS supports only ascending keys and the RMS gateway cannot totally bypass this restriction. Certain simple descending key comparisons, however, are supported. For example, you can, with certain restrictions, do exact search, partial search, and range search on an RMS descending key. For more information, see the RMS Descending Key section.

The following sections describe how to use the read-only access of repeating group feature to retrieve data by explaining:

- Limitations of repeating group support
- How to register an RMS file with repeating group as an Ingres table
- How to retrieve repeating group data from registered table
- Important characteristics of repeating group data retrieval

Usage Guidelines

Because of the unique characteristics of a repeating group and the challenges of handling repeating group in the relational model, the following usage guidelines are imposed on relational access of repeating group:

- Access to RMS repeating group is strictly read-only.
- The RMS record must have two parts, a fixed portion and a portion that defines the repeating group.
- Although not required, it is desirable that the RMS file be an indexed file and registered as an indexed table. The primary key, if any, must be part of the fixed portion of the RMS record.
- The RMS record containing a repeating group must be fixed-length format.
- There should be only one repeating group per RMS record.
- The repeating group must be located at the end of the record. No data follows the repeating group.
- There should be no nesting of repeating groups, although the repeating group may be an aggregate of data with different data types.
- The repeating group must start with a count field to indicate the number of available repeating members. This count field marks the beginning of the repeating group in the RMS file.
- All repeating members must be contiguous, no mix of meaningful and non-meaningful repeating data is allowed in a repeating group.
- The size of repeating group members must be fixed-length.
- In COBOL or CDD+ these characteristics describe the OCCURS DEPENDING ON clause behavior.

Registering a Table with Repeating Group

The following example shows how to register a table against an RMS file that contains a repeating group. It is an employee file that contains employee personal information as fixed data and their spouse/dependent information as repeating data.

```
REGISTER TABLE employee (
  last_name      char(20),
  first_name     char(20),
  sex            char,
  age            smallint,
  salary         integer,
  seq_no         integer is 'imaginary',
  sd_total       integer, /* count of spouse + dependents */
  sd_last_name   char(20), /* may have different last name */
  sd_first_name  char(20),
  status         char, /* S: spouse, D: dependent */
  sd_sex         char,
  sd_age         smallint)
AS IMPORT FROM ' ' WITH DBMS=RMS, STRUCTURE=SORTKEYED,
KEY=(last_name, first_name), ROWS=nnn, NODUPPLICATES.
```

This statement registers an RMS file whose records include repeating group data. The RMS gateway server detects the presence of a repeating group when it encounters the new RMS gateway data type "imaginary" in the extended format field specification of a column of the corresponding registered table.

Imaginary Data Type

For each RMS file that contains a repeating group, its registration as an Ingres table must have a column with the Ingres integer data type and an extended format of "imaginary." This column must be defined immediately before the repeating count field, which precedes the repeating group. The imaginary data type indicates that all the fields following it define the fields of the repeating group.

This imaginary field, which in the preceding example is the seq_no column, exists as part of the Ingres row but does not actually exist in the underlying RMS record. It is treated as a computed field during repeating group retrieval. It must be an integer data type. When data is converted from the RMS record to the corresponding Ingres row, the RMS gateway server calculates its value and inserts it into the proper place in the Ingres row buffer. The value of this field also gets returned to the application. The value contained in this field indicates the sequence number of the particular repeating element in the repeating group of the record.

The sd_total column should contain the number of repeating elements in a repeating group. Since not all of the repeating group may be full, this field must exist in the RMS file so that the RMS gateway will not retrieve nonexistent data in the repeating group. At first glance, this field should not be part of the repeating group since it does not actually repeat itself inside an RMS record. However, it remains an integral part of a repeating group because without its specification, there is no way for the RMS gateway to tell how many repeating members are actually in the group.

Read Data from Repeating Group

The following syntax retrieves repeating group data:

```
SELECT * FROM employee;
```

This syntax breaks each RMS record into the number of rows that `sd_total` specifies. Each row contains the fixed portion of an RMS record, followed by the sequence number of the repeating member (starting with the value "1"), then followed by the total number of repeating members, and finally each component of the repeating group.

The preceding example causes a sequential table scan and displays possibly ($m \times n$) rows of data, where "m" is the number of RMS records in a file and "n" is the maximum number of members that the repeating group may have. Because the scan can be a long process, it is better to retrieve based on primary key and with restrictions to reduce data retrieval time.

The following example chooses all employees that have the last name SMITH:

```
SELECT * FROM employee WHERE last_name = 'SMITH';
```

Since `last_name` is part of the primary key, a key search is performed on the underlying RMS file, (assuming that `last_name` is also part of the RMS file's primary key).

The following example chooses all employees with the specified last names and who each have a total of 3 for the field spouses + dependents:

```
SELECT * FROM employee WHERE last_name IN ('DAVIS',  
      'BRAT', 'SMITH') AND sd_total = 3;
```

The RMS gateway server performs a primary key search to match all records with these three patterns.

Characteristics of Repeating Group Access

The following sections describe characteristics of repeating group access of which you should be aware.

Zero Repeating Group Members

Assume that a valid registration of a repeating group table has been done, but the count field specifies that there are zero repeating members in certain RMS records. How many rows should be returned from a record with a repeating count field of zero? The decision made for this case is to return zero rows, with the following reasoning. If non-zero rows are returned from an RMS record with no repeating members (for example, one row) there is a conflict because for an RMS record with one repeating member, we again return one row. For consistency, it is better to return no rows even when the fixed portion is available. If you want to view the complete fixed portion of records, a new table should be registered against the same RMS file, with the register command containing only the fixed portion of the RMS record.

Unload and Reloaddb	Do not use the unloaddb and reloaddb utilities with repeating group support because the unloaded table will be in a normalized form and reloaddb cannot restore it to its original form.
Row Estimates for the Query Optimizer	In order for the Ingres Query Optimizer to better estimate query cost and choose an efficient query plan, table registration should provide as correct a row estimate as possible. This is true for any RMS gateway table, but it is also important for repeating group tables due to the potentially large number of rows returned from a table with a repeating group.
Column Members Must Be Contiguous	The registration of columns on the repeating portion of an RMS record should be contiguous. Although extended format is allowed, do not use it to skip data in an RMS record, as allowed with normal RMS gateway registration.
Repeating Group Tables Must Be Read-only	If you register a table against an RMS file with a repeating group, you must not use the update option and/or do a delete. Severe data corruption could occur. Read-write access is allowed to non-repeating group tables.

Share Open Files and Multistream Record Access

To support repeating groups and to improve performance, an open file must be available server-wide. In the RMS gateway, an open file remains open until one of the following events occurs:

- The server is shut down
- A table registered against the open file is removed with no outstanding usage of this file by any other registered table
- The server internally decides to close this file
- After the last usage of the file by a user if the `II_RMS_CLOSE_FILE` config parameter is specified

This solves the problem of an RMS file being open only for a short period of time. Its open and close states were query-based, and an open file was not shared by different queries and sessions. This caused excessive RMS file operations and was sometimes a significant factor in causing performance degradation. Sharing open files is not visible to RMS gateway users and can improve performance.

To share RMS files, the DBA and system administrator must define special startup parameters before starting up the RMS gateway server. These parameters are defined in the RMS gateway configuration procedure. To modify parameters after an installation has been configured, the `II_CONFIG:config.dat` file must be updated through CBF.

