

VAX Rdb/VMS

Introduction and Master Index

December 1990

This manual provides an overview of VAX Rdb/VMS, a general description of the VAX Rdb/VMS documentation set, a glossary of VAX Rdb/VMS terms, and a master index to the VAX Rdb/VMS manuals.

Revision/Update Information:	This manual is a revision and supersedes previous versions.
Operating System:	VMS
Software Version:	VAX Rdb/VMS Version 4.0

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation.

Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

Any software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license. No responsibility is assumed for the use or reliability of software or equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

Restricted Rights: Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

© Digital Equipment Corporation 1988, 1989, 1990.

All rights reserved.
Printed in U.S.A.

The Reader's Comments forms at the end of this document request your critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation: ALL-IN-1, CDD/Plus, DEC, DEC/CMS, DECdecision, DECdtm, DECforms, DECintact, DEC/MMS, DECnet, DECTp, DECTrace, DECwindows, MicroVAX, ULTRIX, UNIBUS, VAX, VAX ACMS, VAX Ada, VAX BASIC, VAX C, VAX CDD, VAXcluster, VAX COBOL, VAX DATATRIEVE, VAX DBMS, VAXELN, VAX FMS, VAX FORTRAN, VAX Pascal, VAX RALLY, VAX Rdb/ELN, VAX Rdb/VMS, VAX RMS, VAX SPM, VAXstation, VAX TEAMDATA, VIDA, VMS, VT, and the DIGITAL Logo.

The following are third-party trademarks:

DB2 and IBM are registered trademarks of International Business Machines.
MS-DOS is a registered trademark of Microsoft Corporation.
Macintosh is a trademark of Apple Computer, Inc.

This document is available in printed and online versions.

This document was prepared using VAX DOCUMENT, Version 1.2.

Contents

Preface	v
1 Rdb/VMS Overview	
1.1 Understanding Rdb/VMS Concepts, Features, and Terminology ..	1-2
1.1.1 Understanding Relational Concepts as Implemented in Rdb/VMS	1-2
1.1.2 Understanding Rdb/VMS Features	1-6
1.1.3 Understanding Differences in Relational Terminology	1-11
1.2 Accessing Online Help for Rdb/VMS	1-15
1.3 Setting Up the Sample Database	1-15
1.4 Accessing Rdb/VMS Using SQL	1-20
1.4.1 Using Interactive SQL	1-20
1.4.1.1 Accessing Interactive SQL	1-21
1.4.1.2 Retrieving Data Using SQL SELECT Statements	1-21
1.4.1.3 Using Indexes for Retrieving Data	1-22
1.4.1.4 Joining Data from Multiple Tables	1-23
1.4.1.5 Isolating Unique Data Values	1-24
1.4.2 Using SQL Statements in Programs	1-24
1.4.2.1 Using SQL Module Language	1-25
1.4.2.2 Using Precompiled SQL	1-25
1.4.2.3 Using Dynamic SQL	1-26
1.4.2.4 Using the SQL Online Program Examples	1-26
1.4.3 Accessing Rdb/VMS from Systems Other Than VMS Using SQL/Services	1-28
1.5 Accessing Rdb/VMS Using RDO, RDML, and RDBPRE	1-29
1.5.1 Introducing RDO, RDML, RDBPRE, and Other Interfaces ...	1-29
1.5.2 Using the Online Program Examples for RDO, RDML, and RDBPRE	1-30

1.6	Maintaining and Tuning Rdb/VMS Databases with RMU	1-31
1.7	Rdb/VMS Product Kits	1-33
1.8	Using Rdb/VMS with Other Digital Products	1-34

2 Rdb/VMS Documentation Directory

Glossary

A Rdb/VMS Master Index

Tables

1-1	Sample COLLEGES Table	1-3
1-2	CURRENT_SALARY View	1-3
1-3	Differences in Relational Terminology	1-11
1-4	Syntax Differences Between SQL and RDO	1-12
1-5	SQL Files to Create Sample Databases	1-19
1-6	Program File Types	1-28
1-7	Online Program Examples for RDBPRE and RDML	1-31
A-1	Abbreviations for the Manuals Included in the Master Index	A-1

Preface

The VAX Rdb/VMS software, often referred to as Rdb/VMS in this manual, is a general purpose database management system based on the relational data model.