These config.dat variables are:

```
ii.nodename.rms.*.ii_rms_fab_size
```

```
ii.nodename.rms.*.ii_rms_hash_size
```

where x and y are small integers, for example, $x = 16$, $y = 7$.

II_RMS_FAB_SIZE defines the resource to cache RMS file information.
II_RMS_HASH_SIZE defines the hash table size used for hashing RMS file information block.

The II_RMS_FAB_SIZE config parameter has a default value of 16. The II_RMS_HASH_SIZE config parameter has a default value of 7.

Since an open file is shared by different threads and/or sessions, single stream record access is not adequate. Files must be accessed by multiple record streams. To do this in the RMS gateway server, specify multistream record access (MSE) during file open operation. This is transparent to all users.

If the DBA and system administrator require that a file be closed after the last session has released the file, you can define the parameter `ii_rms_close_file` to be ON before starting up the RMS gateway server, as follows:

The config.dat variable is:

```
ii.nodename.rms.*.ii_rms_close_file on
```

The default value of II_RMS_CLOSE_FILE is OFF.

RMS Descending Key

Ingres only supports ascending keys. RMS supports both ascending and descending keys. Although the RMS gateway server cannot ignore this Ingres restriction, it is possible to partially support RMS descending keys without major modifications to the Ingres DBMS. The limited descending key support allows all descending keys search operations to be based on LITERAL values.

Because the register command does not indicate descending or ascending key types, the RMS gateway server must inquire about this information from an RMS file when the file is opened, and then buffer this information while the file remains open. Inquiry and buffering consume CPU and memory. The maximum number of keys per indexed file allowed by RMS is large (255). The RMS gateway uses up to a maximum of eight keys, including the primary key. The corresponding RMS key numbers must be from zero to seven.

Read-Regardless-Lock Option

The read-regardless-lock option allows reads to continue although the underlying RMS file record is locked. It is designed for the situation where a user's activity through the RMS gateway is mostly read-only, and it is not important that there may be possible inconsistencies in viewing of data.

To enable this option, define the following parameter before starting the RMS gateway server.

Note: This parameter is defined in the RMS gateway configuration procedure. To modify a parameter after an installation has been configured, the `II_CONFIG:config.dat` file must be updated.

The `config.dat` variable is:

```
ii.nodename.rms.*.ii_rms_rrl on
```

The default value of `II_RMS_RRL` is OFF.

If you access an RMS record through the RMS gateway server, you may get a read error due to a non-gateway application's lock on that record.

This is a server-wide flag and should be defined by the system administrator and/or DBA. Once defined, it is available to all users throughout the life of the server installation.

If viewing strictly consistent data is more important, set this parameter's value to OFF before starting the RMS gateway server.

Read-Only Server Option

The Read-Only Server option prevents the RMS gateway server from modifying data in the RMS files that it accesses.

The Read-Only Server option should be used if the RMS files are used mainly as a data warehouse, if there are no expected update activities through the RMS gateway, and users demand a guarantee of no data alternation from the RMS gateway.

To bring up a read-only RMS gateway server, simply define a parameter as follows before starting the RMS gateway server.

Note: This parameter is defined in the RMS gateway configuration procedure. To modify a parameter after an installation has been configured, the `II_CONFIG:config.dat` file must be updated.

The `config.dat` variable is:

```
ii.nodename.rms.*.ii_rms_readonly on
```

The default value of `II_RMS_READONLY` is OFF.

This server-wide flag remains effective until the server is brought down. To bring up a read-write server again, this parameter must be set to OFF.

Buffering Record File Address (RFA) Option

The RMS Record File Address (RFA) can be optionally buffered if the server is brought up as a read-only server. The buffer address is returned as a unique Ingres Tuple Identifier (TID). The TID is unique only in a query. Also, the RMS RFA of a record is buffered only if it is fetched through a secondary index.

To use RFA buffering, set a parameter value as follows before starting the RMS gateway server.

Note: This parameter is defined in the RMS gateway configuration procedure. To modify a parameter after an installation has been configured, the `II_CONFIG:config.dat` file must be updated.

The `config.dat` variable is:

```
ii.nodename.rms.*.ii_rms_buffer_rfa on
```

The `II_RMS_BUFFER_RFA` default depends upon whether the value of `II_RMS_READONLY` is OFF. If `II_RMS_READONLY` is ON, this value will also be ON. Also, the TID is unique only within a query, not within a session.

Previously, a TID was provided by transforming the RFA of an RMS record. However, the file could not be larger than 4 GB and/or each RMS bucket could not contain more than 512 records. The 512 records limit is easily reached if, for example, frequent insert and delete of RMS records happens more than 512 times in the same RMS bucket. When this happens, you need to run the OpenVMS convert utility to correct the problem. Too, errors will occur if the RMS bucket size is very large and the RMS record size is very small so that a bucket is packed with more than 512 records. In this case, the only solution is to create an RMS file with the same records but smaller bucket size.

In the RMS gateway, the only time a unique TID must be returned is when this record access is through a secondary index. The returned TID may later be used to access the record in the base table. If a record is accessed from the base table, the value of the TID returned is inconsequential. For this reason, even for a read-only server, only the RFA from an index probe is buffered.

RMS Gateway Database Commands

The Ingres commands `optimizedb`, `statdump`, and `sysmod` are supported in the RMS gateway. When specifying the RMS gateway database name, use the `/rms` extension.

Limitations of Optimizedb

Be aware of the differences between the following two commands:

- The `-zs[s]n` sampling option of `optimizedb` is not supported for RMS gateway. This option requires the full support of Ingres tuple identifiers (TIDs) in the target base table. The RMS gateway does not fully support TIDs for base tables registered to RMS files.
- The `-zp` option of `optimizedb` is expanded for RMS gateway. The previous function is retained: When used with the `-i` flag, this option directs `optimizedb` to read the row and page count values in the file specified with the `-i` flag and to store those values in the appropriate system catalog. See the Specialized Statistics Processing section in the chapter “Using the Query Optimizer” in the *Database Administrator Guide* for more information about this feature.

The `-zp` option functions as it does because, unlike native Ingres tables, base tables in an RMS gateway database are registered against RMS files. These files may be accessible to non-gateway applications. For this reason, there is no way to keep system catalog values for a table’s row count up to date; furthermore, the notion of page counts is not a direct mapping between the underlying RMS file’s block counts (or bucket counts) to Ingres table pages. The `optimizedb` command normally disregards these input values when creating statistics and leaves the catalog untouched because of their critical nature when native Ingres tables are operated on.

When `-zp` is specified in `optimizedb` without the `-i` flag, the command uses the row count value from its scan of the subject base table and stores it into the appropriate system catalog. The page count values remain as they are in the system catalog. In this way, the row and page count values for a base table in the system catalogs and the newly generated statistics on that subject table will be in sync at the time `optimizedb` completes.

There is an added benefit to specifying the `-zp` option when secondary indexes are present. If one or more register index commands had instantiated the presence of Ingres secondary index tables in the RMS gateway database, then the `-zp` option also directs `optimizedb` to update the estimated number of rows for the secondary index table to be exactly equal to the value of `num_rows` in its associated base table. Similarly, the `-zp` option directs `optimizedb` to update the estimated number of physical pages for the secondary index table to be exactly equal to the value of `number_pages` in the same associated base table.