Purpose of This Manual

This manual serves four functions. It provides:

- 1 An overview of Rdb/VMS terms, concepts, and components
- 2 A brief summary of the purpose and contents of the manuals in the Rdb/VMS documentation set
- 3 A glossary of Rdb/VMS terms
- 4 A master index for the Rdb/VMS manuals

Intended Audience

This book is intended for the data processing professional who wants to become acquainted with the components of VAX Rdb/VMS, the relational database system from Digital for VMS systems. You do not need to be an expert with relational databases before reading this book. However, you should be familiar with the VMS operating system and VAX RMS, a record management service, before reading this manual. If you are not familiar with the VMS operating system, refer to the VMS documentation set for more information.

Operating System Information

Information about the versions of the operating system and related software that are compatible with this version of Rdb/VMS is included in the Rdb/VMS media kit and the *VAX Rdb/VMS Installation Guide*.

For information on the compatibility of other software products with this version of Rdb/VMS, refer to the System Support Addendum (SSA) that comes with the Software Product Description (SPD). You can use the SPD/SSA to verify which versions of your operating system are compatible with this version of Rdb/VMS.

Structure

This manual consists of two chapters, a glossary, and a master index.

Chapter 1	Introduces Rdb/VMS terms, concepts, and components.
Chapter 2	Summarizes the purpose and contents of each Rdb/VMS manual.
Glossary	Defines terms used with Rdb/VMS.
Appendix A	Contains a master index for the Rdb/VMS documentation set.

Conventions

In examples, an implied carriage return occurs at the end of each line, unless otherwise noted. You must press the RETURN key at the end of a line of input.

This section explains the conventions used in this manual:

.	A vertical ellipsis in an example means that information not directly related to the example has been omitted.
...	A horizontal ellipsis in statements or commands means that parts of the statement or command not directly related to the example have been omitted.
e, f, t	Index entries in the printed manual may have a lowercase e, f, or t following the page number; the e, f, or t is a reference to the example, figure, or table, respectively, on that page.
Color	In printed manuals, color in examples shows user input.
\$	The dollar sign represents the DIGITAL Command Language prompt. This symbol indicates that the DCL interpreter is ready for input.

References to Products

The Rdb/VMS documentation to which this document belongs often refers to products by their abbreviated names:

- DEC RdbExpert software is referred to as RdbExpert.
- DECdecision software is referred to as DECdecision.
- DECtrace software for VMS is referred to as DECtrace.
- The SQL interface to VAX Rdb/VMS is referred to as SQL. The SQL interface is Digital Equipment Corporation's implementation of the SQL standard ANSI X3.135-1989, ISO 9075:1989, commonly referred to as ANSI/ISO.
- VAX ACMS software is referred to as ACMS.
- VAX BASIC software is referred to as BASIC.
- VAX C software is referred to as C.
- VAX CDD/Plus software is referred to as CDD/Plus, the data dictionary, and the dictionary.
- VAX COBOL software is referred to as COBOL.
- VAX Data Distributor software is referred to as Data Distributor.
- VAX DATATRIEVE software is referred to as DATATRIEVE.
- VAX FORTRAN software is referred to as FORTRAN.
- VAX Pascal software is referred to as Pascal.
- VAX PL/I software is referred to as PL/I.
- VAX RALLY software is referred to as RALLY.
- VAX Rdb/ELN software is referred to as Rdb/ELN.
- VAX Rdb/VMS software is referred to as Rdb/VMS. Version 4.0 of VAX Rdb/VMS software is often referred to as V4.0.
- VAX TDMS software is referred to as TDMS.
- VAX TEAMDATA software is referred to as TEAMDATA.



Rdb/VMS Overview

Rdb/VMS is the Digital relational database management product that supports a full set of utilities and data definition and manipulation languages that let you create, query, and maintain your Rdb/VMS databases.

You can use Rdb/VMS for interactive queries or from within application programs. One advantage of a relational database is the ease with which you can retrieve precisely the information you want, even if you must gather the information from data stored in different places. You can access an Rdb/VMS database using any of the interfaces provided in your Rdb/VMS kits. Additionally, there are a number of other Digital products (discussed later in this chapter) through which you can access an Rdb/VMS database.

One of the interfaces supplied with your Rdb/VMS kit is SQL (structured query language), an industry-standard data definition and data manipulation language for relational databases. The SQL interface is Digital Equipment Corporation's implementation of the SQL standard ANSI X3.135-1989, ISO 9075:1989, commonly referred to as ANSI/ISO. Digital recommends that you use SQL to access your Rdb/VMS database.

Using SQL, you can design and create an Rdb/VMS database, load it with data, and read and update both data and data definitions. You can use SQL interactively or in application programs. SQL provides both a module language processor that lets you use SQL with any language that supports the VAX Procedure Calling Standard, whether or not there is a precompiler for that language, and a precompiler that supports VAX Ada, VAX C, VAX COBOL, VAX FORTRAN, VAX Pascal, and VAX PL/I.

When using SQL module language, you put SQL statements in a separate module language file which is called by your application program. SQL and the other Rdb/VMS interfaces are discussed in Section 1.4 and Section 1.5.

For additional introductory information, you can run an online demonstration procedure that introduces Rdb/VMS and shows how to create, use, and maintain a simple database. Run the online demonstration by entering:

```
$ @RDM$DEMO:DEMO
```

This chapter discusses the following topics:

- Relational concepts, Rdb/VMS features, and terminology
- Accessing online help
- Setting up the sample databases
- Accessing Rdb/VMS using SQL
- Accessing Rdb/VMS using other interfaces
- The online program examples for SQL and other interfaces
- Maintaining and tuning Rdb/VMS databases with the Rdb/VMS Management Utility (RMU)
- Rdb/VMS product kits
- Using Rdb/VMS with other Digital products

1.1 Understanding Rdb/VMS Concepts, Features, and Terminology

Before you create an Rdb/VMS database, you need to understand the relational model of database management systems, the Rdb/VMS features available to help you create, access, and maintain your database, and the differences in terminology among the different interfaces. This section discusses these topics.

1.1.1 Understanding Relational Concepts as Implemented in Rdb/VMS

In a relational database, data is organized in tables (or relations). Each **table** is made up of columns and rows. The **columns** are equivalent to data elements, or fields. The **rows** are equivalent to records; they indicate occurrences of column values in the table. If you are familiar with manipulating data in separate data files, you may think of tables as being roughly equivalent to data files. Columns and rows in a table are analogous to fields and records in a data file.

For example, Table 1-1 shows the five columns of the COLLEGES table of the Rdb/VMS sample personnel database, along with the first four rows (record occurrences). AU begins the American University row; BATE begins the Bates College row, and so forth.

Table 1-1 Sample COLLEGES Table

COLLEGE_CODE	COLLEGE_NAME	CITY	STATE	POSTAL_CODE
AU	American University	Washington	DC	20016
BATE	Bates College	Lewiston	ME	04240
BOWD	Bowdoin College	Brunswick	ME	04011
CALT	Cal. Institute of Tech.	Pasadena	CA	91125

Not all tables contain data that is physically stored in the database. A **view** is a logical table whose data is not physically stored in the database. You would access a view just as you would access a table whose data is physically stored. Views provide a way to define database access to fit the needs and access rights of particular users. You can include in your view definition any combination of columns and rows from other tables to tailor the database to suit your needs.

Table 1-2 presents the **CURRENT_SALARY** view from the sample personnel database. Columns from different tables comprise the columns of the **CURRENT_SALARY** view. For example, the columns **LAST_NAME**, **FIRST_NAME**, and **EMPLOYEE_ID** are from the **EMPLOYEES** table; and the **SALARY_START** and **SALARY_AMOUNT** columns are from the **SALARY_HISTORY** table.

Table 1-2 CURRENT_SALARY View

LAST_NAME	FIRST_NAME	EMPLOYEE_ID	SALARY_START	SALARY_AMOUNT
Toliver	Alvin	00164	14-JAN-1983	\$51,712.00
Smith	Terry	00165	1-JUL-1982	\$11,676.00

Tables and views are defined in the schema for the database. A **schema** contains the data definitions for a database. The data definitions include the tables, views, constraints, indexes, domains, triggers, and, for multifile databases, storage areas and (optionally) storage maps. (In a single-file database, the data definitions and the actual data are in the same file.) The **SQL CREATE SCHEMA** statement actually creates the database that is based on the specified schema definitions. You can easily modify schema definitions in **Rdb/VMS** as required by your application needs.