For more information see the `optimizedb` command description in the *SQL Reference Guide*.

Using Register and Remove Commands with Stored Statistics

The `remove index` command removes a registered secondary index, but it does not affect any existing statistics for its associated base table.

No statistics exist for secondary index “tables,” but if the index had existed at the time that `optimizedb` was run and one or more of its associated columns were specified to have statistics gathered with the `-zk` option, then these statistics persist even after the `remove index` command. A new `register index` command can be invoked, and any subsequent query plan that uses any of the columns specified by that new command can use any of the persistent statistics.

The `remove table` command removes both the registered table and any associated registered secondary indexes. All statistics pertaining to the removed base table are deleted from the database.

A `register table` command with a `rows=` parameter creates in the system catalogs, among other values, the values for the estimated number of rows and the estimated number of physical pages.

Note: A `register table` command without a `rows=` parameter uses a default value for the number of rows, and so acts the same—as far as these database utilities are concerned—as the case in which the `register table` command with a `rows=` parameter is specified.

Data Types of Columns

The `optimizedb` command works on data in a column in its converted Ingres data type format. No consideration for the underlying RMS data type format is made. If the RMS column data type is mapped to an Ingres column data type, the statistics generated from that column should, when used by the query optimizer in determining the best query plan strategy, accurately reflect the underlying RMS data.

However, if an Ingres column is mapped in an unnatural manner or to a segmented RMS alternate index, then care must be taken. Usually, statistics generated on columns of RMS integer, floating-point, or string data types mapped and converted by the RMS gateway server to their corresponding Ingres data types will exhibit proper histograms. You can experiment with the `-zr#` and `-zu#` options of `optimizedb` to generate the best exact or inexact histograms for a particular column’s data values.

Statistics on Ingres date data types are not fully supported. Statistics should not be generated for any Ingres column data type that is mapped to an aggregate of RMS fields of disparate data types.

Important! When doing retrievals of data with partial range search conditions from indexed RMS files, be aware of the following: When the RMS file contains a primary key that is defined as a simple key, it includes one or more contiguous bytes. If the data types of the included fields have compatible sort order sequences, then partial range searches using the aggregate of these fields as a simple primary key will produce correct results. However, if a simple key is made up of one or more fields of incompatible data types (for example, a left-justified string of unsigned, 8 bit-values and a signed 2-byte integer with at least one positive and one negative actual field value), the resulting partial range search could produce incorrect results.

When to Run the Utilities

RMS files registered as a base table in an RMS gateway database are not exclusively accessed by the RMS gateway server. Therefore, it is recommended that you run the `optimizedb`, `statdump`, and `sysmod` database utilities at the following times:

- Immediately after using `register table` and `register index` commands in the database
- After enough change activity has occurred in the database but before existing query plans to that database become grossly inefficient

Statistics generated by `optimizedb` affect the speed of query processing, not whether the query will execute. More complete and accurate statistics generally result in more efficient query execution strategies, which further result in faster system performance.

For the most accurate and trouble-free generation of statistics or modification of system catalog structures, it is recommended that all write or change activity to any and all RMS files referred to by the target registered base tables be quiesced while running these database utilities against one or more base tables in an RMS gateway database. This is true for gateway databases for which significant non-gateway accesses might be occurring to underlying RMS files during gateway-server access and particularly during the execution of these database utilities.

Suitability to RMS Gateway Tables with Repeating Groups

The `optimizedb`, `statdump`, and `sysmod` database utilities can be used on base tables registered to RMS files that contain repeating members if these files are properly registered as repeating group tables in the RMS gateway database.

Note: Because the query optimizer assumes that it is working with flat relational tables, it cannot handle optimizations of queries to what are in effect nested relations. A table with a repeating group always appears to the query optimizer as a flat relational table. Generating statistics on individual columns of these flat relational tables does not generate any statistical information about the nested relationships between any columns belonging to the repeating group.

For example, if any master-detail type relationships exist between data in different columns of a table with a repeating group, the statistics generated by `optimizedb` will not directly reflect this to the query optimizer, which could not, in any case, do anything with this kind of information.

Suggested Command Sequence

You should update the Ingres system catalogs for a base table registered to an RMS file and any of its associated secondary indexes with an accurate estimate of the number of rows and the number of physical pages it contains. To do this, use the `-zp` option when using `optimizedb` to create statistics. Then run `sysmod`. The following section provides examples.

Examples

Generate full statistics for all columns in all tables in the `empdata/rms` database, and update the system catalogs with row and page count values used when generating these statistics, as follows:

```
$ OPTIMIZEDB -zp empdata/rms
```

Optimize the system tables in the `empdata/rms` database using:

```
$ SYSMOD empdata/rms
```

You can specify that only the system tables affected by `optimizedb` be processed by using:

```
$ SYSMOD empdata/rms iistatistics iihistogram iirelation
```

Generate statistics for key or indexed columns in the `employee` and `dept` tables. Plus, generate statistics for the `dno` column in the `dept` table. For example:

```
$ OPTIMIZEDB -zp -zk empdata/rms -remmployee -rdept -adno
```

Next, generate minmax statistics on the additional column `div` in the `dept` table. Because the row and page counts in the system catalogs for that base table were updated in the previous example, do not repeat this.

```
$ OPTIMIZEDB -zx empdata/rms -rdept -adiv
```

Optimize only the system tables in the `empdata/rms` database affected by `optimizedb`. Because `-zp` was not used `iirelation` was not affected, so it does not have to be specified. For example:

Chapter 7: Managing RMS Gateway Databases

This chapter discusses how to manage your RMS gateway databases, including:

- Statements and utilities available to the database administrator of an RMS gateway database
- Backing up and recovering RMS gateway databases
- Error handling through the RMS gateway

Granting Privileges for RMS Gateway Databases

Ingres provides several statements and utilities to control access to RMS gateway databases:

- The `accessdb` utility lets you grant access to specific databases, as well as grant permission to create databases. You must have Ingres superuser privileges to use the `accessdb` utility. For more information, see the *Command Reference Guide*.
- The `grant` statement lets the database administrator grant specific permissions—for example, `select`, `delete` or `update`—on tables and views in RMS gateway databases.

When a user issues a query through the RMS gateway, the gateway verifies these Ingres privileges, along with any privileges granted directly through RMS. However, if you issue a query directly through RMS, privileges given with the `grant` statement are not verified.

Be aware that the DBA cannot grant privileges for tables that other users own. In an RMS gateway database, the user who registers a table is its owner. See the Registering Sharable Tables section in the chapter “Registering RMS Data to the Gateway” for more information.

- The `drop permit` statement removes from tables and views in RMS gateway databases any permissions that were issued with the `grant` statement. It has no effect on permissions granted directly through RMS.

See the *SQL Reference Guide* for detailed descriptions of the `grant` and `drop permit` statements.

Note: OpenSQL does not support the `grant` and `drop permit` statements.

Using Ingres Database Commands and Utilities with RMS Gateway Databases

Ingres provides a variety of commands and utilities that you can use to manage databases. This section describes the implications of using these functions with RMS gateway databases. You must be the database administrator for a database to use any of these commands and utilities with that database. Additional utilities that are used with repeating group data are also available. See the RMS Gateway Database Commands section in the chapter “Access to Repeating Group Data in the RMS Gateway” for more information.