Constraints are conditions that restrict what values can be stored in a table. When you insert and update columns, the constraint checks the values you attempt to insert against the conditions specified by the constraint. For example, a constraint could ensure that the value entered for the **SEX** column would be either M or F. If a value violates the conditions set by the constraint, SQL generates an error message. You specify constraints in SQL **CREATE TABLE** and SQL **ALTER TABLE** statements.

Table-specific constraints are a type of constraint that prevent changes from being made that would violate the referential integrity of the database. For instance, a table-specific constraint could prevent the deletion of a row from the **EMPLOYEES** table if rows still exist in the **JOB_HISTORY** table for that employee.

Triggers are used to ensure that if you make a change to a table, appropriate changes will be made in other tables throughout the database. For example, you can define a trigger to specify that before a row is deleted from the **EMPLOYEES** table, all records for that employee would be deleted from other tables as appropriate. (You could store employee information about to be deleted in an archival database before the triggered action takes place.)

A **domain** can be thought of as a template for one or more table column definitions. The SQL **CREATE DOMAIN** statement associates a data type, and optionally other characteristics, with a domain name. Subsequent SQL **CREATE TABLE** statements use the domain definitions in their column definitions. User-defined domains let you standardize information with common characteristics so that the same elements are not defined differently in different tables.

For example, the domain **ID_DOM** is used for the **EMPLOYEE_ID** column in several tables and for the **SUPERVISOR_ID** column in the **JOB_HISTORY** table; the domain **LAST_NAME_DOM** is used for the **LAST_NAME** column in the **EMPLOYEES** and **CANDIDATES** tables, and the **DATE_DOM** domain is used for date columns in the **EMPLOYEES** and **JOB_HISTORY** tables. The following example shows some domain definitions and indicates which table uses those domains:

```
SQL> CREATE DOMAIN ID_DOM CHAR(5);
SQL> !
SQL> CREATE DOMAIN LAST_NAME_DOM CHAR(14);
SQL> !
SQL> CREATE DOMAIN FIRST_NAME_DOM CHAR(10);
SQL> !
SQL> CREATE DOMAIN DATE_DOM DATE;
SQL> !
SQL> SHOW TABLE EMPLOYEES;
```


Column Name	Data Type	Domain
EMPLOYEE_ID	CHAR(5)	ID_DOM
LAST_NAME	CHAR(14)	LAST_NAME_DOM
FIRST_NAME	CHAR(10)	FIRST_NAME_DOM

BIRTHDAY	DATE	DATE_DOM
----------	------	----------

SQL> SHOW TABLE JOB_HISTORY;

Column Name	Data Type	Domain
EMPLOYEE_ID	CHAR(5)	ID_DOM
JOB_START	DATE	DATE_DOM
JOB_END	DATE	DATE_DOM

SUPERVISOR_ID	CHAR(5)	ID_DOM
---------------	---------	--------

SQL> SHOW TABLE CANDIDATES;

Column Name	Data Type	Domain
LAST_NAME	CHAR(14)	LAST_NAME_DOM
FIRST_NAME	CHAR(10)	FIRST_NAME_DOM

Lists (SQL) or segmented strings (RDO) are special data types for the storage and retrieval of unstructured data. Examples of unstructured data include the following formats:

- Large amounts of text
- Long strings of binary input from a data collecting device
- Graphics data

In Rdb/VMS, this data is stored in unstructured bytes and divided into segments no larger than 64K bytes. Except for the length of these segments, Rdb/VMS does not know anything about the type of data contained in these data segments.

A transaction is a set of operations on a database that must complete as a unit or not complete at all. A transaction lets you defer the action of making permanent (committing) database changes until you are sure that all statements in a unit have executed successfully and have done what you expected them to do. Because a transaction groups SQL statements into a unit, you can apply (commit) or undo (roll back) the changes made by the statements within the transaction. For example, you can defer committing the changes (making the changes permanent) until you are sure that all update operations have been made correctly.

A distributed transaction groups more than one database or more than one database attachment into one transaction, even if the databases are located on different nodes. You apply changes to more than one database within one transaction; Rdb/VMS ensures that transactions involving changes to multiple databases are handled as a unit by using a two-phase commit protocol.

1.1.2 Understanding Rdb/VMS Features

Once you understand the basic structure of an Rdb/VMS database, you need to be aware of the varied features that let you create, use, and maintain the database. Rdb/VMS is a full-feature relational database system that offers:

- **Data independence and consistency**

You can remove data definitions from application programs and store them in the database with the data. Because Rdb/VMS reduces the storing of redundant data, it helps ensure that updates do not result in inconsistent data. Rdb/VMS also lets you centralize the management of data definitions, both within the database file and within the data dictionary (VAX CDD/Plus). You can use views to manage data stored in separate tables in the database. Table-specific constraints and triggers can be used to ensure that changed data remains consistent throughout the database.

- **Data integrity**

Rdb/VMS maintains the integrity of the database in the event of user errors, hardware or software failures, and concurrent use of the database. A before-image journal (also called recovery-unit journal (RUJ)) contains copies of records before they were updated. Rdb/VMS uses the before-image journal to automatically undo updates to a database when a rollback or system failure occurs. An after-image journal (AIJ) contains copies of rows after they have been updated. You can use the after-image journal to reconstruct a backup database to include the last successfully completed transaction. Integrity constraints ensure that data remains correct even when users try to modify it incorrectly.

- **Interactive, multiuser environment**

You can have access to the data at the same time as other users, yet each user can work from a customized view that may include only a subset of the entire database. Rdb/VMS also ensures that your operations on the database do not lead to inaccurate results for other users. Locking mechanisms ensure the maintenance of your data's integrity and consistency. Rdb/VMS uses snapshot files to ensure that you read a consistent view of the data even while other users are updating the database.

- **Data security**

Rdb/VMS complies with the C2 class security requirements specified by the National Computer Security Center (NCSC) in the Department of Defense (DoD) Trusted Computer System Evaluation Criteria, (the *Orange Book*). C2 class security requirements include:

- **Discretionary protection ("need-to-know protection")**

The creator of a database object can control the protection on that object.

- **User identification and authentication**

Rdb/VMS allows use of the database by authenticated VMS users.

- **Resource isolation**

This feature protects database objects from interference by non-database processes.

- **Accountability**

Rdb/VMS audits user actions that affect the security of the database.

Rdb/VMS implements C2 class security through a number of database security mechanisms:

- **Access privilege sets**

You can use an access privilege set to limit database object access to particular users and to specify a list of privileges for those users.

- **Role-oriented privileges**

You can designate users with role-oriented privileges to perform necessary tasks such as a backup operation without compromising the security of the data.

- **Security audit logging**

You can audit events that affect the security of the database, and review the audit records generated by those events, by using facilities provided by RMU.

- **Optimized data access**

Rdb/VMS includes a query optimizer that automatically analyzes each query to determine the most efficient method of access to the data. Because of the query optimizer, you do not have to be overly concerned with how you construct your queries.

- **Ability to create multifile databases**

You can define multiple files for one database, and you can distribute the individual files across many disks on your VAX system or VAXcluster configuration. You can create either a single-file or multifile Rdb/VMS database. In a single-file database, all user-stored data and internal Rdb/VMS data resides in a single database root file with the default file type RDB. A single-file database also includes a snapshot file, which provides temporary storage for read-only data retrieval and has a default file type SNP. By default, both files are located on the same device and in the same directory. However, these locations can be changed.

A multifile database includes one or more user-defined storage-area files in which user-stored data from one or more tables or indexes will reside. A multifile database consists of a database root file (file type RDB), one or more storage area files (file type RDA), and one or more snapshot files (file type SNP). One snapshot file is created for each storage area file. These files can be placed on different devices or in different directories.

- **Choice of interfaces to the database**

You can access an Rdb/VMS database through various supplied interfaces. These interfaces include:

- SQL (structured query language)
- SQL module language
- SQL precompiler
- RDO (Relational Database Operator)
- RDML (Relational Data Manipulation Language)
- RDBPRE preprocessor

SQL is used in most examples in the Rdb/VMS documentation set. However, the Rdb/VMS documentation set includes reference manuals for RDO and RDML, as well as a guide to using RDO, RDML, and RDBPRE.

Note *You can use either the SQL or the RDO interface to work with your databases, regardless of which interface was used to create them.*

- **Access from systems other than VMS**

You can access an Rdb/VMS database from various platforms, such as VMS, MS-DOS, and ULTRIX, with SQL/Services, a client/server component of Rdb/VMS that provides a callable routine interface similar to dynamic SQL. (**Dynamic SQL** is a set of SQL statements and data structures that allows programs to formulate and execute SQL statements at run time.) Application programs call routines in the SQL/Services Application Programming Interface (API) library. The API library communicates through DECnet with a server process on the VMS system on which the Rdb/VMS database resides.