For more information, see the *Database Administrator Guide* (except for specific commands included in the list that follows).

The following table lists the RMS gateway database administration commands and utilities:

Utility	Description	RMS Gateway Implications
accessdb	Lets the Ingres system administrator or other Ingres superuser define user access to databases.	Provides access to RMS gateway databases only. Use RMS file protections and Access Control Lists to control access to RMS files registered in gateway databases.
catalogdb	Lets you view information about all databases that you create.	
copydb unloaddb	Copies the registration of tables and indexes from one RMS gateway database to another.	
createdb	Creates RMS gateway databases in which you register your RMS data files and indexes. You must be authorized by an Ingres superuser to create a database. See the chapter “Creating RMS Gateway Databases” for more information about creating RMS gateway databases.	

Utility	Description	RMS Gateway Implications
destroydb	Destroys RMS gateway databases. This removes the registration structures but does not affect the underlying RMS data.	
iinamu	Displays information about all data managers currently registered with the Ingres Name Service process; lets you add or delete data managers.	You must specify a server type of rms for the RMS gateway data manager. The GCF address for the RMS gateway data manager begins with the string II_RMS_.
iimonitor	Monitors or stops the operation of all data managers at an Ingres site.	Can be used to monitor or stop an RMS gateway data manager.
lockstat	Provides information about all Ingres locks currently being held.	Does not include any OpenVMS locks on an RMS file if concurrent updates are done directly through RMS. The RMS gateway only holds Ingres locks at the table level.
logstat	Provides information about the Ingres logging system.	Log files only contain information about catalogs in RMS gateway databases; they do not maintain any information about the underlying RMS files registered in those databases.
netutil	The Ingres Net network management utility.	No difference with the RMS gateway.
optimizedb	Used to generate statistics on the specified columns. These statistics are used by the query optimizer to select an efficient query processing strategy. See the <i>Command Reference Guide</i> for more information about the optimizedb utility.	
rcpconfig	Reconfigures the locking and logging system values for an Ingres data manager.	No difference with the RMS gateway.

Utility	Description	RMS Gateway Implications
statdump	Prints statistics contained in the iistats and iihistograms catalogs of the Standard Catalog Interface. See the <i>Command Reference Guide</i> for more information about the statdump utility.	
sysmod	Modifies the system tables to predetermined storage structures. See the <i>Command Reference Guide</i> for more information about the sysmod utility.	
upgradedb	Converts RMS databases to new release.	
verifydb	Used to recover RMS gateway databases that have been corrupted. See the <i>Command Reference Guide</i> for more information about the verifydb utility.	

Backup and Recovery of RMS Gateway Databases

The Ingres backup and recovery functions are only relevant to the Ingres catalog entries that an RMS gateway database contains; they have no effect on the underlying RMS data files registered in the database. The appendix “RMS Gateway Catalogs” describes the Ingres catalog entries that describe your registration structures.

To protect your RMS data, use the OpenVMS backup and recovery techniques already in place for your RMS data files. Consult the appropriate OpenVMS documentation for information.

The following table describes how various Ingres commands and utilities function with RMS gateway databases:

Command	Description
ckpdb	Saves the entries to the Ingres catalogs in the database at the time the checkpoint is taken
rollforwarddb	Restores the catalogs in an RMS gateway database
set journaling	Maintains journaling on all entries to the Ingres catalogs in an RMS gateway database

See the *Command Reference Guide* for more information on ckpdb, rollforwarddb, and set journaling.

RMS Gateway Error Handling

This section discusses how the RMS gateway handles runtime errors. There are three classes of errors that can occur when you use the RMS gateway:

- Standard errors in Ingres operations
- Gateway errors resulting from user operations
These errors are mapped to Ingres Generic Errors.
- Internal gateway or RMS errors
These errors are written to the Ingres installation log file and a single, standard error is returned to the Ingres application.

Standard Ingres Errors

While using the RMS gateway to access RMS files with an Ingres tool or application, you may encounter errors in standard Ingres operations—for example, an SQL statement syntax error.

Handle these errors in the same way you would when accessing Ingres databases in standard Ingres:

- With an interactive tool or application, the error text is displayed on the screen.
- In an embedded SQL application, you can retrieve the error through the SQL Communications Area (SQLCA).

See the *SQL Reference Guide* for more information about using the SQLCA.

RMS Gateway User Errors

The RMS gateway maps RMS-specific errors to a series of gateway User Errors used throughout all Ingres gateways. These errors are detected by the Generic Gateway Facility (GWF), which is the part of the data manager that contains the RMS gateway.

When the gateway detects an error in an RMS input or output operation, it returns the:

- Ingres Generic Error code
- GWF error code
- GWF error text

With an interactive Ingres tool or application, gateway user errors are displayed on your terminal screen in the same way as standard Ingres errors. The following section describes how to retrieve gateway user errors in an embedded SQL application.

The appendix "Generic Errors" lists the errors in RMS input/output operations that the RMS gateway can return as gateway user errors.

Retrieving Errors in an Embedded SQL Application

In an embedded SQL application, you can access various parts of an error with the `inquire_ingres` statement and the SQLCA (SQL Communications Area) variables. The following table describes how to retrieve specific parts of an error:

To Retrieve	Use
Generic Error Code	SQLCA.SQLCODE variable
INGRES SQLCODE	SQLCA.SQLERRD(0) variable
First 70 characters of the GWF error message text	SQLCA.SQLERRM variable
Generic Error code	<code>inquire_ingres</code> <code>errorno</code> statement
Entire GWF error message text	<code>inquire_ingres</code> <code>errortext</code> statement
Entire GWF error message text	<code>whenever SQLERROR</code> calls <code>SQLPRINT</code> statement

Avoiding Registration Errors

Runtime errors frequently occur because you register your RMS files incorrectly or specify invalid data type mappings. The following are some of the most common errors that can occur:

- RMS rejects data inserted into an RMS date field because the data is not in a valid OpenVMS date format (usually the data is in an Ingres date format).
- RMS double-precision floating point (G format/T format) data values are beyond the range of the corresponding SQL float data type.
- You specify an incorrect offset when registering non-contiguous file fields or registering a file as multiple tables.
- You register an RMS index incorrectly—by listing the key fields in the wrong order, for example.

The appendix “Generic Errors” includes the RMS gateway registration errors. See the chapter “Registering RMS Data to the Gateway” for details about registering tables and indexes and converting data types.

The errlog.log Error Log File

The RMS gateway makes use of the error log file that the data manager (server) maintains. The location of this file is:

```
ii_system:[ingres.files]errlog.log
```

The errlog.log file contains information about each request that the server receives, including:

- Server name
- Session ID
- Date and time at which the error occurs
- Brief description of the problem

The errlog.log file grows continuously and can become very large. To reduce the size of the errlog.log file:

1. Stop the RMS gateway server process and any other servers running at your Ingres installation.

You must do this because all data managers at an installation share the same log file.

2. Delete the errlog.log file.
3. Restart all servers.

Ingres creates a new, empty file.

See the *System Administrator Guide* for information about starting and stopping server processes.

Debugging Complex Query Errors

If you get an error when you are using the RMS gateway to execute complex queries and joins, these steps can help you isolate the problem:

1. Simplify the query as much as possible by removing:
 - The order by clause
 - The where clause
 - Any aggregates
 - Any functions
2. Access the RMS gateway database locally (if you have been going through Ingres Net).
3. Test the query on a valid table in a standard Ingres database.

Appendix A: Generic Errors

Ingres uses a Generic Error mapping scheme to make error handling consistent across all Ingres gateways. When the Ingres RMS gateway data manager encounters an error in an RMS input or output operation, it maps that error to an Ingres Generic Error.

The table below lists the Generic Errors that the gateway can return. For each error, the table provides:

- Ingres Generic Error code and message
- Generic Gateway Facility (GWF) status code and text description

The GWF is the component of the Ingres data manager that contains the RMS gateway and performs the gateway exit routines.

Generic Error Codes

In the descriptions provided in the following table, the notation xxx represents a specific value that appears in the actual GWF text description.

As the table indicates, a single Generic Error may represent any of several GWF errors. You can retrieve information about the specific error with one of the methods described in the RMS Gateway Error Handling section in the chapter “Managing RMS Gateway Databases.”

Generic Error Code and Message	GWF Status Code and Description
31000 E_GE7918_SYNTAX_ERROR Statement syntax error	E_GW54A0_WRONG_NUM_PARMS The wrong number of data type specifiers was provided on one of the RMS field definitions in the REGISTER TABLE command. Check the syntax of the REGISTER TABLE command and re-issue the command.
	E_GW54A1_NO_SUCH_DATATYPE An unknown data type was provided on one of the RMS field definitions in the REGISTER TABLE command. Check the syntax of the REGISTER TABLE command and re-issue the command.

Generic Error Code and Message	GWF Status Code and Description
	E_GW54A2_INVALID_INTEGER An invalid value was provided for the offset, length, or scale of one of the RMS field definitions in the REGISTER TABLE command. Check the syntax of the REGISTER TABLE command and re-issue the command.
40203 E_GE9D0B_DATAEX_NUMOVR Data exception; exact numeric data, loss of significance (decimal overflow)	E_GW7015_DECIMAL_OVF Packed decimal overflow occurred when attempting to convert value to a precision <i>xxx scale xxx</i> packed decimal value in the RMS gateway.
40206 E_GE9D0E_DATAEX_DTINV Data exception; invalid datetime format	E_GW7003_DATE_OUT_OF_RANGE The OpenVMS date contains the year <i>xxx</i> that is out of the range of valid Ingres dates.
40206 E_GE9D0E_DATAEX_DTINV Data exception; invalid datetime format	E_GW7006_ING_TO_VMS_DATE Error occurred attempting to convert an Ingres date to an OpenVMS date. The Ingres date value was not a valid OpenVMS date (check range of valid dates in OpenVMS documentation).
	E_GW7011_BAD_VMSDATE_VAL An OpenVMS date value was passed to an OpenVMS system routine (SYS\$ASCTIM or SYS\$NUMTIM), which in turn reported an error. This indicates that the OpenVMS date was invalid. Be sure your offsets were correctly specified in the REGISTER command.
	E_GW7027_MNY_STR_TOO_LONG Attempt was made to convert a string longer than 2000 characters into a money value in the RMS gateway; this is not allowed.
	E_GW7028_DATE_IS_INTERVAL Attempt was made to convert an Ingres date interval into an OpenVMS date in the RMS gateway. Only absolute dates may be converted to OpenVMS dates.

Generic Error Code and Message	GWF Status Code and Description
	E_GW7029_YEAR_NOT_1900 Attempt was made to convert an Ingres date with year xxx to an integer in the RMS gateway. Only dates in the 20th century (that is, with year between 1900 and 1999 inclusive) may be converted to integer.
40220 E_GE9D1C_DATAEX_FIXOVR Data exception; fixed point overflow	E_GW7000_INTEGER_OVF Integer overflow occurred when attempting to convert value to an xxx byte.
40221 E_GE9D1D_DATAEX_EXPOVR Data exception; exponent overflow	E_GW7001_FLOAT_OVF Floating point overflow occurred when attempting to convert value to an xxx byte float in the RMS gateway.
40226 E_GE9D22_DATAEX_EPUNF Data exception; exponent underflow	E_GW7002_FLOAT_UND Floating point underflow occurred when attempting to convert value to an xxx byte float in the RMS gateway.
	E_GW7004_MONEY_OVF Money overflow occurred in the RMS gateway.
40228 E_GE9D24_DATAEX_OTHER Other unspecified math exception	E_GW7010_BAD_FLOAT_VAL The OpenVMS system routine OTS\$CNVOUT_x returned an error when given an RMS G_FLOAT or H_FLOAT value; be sure offsets are correctly specified in REGISTERed table.
	E_GW7012_BAD_STR_TO_INT Cannot convert xxx to integer in the RMS gateway.
40228 E_GE9D24_DATAEX_OTHER Other unspecified math exception	E_GW7013_BAD_NUM_TO_INT Error occurred attempting to convert a numeric string value (such as LEADING NUMERIC, UNSIGNED NUMERIC, ZONED NUMERIC, or OVERPUNCHED numeric) to an integer value in the RMS gateway. Be sure your offsets were correctly specified in the REGISTER command.

Generic Error Code and Message	GWF Status Code and Description
	E_GW7014_BAD_STR_TO_VDATE Error occurred attempting to convert xxx to an OpenVMS date using the OpenVMS system routine LIB\$CONVERT_DATE_STRING in the RMS gateway. Make sure your date format attributes are correctly set for the RMS data you are attempting to convert, and make sure your offsets are correctly specified in the REGISTER command.
	E_GW7016_BAD_STR_TO_DEC Error occurred when attempting to convert xxx to a packed decimal value in the RMS gateway.
	E_GW7020_BAD_STR_TO_FLT Error occurred when attempting to convert xxx to a floating point value in the RMS gateway.
40228 E_GE9D24_DATAEX_OTHER Other unspecified math exception	E_GW7021_BAD_NUM_TO_FLT Error occurred attempting to convert a numeric string value (such as LEADING NUMERIC, UNSIGNED NUMERIC, ZONED NUMERIC, or OVERPUNCHED numeric) to a floating point value in the RMS gateway. Be sure your offsets were correctly specified in the REGISTER command.
	E_GW7022_BAD_INT_TO_DATE Error occurred attempting to convert the integer xxx to an Ingres date value in the RMS gateway. Check the current setting of II_DATE_FORMAT to make sure it supports the correct ordering of days, months, and years which is intended; also be sure your offsets were correctly specified in the REGISTER command.