- **Operations involving multiple databases**

You can start a distributed transaction that combines data from more than one database in a single query or update operation. These databases need not be on the same node. Your application can attach to more than one database using authorization identifiers to uniquely identify each database. If the distributed transaction makes changes to more than one database, Rdb/VMS uses a two-phase commit protocol to ensure that all changes are either committed or rolled back.

- **Ability to use read-only storage areas on a compact disk**

You can store an Rdb/VMS read-only storage area on a compact disk. Compact disk read-only memory (CDROM) software can be used to permit fast and efficient online access to the data. Use of a compact disk enhances online retrieval and conserves storage space.

- **Use of indexes for quick data retrieval**

You can locate particular records based on key column values by using an index. Sorted and hashed indexes both allow for quick and efficient data retrieval based on index keys. Sorted indexes can be used to retrieve a range of values for a column; hashed indexes are best used to retrieve only a specific value. Hashed indexes can be used only with multifile databases. Sorted indexes can be used with single-file and multifile databases.

- **Language-Sensitive Editor (LSE) templates for SQL statements**

You can use the VAX Language-Sensitive Editor (LSE) templates to code SQL modules for incorporation in application programs. LSE expands the SQL statements to provide syntax for you, letting you simply fill in the blanks in the supplied language template. Templates are not available for RDO, RDBPRE, or RDML.

- **Distributed databases**

You can access an Rdb/VMS database on a remote node by supplying the node name with the file specification for the database. You can also use VAX Data Distributor to make complete or partial copies of your Rdb/VMS databases on remote nodes. A replication database would be periodically updated, an extraction database would not.

- **Centralized database management**

A single user can handle the responsibility of administering the database. This centralization of database administration tasks helps ensure the consistency of the database. A set of simple RMU commands and SQL or RDO statements make database maintenance and control easy.

- **Control over disk space management**

You can place the database root file, data files, snapshot files, journal files, and backup files on different disks to ensure both the availability of space and the preservation of the database in the event of a disk failure.

- **Maintenance and performance tools**

Rdb/VMS supplies a variety of maintenance and performance tools for you to use. The Rdb/VMS Management Utility (RMU) lets you perform the following functions:

- Back up and restore the database either on a database-wide level or by area
- Open and close the database
- Unload specific tables and views into files
- Reload data from these files or from record management services (RMS) files into a specific table
- Recover the database from the after-image journal file
- Analyze the efficiency of retrieval through indexes
- Display database statistics
- Verify the integrity of the database

- **VAXcluster support, including full journaling and recovery**

Rdb/VMS is fully operational in a VAXcluster environment.

- **Full online help facilities**

Online help is available through the DIGITAL Command Language (DCL) level as well as through the interactive Rdb/VMS utilities. See Section 1.2 for more information about accessing online help for Rdb/VMS.

1.1.3 Understanding Differences in Relational Terminology

As Section 1.1.2 presents, there are several interfaces to choose from when using Rdb/VMS. In general, terminology for relational database systems varies from system to system and from interface to interface. Thus, the Rdb/VMS interfaces use different terms to mean the same thing. For instance, some terms used by SQL differ from terms used by other interfaces, such as RDO or RDML.

Table 1-3 shows some of the terms used to describe basic relational database concepts. For example, when you use documentation pertaining to SQL, you will encounter the terms table, column, and row. However, when you use documentation pertaining to RDO or RDML, you will encounter the terms relation, field, and record. This manual primarily uses SQL terminology.

Table 1-3 Differences in Relational Terminology

SQL	RDO, RDML	ANSI/ISO SQL STANDARD
Alias	Context variable	Alias
Authorization identifier	Database handle	
Cartesian product	Cross product	Cartesian product
Column	Field	Column
Column select expression	Record selection expression	Column select expression
Consistency level 2	Concurrency	Consistency level 2 (SQL2) ¹
Consistency level 3	Consistency	Consistency level 3 (SQL2) ¹
Context files	N/A ²	N/A ²
Domain	Global field	Domain (SQL2) ¹
List	Segmented String	
Parameter	Host language variable	Parameter
Predicate	Conditional expression	Predicate
READ ONLY	READ_ONLY	READ ONLY
READ WRITE	READ_WRITE	READ WRITE
Result table	Record stream	Result table
Row	Record	Row

¹SQL2 is the industry standard currently being developed.