Generic Error Code and Message	GWF Status Code and Description
	<p>E_GW7023_BAD_STR_TO_DATE</p> <p>Error occurred attempting to convert xxx to an Ingres date value in the RMS gateway. Check the current setting of II_DATE_FORMAT to make sure it supports the correct date formats you are intending to use; also be sure your offsets were correctly specified in the REGISTER command.</p>
<p>40228</p> <p>E_GE9D24_DATAEX_OTHER</p> <p>Other unspecified math exception</p>	<p>E_GW7024_BAD_STR_TO_MNY</p> <p>Error occurred attempting to convert xxx to an Ingres money value in the RMS gateway. Check the current setting of the II_DECIMAL and II_MONEY_FORMAT logicals to ensure you have the proper money formats you are intending to use; also be sure your offsets were correctly specified in the REGISTER command.</p>
	<p>E_GW7025_BAD_NUM_TO_MNY</p> <p>Error occurred attempting to convert a numeric string value (such as LEADING NUMERIC, UNSIGNED NUMERIC, ZONED NUMERIC, or OVERPUNCHED numeric) to a money value in the RMS gateway. Be sure your offsets were correctly specified in the REGISTER command.</p>
	<p>E_GW7026_BAD_FLT_TO_MNY</p> <p>Internal error. ADF returned an error when attempting to convert a float value to a money value in the RMS gateway; this should never happen.</p>
	<p>E_GW702A_FLT_STR_TOO_LONG</p> <p>Attempt was made to convert a string longer than 2000 characters into a float value in the RMS gateway; this is not allowed.</p>

Appendix B: RMS Gateway Catalogs

Each RMS gateway database contains a set of catalogs that contain information about various aspects of the database. An Ingres catalog is simply a database table; each catalog has columns that describe a particular database object—a table, view, or index.

You can query an Ingres catalog as you would any Ingres table to retrieve information about the objects in a database. In addition, the Ingres query optimizer uses statistics contained in certain Ingres catalogs to formulate the most efficient query plans.

An RMS gateway database contains both standard Ingres catalogs and “extended” catalogs specific to the RMS gateway. This appendix describes:

- The Ingres catalogs that are created when you create an RMS gateway database
- The catalog entries that the gateway makes when you register RMS files and indexes

RMS Gateway Catalogs

When you create an RMS gateway database, Ingres creates the following catalogs in the database:

- A set of the standard Ingres system catalogs

These catalogs are described in the *Database Administrator Guide*.

The iirelation catalog, which contains a row for each table or view in a database, has two additional columns:

- relgwid(i2): contains the value “02” to indicate an RMS gateway table
- relgwother(i2): reserved for any additional gateway information

Also, the relstat column of the iirelations catalog is modified to hold a flag that indicates an RMS gateway table.

- A set of *extended* catalogs that contain RMS gateway-specific information; these are:
 - iigw02_relation: contains one row for each RMS file that you register in a database
 - iigw02_attribute: contains one row for each column of each registered RMS file
 - iigw02_index: contains one row for each RMS secondary index that you register in a database

The "02" in the catalog names is the Ingres identifier for the RMS gateway. The tables in the following section describe the columns in each of these catalogs.

In addition, entries are made for the new RMS gateway database in the catalogs in the Ingres master database, iidbdb. The entry contains a value in the dbservice column to inform Ingres that you have created an RMS gateway database.

RMS Gateway Extended Catalogs

As mentioned in the previous section, each RMS gateway database contains three extended catalogs that hold information about the registered tables and indexes in the database. The tables that follow describe the columns in each of these catalogs.

The following table lists the columns of the iigw02_relation catalog:

Column Name	Data Type	Description
xreltid	i4	First part of the Ingres table identifier for the registered RMS file
xreltidx	i4	Second part of the Ingres table identifier for the registered RMS file
file_name	char(255)	Directory path and name of a registered RMS file

The following table lists the columns of the iigw02_attribute catalog:

Column Name	Data Type	Description
xrelid	i4	First part of the Ingres table identifier for the registered RMS file
xrelidx	i4	Second part of the Ingres table identifier for the registered RMS file
xattid	i2	Number of the column in the Ingres table that corresponds to the field of the RMS file
gtype	i4	RMS data type
glength	i4	Length of the field
goffset	i4	Position of the field in the RMS file

Column Name	Data Type	Description
gprec_scale	i2	A 2-byte value that indicates precision (number of digits) and scale (decimal point placement) for the value in the RMS file. This value is relevant for numeric string and packed decimal data types.
gflags	i2	A flag value that indicates whether to use default or non-default data type mapping

The following table lists the columns of the iigw02_index catalog:

Column Name	Data Type	Description
xbaseid	i4	First part of the Ingres table identifier of the base RMS file for the index
xindexid	i4	Ingres index identifier for the RMS secondary index
key_ref	i2	Identifier indicating whether the index is keyed or sortkeyed

Catalog Entries for Registered RMS Files

When you register an RMS file as an Ingres table in an RMS gateway database, the following catalog entries are made:

- One row is added to the iirelation catalog for the new table; this entry is the same as for a newly-created Ingres table, except that the relstat column contains the RMS gateway indicator.
- One row is added to the iigw02_relation catalog for the new table; this row indicates the name and directory path of the RMS file source.
- A row is added to the iiattribute catalog for each column in the gateway table.
- A row is added to the iigw02_attribute catalog for each column in the gateway table; this row describes the corresponding RMS file field.

When you issue the remove table statement for a registered table, all the above entries are removed from the catalogs without affecting the underlying RMS data.

Catalog Entries for Registered RMS Indexes

When you register an RMS alternate key as a secondary index in an RMS gateway database, the following catalog entries are made:

- One row is added to the `iiindexes` catalog; this is similar to the record for an Ingres index created with the `create index` statement.
- One record is added to the `iigw02_index` catalog.

Appendix C: Installation and Configuration Notes

This appendix provides information on installing, configuring, starting and stopping the RMS gateway.

Installation Requirements

Before you install the RMS gateway, be sure that:

- The disk on which the existing Ingres installation resides contains at least 10,000 free blocks; for a new installation, add 10,000 blocks to the requirements for Ingres listed in the *Installation Guide*.
- You have made a recent backup of your system disk.

How the RMS Gateway Is Installed

To install the gateway, use `vminstal` to load the software and to run gateway setup procedures. For details on running `vminstal`, see the *Installation Guide*.

The `vminstal` procedure:

- Verifies that your system meets installation requirements.
- Loads all RMS gateway files from media into the Ingres installation.
- Executes the `iisurms.com` procedure that prompts for configuration information.

The next time you invoke `ingstart`, the RMS gateway server is also started.

Changing Log File Size

Ingres uses one transaction log file across an entire installation. This file is stored in the location you specify for `II_LOG_FILE` when you install Ingres.

The RMS gateway data manager does not log RMS data that you insert or update. Therefore, the RMS gateway does not appreciably increase the amount of space needed for the log file. If you want to enlarge the log file after you install the RMS gateway, an increase of 10% is probably sufficient.

See the *System Administrator Guide* for details about specifying the location and size of the log file.

Configuring Logging and Locking System Values

When you install Ingres, you specify values for parameters of the Ingres locking and logging systems. These systems handle logging, journaling, recovery, and locking coordination for your entire Ingres installation, including the RMS gateway.

The RMS gateway installation procedure does not prompt you for logging and locking system values; rather, it uses the values defined for the installation. You can reconfigure the logging and locking system to include the RMS gateway using Configuration-By-Forms (CBF).

The logging and locking systems apply to your entire Ingres installation. Therefore, when you add the RMS gateway, you may want to increase certain system parameters that are shared among all data managers.

In particular, you may need to increase the following values:

- Maximum number of open databases
- Maximum number of transactions
- Maximum number of locks

See the *System Administrator Guide* for additional information about configuring the locking and logging systems.