²N/A means that the term is not applicable or not used with the listed product, standard, or system.

(continued on next page)

Table 1-3 (Cont.) Differences in Relational Terminology

SQL	RDO, RDML	ANSI/ISO SQL STANDARD
Storage area	Storage area	N/A ²
Storage map	Storage map	N/A ²
Table	Relation	Table

²N/A means that the term is not applicable or not used with the listed product, standard, or system.

Just as different terms mean the same thing between RDO and SQL, different statements do the same thing. Table 1-4 presents the SQL data definition and manipulation statements and supplies the RDO equivalents.

Table 1-4 Syntax Differences Between SQL and RDO

SQL Statement	RDO Statement
ALTER DOMAIN	CHANGE FIELD
ALTER INDEX	CHANGE INDEX
ALTER SCHEMA	CHANGE DATABASE
ALTER STORAGE MAP	CHANGE STORAGE MAP
ALTER TABLE	CHANGE RELATION
BEGIN DECLARE	No equivalent
CLOSE	END_STREAM
CLOSE LIST CURSOR	END_SEGMENTED_STRING
COMMENT ON	DESCRIPTION IS
COMMIT	COMMIT
CREATE COLLATING SEQUENCE	DEFINE COLLATING_SEQUENCE
CREATE DOMAIN	DEFINE FIELD
CREATE INDEX	DEFINE INDEX
CREATE SCHEMA	DEFINE DATABASE
CREATE STORAGE AREA clause	DEFINE STORAGE_AREA clause
CREATE STORAGE MAP	DEFINE STORAGE MAP
CREATE TABLE	DEFINE RELATION
CREATE TRIGGER	DEFINE TRIGGER

(continued on next page)

Table 1-4 (Cont.) Syntax Differences Between SQL and RDO

SQL Statement	RDO Statement
CREATE VIEW	DEFINE VIEW
DCL Invoke (\$)	DCL Invoke (\$)
DECLARE CURSOR	START_STREAM and DECLARE_STREAM
DECLARE SCHEMA	INVOKE DATABASE
DECLARE STATEMENT	No equivalent
DECLARE TABLE	No equivalent
DECLARE TRANSACTION	START_TRANSACTION
DELETE	ERASE
DESCRIBE	No equivalent
DROP COLLATING SEQUENCE	DELETE COLLATING_SEQUENCE
DROP CONSTRAINT	DELETE CONSTRAINT
DROP DOMAIN	DELETE FIELD
DROP INDEX	DELETE INDEX
DROP PATHNAME	DELETE PATHNAME
DROP SCHEMA	DELETE DATABASE
DROP STORAGE MAP	DELETE STORAGE MAP
DROP TABLE	DELETE RELATION
DROP TRIGGER	DELETE TRIGGER
DROP VIEW	DELETE VIEW
EDIT	EDIT
END DECLARE	No equivalent
Execute @	Execute @
EXECUTE	No equivalent
EXECUTE IMMEDIATE	No equivalent
EXIT	EXIT
EXPORT	EXPORT
FETCH	FETCH
FINISH	FINISH
GRANT	DEFINE PROTECTION
GRANT ANSI style	No equivalent

(continued on next page)

Table 1-4 (Cont.) Syntax Differences Between SQL and RDO

SQL Statement	RDO Statement
HELP	HELP
IMPORT	IMPORT
INCLUDE	No equivalent
INSERT	STORE
INTEGRATE	INTEGRATE DATABASE
OPEN CURSOR	START_STREAM
OPEN INSERT ONLY LIST CURSOR	CREATE_SEGMENTED_STRING
OPEN READ ONLY LIST CURSOR	START_SEGMENTED_STRING
PREPARE	No equivalent
PRINT	PRINT
QUIT	EXIT
RELEASE	No equivalent
REVOKE	DELETE PROTECTION
REVOKE ANSI style	No equivalent
ROLLBACK	ROLLBACK
SELECT	FOR-GET-END_FOR or FOR-PRINT-END_FOR
SET	SET
SET ALL CONSTRAINTS	No equivalent
SET TRANSACTION	START_TRANSACTION
SHOW	SHOW
UPDATE	MODIFY
WHENEVER	ON ERROR
No equivalent	ANALYZE
No equivalent	CLOSE
No equivalent	OPEN
No equivalent	PLACE
Use CREATE TABLE	DEFINE CONSTRAINT