Be aware that you cannot recover RMS data through the RMS gateway. The Ingres logging system only logs updates to RMS gateway catalogs that describe registered RMS files and indexes. The logging system is not aware of changes to the actual data in the RMS files.

Initializing the RMS Gateway

After the RMS gateway is successfully installed, you can start it on your OpenVMS system as follows:

1. Shut down your Ingres installation by entering the following command:

```
$ ingstop
```

See the *System Administrator Guide* for more information about shutdown procedures. If you are running the RMS gateway on an OpenVMSCluster, you must shut down the installation on each node.

2. Restart the installation by entering the following command:

```
$ ingstart
```

You must execute this command for each node of an OpenVMSCluster.

The iistartup procedure creates a new process for the RMS gateway server. The process name has the form `II_RMS_xx_yyy_` where:

- xx is the test installation ID.
- yyy represents the last three digits of the RMS gateway server process ID.

The RMS gateway now is available to use on your system.

Configuring the RMS Gateway Data Manager

The RMS gateway configuration procedure (iisurms.com) configures the RMS gateway data manager (server) by choosing default values for various server parameters. Except as described below, the RMS gateway installation procedure does not prompt you for these values.

Specifying Server Parameter Values During Gateway Configuration

The RMS gateway configuration procedure prompts you for the following information used in configuring the RMS gateway data manager:

- Number of user sessions on the RMS gateway server
This value sets two parameters:
 - ACTIVE_SESSIONS (sessions actively competing for quanta—time slices—in the server)
 - CONNECTED_SESSIONS (total session connections available to the server)
- Number of databases that can be accessed by the RMS gateway server
This value sets the parameter DATABASE_COUNT (maximum number of databases accessible by the server).

The following table lists the necessary RMS startup parameters defined by the configuration utility.

Note: If you choose Express option for configuring the RMS gateway, you will not be prompted for these configuration parameters. Default values are used instead. Prior versions of the RMS gateway defined the configuration parameters as logicals. They are now maintained by CBF. When upgrading from prior releases, the logicals are automatically migrated to the CBF definition.

Configuration Parameter	Parm	Data Type	Default
II_RMS_FAB_SIZE	number	integer	16
II_RMS_HASH_SIZE	number	integer	7
II_RMS_RRL	on off	boolean	off

Configuration Parameter	Parm	Data Type	Default
II_RMS_READONLY	on off	boolean	off
II_RMS_BUFFER_RFA	on off	boolean	off
II_RMS_CLOSE_FILE	on off	boolean	off

The following table lists the RMS gateway-specific startup parameters definitions, all of which are defined in II_CONFIG:config.dat:

Symbol	Definition
II_RMS_FAB_SIZE	Defines the resource to cache RMS file information. Set this value to an estimate of the largest number of RMS files that will be open concurrently by the gateway.
II_RMS_HASH_SIZE	Defines the hash table size used for hashing RMS file information blocks. This value should be about one-half to one-third of the II_RMS_FAB_SIZE.
II_RMS_RRL	Instructs the RMS gateway server to read a record from the RMS file regardless of whether the record is locked by other sessions or applications.
II_RMS_READONLY	Defines the RMS gateway server as a read-only server. Once defined this way, all non-read activities are not allowed.
II_RMS_CLOSE_FILE	Lets server close a file when there is no session accessing this file. By default (OFF), a file stays open until server is taken down.
II_RMS_BUFFER_RFA	Informs the server that the Record File Address (RFA) associated with an RMS record should be buffered. This logical should be defined in conjunction with a positive value of II_RMS_READONLY.

When you run the RMS gateway installation procedure, you can accept the default values that the procedure provides or specify different values at each of the prompts.

Configuration Differences for RMS Gateway Data Manager

The RMS gateway currently does not contain Ingres multi-server capability. You can start only one RMS gateway server on any given node in an installation.

Starting and Stopping the RMS Gateway

The RMS gateway is available on your system whenever Ingres is running. To verify the presence of the RMS gateway server process, use the `iinamu` utility, as described in the *System Administrator Guide*. The gateway server process name begins with the string `II_RMS_`.

If the RMS gateway process is not present after Ingres startup, you can start it by entering the following command from the Ingres system administrator account:

```
$ ingstart -RMS
```

You can use the Ingres `iimonitor` utility to stop the RMS gateway without shutting down your entire Ingres installation. For details, see the *System Administrator Guide*.

Index

/

- /d database extension
 - accessing distributed databases, 5-4
 - creating distributed databases, 3-3
- /rms database extension, 1-2, 5-3
 - calling Ingres, 5-5
 - creating RMS gateway databases, 3-2
 - in connect statement, 5-3
 - in isql command, 4-2

A

- Access Control Lists, 5-1
- accessdb (utility), 7-1, 7-2
- ACTIVE_SESSIONS (server parameter), C-3
- Applications-by-Forms (ABF)
 - accessing through the RMS gateway, 5-5

B

- backup and recovery
 - RMS gateway databases, 7-4

C

- catalogdb (command), 7-2
- ckpdb (command), 7-5
- columns
 - data types of, 6-10
- commands
 - catalogdb, 7-2
 - ckpdb, 7-5
 - copydb, 7-2
 - createdb, 3-2, 7-2
 - definition, 1-5
 - destroydb, 3-4, 7-3

- isql, 4-2
- optimizedb, 7-3
- register, 6-10
- remove, 6-10
- rollforwarddb, 7-5
- rungrms, C-5
- sequence for updating system catalogs, 6-12
- statdump, 7-4
- sysmod, 7-4
- unloaddb, 7-2
- upgradedb, 7-4
- verifydb, 7-4

- connect (statement), 5-3
- CONNECTED_SESSIONS (server parameter), C-3
- copydb (command), 7-2
- create integrity (statement), 5-7
- createdb (command), 3-2, 7-2

D

- data manager
 - configuring, C-3
 - Ingres, 2-2
 - RMS gateway, 1-1
- data types, 4-12
 - columns, 6-10
 - converting, 4-9, 4-14
 - default conversions, 4-9
 - imaginary, 6-3
 - registration errors, 7-7
- database administrator (DBA)
 - granting privileges, 3-2, 7-1
 - responsibilities, 7-2
 - table ownership, 4-2
- database commands
 - RMS gateway, 6-9
- database management utilities
 - RMS gateway, 7-2

DATABASE_COUNT (server parameter), C-3

databases

- connecting, 5-2
- RMS gateway, 3-1

default conversions, 4-12

delete (statement), 5-6

destroydb (command), 3-4, 7-3

distributed databases

- accessing Ingres data, 5-6
- accessing RMS data, 5-4
- including RMS gateway databases, 3-3
- Ingres Star, 2-4

drop integrity (statement), 5-7

drop permit (statement), 7-1

E

embedded SQL

- accessing through the RMS gateway, 5-5
- connect statement, 5-3
- retrieving errors, 7-6

errlog.log (error log file), 7-7

errors

- gateway user, 7-6
- generic codes, A-1
- handling, 7-5
- log files, 7-7
- query, 7-8
- registration, 7-7

extended formats

- specifying in register table statement, 4-14

G

grant (statement), 7-1

I

II_LOG_FILE, C-1

iimonitor (utility), 7-3, C-5

iinamu (utility), 7-3

iirmsbuild (procedure), C-1

indexes

- naming, 3-2
- registering, 4-18
- removing registrations, 4-20
- system catalog entries, B-4

Ingres

- calling user interfaces, 5-3
- components, 2-2
- components supported by RMS gateway, 5-5
- data manager, 2-2
- errors, 7-5
- Ingres Star, 2-4
- locks, 5-8
- Net, 2-3
- overview, 2-1
- system utilities, 7-3
- tools, 2-2
- user interfaces, 2-2

Ingres Net

- accessing databases, 5-4

inquire_ingres (statement), 7-6

insert (statement), 5-6

integrity

- RMS files, 5-7

ISO Entry SQL92 compliance, 1-5

isql (command), 4-2

K

keys

- registering, 4-7
- RMS descending, 6-6

L

locking

- configuring, C-2

locks

- supported by RMS gateway, 5-8

lockstat (utility), 7-3

logging

 configuring, C-2

 file size, C-1

logicals

 in RMS file names, 4-7

logstat (utility), 7-3

N

Net

 access privileges, 5-2

netu (utility), 7-3

O

Object Management Extension, 5-8

OpenROAD

 accessing through the RMS gateway, 5-5

OpenSQL

 data type restrictions, 4-12

 RMS gateway, 2-6

 unsupported statements, 5-5, 7-1

OpenVMS privileges, 5-1

optimizedb (command), 7-3

options

 read-only server, 6-7

 read-regardless-lock, 6-6

 Record File Address (RFA), 6-8

P

performance

 RMS gateway applications, 5-10

privileges

 granting, 7-1

 OpenVMS, 5-1

Q

queries

 errors, 7-8

 repeat, 5-7

 Structured Query Language (SQL)

 required, 2-6

query languages, 1-5

query processing

 RMS gateway, 1-3

Query-By-Forms (QBF)

 accessing through the RMS gateway, 5-5

R

rcpconfig (utility), 7-3

record access

 multistream, 6-5

 sharing open files, 6-5

record size

 RMS gateway, 4-6

register (command), 6-10

register index (statement), 4-18

 example, 4-20

register table (statement), 4-3

 examples, 4-17

 specifying extended formats, 4-14

registration

 defining RMS data to gateway, 4-1

 errors, 7-7

 extended formats, 4-14

 performance tuning, 5-11

 primary keys, 4-7

 removing, 4-16, 4-20

 RMS files, 4-3

 RMS indexes, 4-18

 specifying row count, 4-9

remove (command), 6-10

remove index (statement), 4-20

remove table (statement), 4-16

repeat queries

 supported by RMS gateway, 5-7

-
- repeated (keyword), 5-7
 - repeating groups
 - access, 6-1
 - access characteristics, 6-4
 - read data from, 6-4
 - table registration, 6-3
 - usage guidelines, 6-2
 - Report-By-Forms (RBF)
 - accessing through the RMS gateway, 5-5
 - Report-Writer
 - accessing through the RMS gateway, 5-5
 - RMS files
 - access, 5-1
 - DECNet nodes in names, 4-7
 - integrity constraints, 5-7
 - locking, 5-8
 - logicals in names, 4-7
 - modifying storage structures, 5-6
 - protecting updates, 5-10
 - registering, 4-3
 - registering as multiple tables, 4-7
 - registering as updatable, 4-6
 - registering primary keys, 4-7
 - removing registrations, 4-16
 - specifying names in register table statement, 4-7
 - system catalog entries, B-3
 - updating through RMS gateway, 5-6
 - using utilities on tables with repeating groups, 6-11
 - RMS gateway, 1-2
 - accessing data, 2-5
 - accessing Ingres data, 5-6
 - and OpenSQL, 2-6
 - connecting to RMS database, 5-2
 - controlling database access, 7-1
 - creating databases, 3-2
 - data manager, 1-1
 - data type conversion, 4-9, 4-14
 - database backup and recovery, 7-4
 - database commands, 6-9
 - database management utilities, 7-2
 - database system catalogs, B-1
 - databases, 3-1
 - destroying databases, 3-4
 - environment, 2-1
 - error handling, 7-5
 - error log files, 7-7
 - errors, A-1
 - extended catalogs, B-2
 - including in distributed databases, 3-3
 - Ingres components supported, 5-5
 - Ingres sytem utilities, 7-3
 - initializing, C-2
 - installation requirements, C-1
 - integrity constraints, 5-7
 - locking, 5-8
 - naming databases, 3-2
 - Object Management Extension, 5-8
 - performance, 5-10
 - process name, C-2
 - query processing, 1-3
 - record size, 4-6
 - repeat queries supported, 5-7
 - repeating group access, 6-1
 - RMS file protections, 5-1
 - server parameters, C-3
 - SQL statements supported, 5-5
 - starting and stopping, C-5
 - system catalogs, B-1
 - transaction handling, 5-8
 - tuple size, 4-6
 - unsupported SQL statements, 5-6
 - rollforwarddb (command), 7-5
 - rows
 - specifying count in register table statement, 4-9
 - rungwrms (command), C-5
- ## S
- set journaling (statement), 7-5
 - set lockmode (statement), 5-9
 - SQL
 - statements supported by RMS gateway, 5-5
 - SQLCA (SQL Communications Area), 7-6
 - statdump (command), 7-4
 - statements
 - definition, 1-5
 - grant, 7-1
-

- inquire_ingres, 7-6
- register index, 4-18
- register table, 4-3, 4-14
- remove index, 4-20
- remove table, 4-16
- set journaling, 7-5
- whenever, 7-6

sysmod (command), 7-4

system catalogs, 3-1

- extended, B-2
- index entries, B-4
- RMS file entries, B-3
- RMS gateway, B-1
- updating, 6-12

T

tables

- access, 5-2
- accessing Ingres data, 5-6
- modifying storage structures, 5-6
- naming, 3-2
- ownership, 4-2, 5-2, 7-1
- privileges, 7-1
- registering, 4-3
- removing registrations, 4-16
- sharing, 4-2

Terminal Monitor

- accessing through the RMS gateway, 5-5
- calling, 4-2

TID (Tuple Identifier), 6-8

Time Zone Differential

- in data type conversion, 4-12

timeouts

- locking RMS files, 5-9
- setting value, 5-9

transaction handling

- RMS gateway, 5-8

Tuple Identifier (TID), 6-8

tuple size

- RMS gateway, 4-6

U

- u flag
 - in isql command, 4-2
- unloaddb (command), 7-2
- update (statement), 5-6
- updates
 - locking RMS files, 5-8
 - protecting RMS files, 5-10
 - specifying in register table statement, 4-6
- upgradedb (command), 7-4
- utilities
 - accessdb, 7-2
 - definition, 1-5
 - iimonitor, 7-3, C-5
 - iinamu, 7-3
 - lockstat, 7-3
 - logstat, 7-3
 - netutil, 7-3
 - rcpconfig, 7-3
 - suitability to RMS gateway tables, 6-11
 - when to run, 6-11

V

- verifydb (command), 7-4
- VIFRED (Visual-Forms-Editor)
 - accessing through the RMS gateway, 5-5
- Vision
 - accessing through the RMS gateway, 5-5
- Visual-Forms-Editor (VIFRED)
 - accessing through the RMS gateway, 5-5
- vmsinstal
 - OpenVMS installation procedure, C-1

W

- whenever (statement), 7-6