1.2 Accessing Online Help for Rdb/VMS

Complete online help is available for Rdb/VMS. From the DCL level, you can access help for Rdb/VMS, SQL, RDO, RDML, and RMU. For example, to access a general help screen for Rdb/VMS, enter the following command at the DCL prompt:

```
$ HELP RDBVMS
```

(Note the absence of the slash between Rdb and VMS.)

Entering a question mark at the topic prompt will cause DCL to display a list of all topics included under RDBVMS help.

Additionally, more detailed help is available from within the interactive interfaces, SQL and RDO. To access this help, enter the interactive interface of your choice, and then enter help:

```
$ SQL  
SQL> HELP
```

Each primary HELP command displays a list of topics on which information is available. At the topic prompt, enter one of the listed subjects to display information, and, in some cases, a list of subtopics.

You may prefer to access a specific topic or subtopic with one simple HELP command. For example, to access the syntax diagram for the SQL CREATE SCHEMA statement, you can enter:

```
SQL> HELP CREATE SCHEMA
```

You can exit help by either pressing the RETURN key until you reach the prompt from which you first entered the HELP command, or by pressing CTRL/Z at any point.

1.3 Setting Up the Sample Database

To help you learn and test Rdb/VMS features, Rdb/VMS provides files to create the various forms of the sample personnel database: PERSONNEL, a single-file database, and MF_PERSONNEL, a multfile database. These forms can be built using either SQL or RDO data definitions. Additionally, you can choose whether or not to store data definitions in the data dictionary.

This section guides you through the process of creating your own copy of the sample Rdb/VMS databases. Many examples in the documentation are based on the sample personnel database. You create copies of the personnel database

by executing the DCL command procedure PERSONNEL.COM located in the directory RDM\$DEMO. By default, this command procedure builds the single-file PERSONNEL database using SQL data definitions. A prompt appears regarding use of the data dictionary:

```
$ @RDM$DEMO:PERSONNEL
```

This command file builds the VAX Rdb/VMS sample database, PERSONNEL. It requires approximately 2900 blocks of disk space.

You may enter the name of a VMS directory in which to create the database or use your current default directory. You can also have the database definitions placed in the common data dictionary if VAX CDD/Plus is installed on your system. In this case you must specify a VAX CDD/Plus directory in which to create the definitions.

NOTE: If a copy of PERSONNEL already exists in the location you specify, this command procedure will stop.

Do you wish to use CDD/Plus? (Yes or No or Exit) NO

Enter VMS directory specification for the database files

Directory; default is DISK1:[JONES.TEMP]:

>> Creating database in specified directory

Creating domains for the sample database

Creating tables for the sample database

Creating views for the sample database

Adding comments for domain and table definitions

Storing sample data in several tables. (Other tables to be loaded later by separate programs.)

1 row inserted

1 row inserted

1 row inserted

1 row inserted

1 row inserted

1 row inserted

>> Loading PERSONNEL database from data files

Program: Loading EMPLOYEES

Program: EMPLOYEES Loaded. Normal End-of-Job

Program: Loading JOBS

Program: JOBS Loaded. Normal End-of-Job

Program: Loading DEPARTMENTS

Program: DEPARTMENTS Loaded. Normal End-of-Job

Program: Loading JOB_HISTORY

Program: JOB_HISTORY Loaded. Normal End-of-Job

Program: Loading SALARY_HISTORY

Program: SALARY_HISTORY Loaded. Normal End-of-Job

Program: Loading COLLEGES

Program: COLLEGES Loaded. Normal End-of-Job

Program: Loading DEGREES

Program: DEGREES Loaded. Normal End-of-Job

```
Loading RESUMES table
1 row inserted
1 row inserted
1 row inserted
```

```
Creating remaining sorted indexes for MF_PERSONNEL
```

```
Creating triggers for the sample database
```

```
>> End of command procedure.
```

```
>> Examine the log file DISK1:[JONES.TEMP]PERSONNEL.LOG for the
>> database definitions and possible errors which occurred during the
>> database creation.
```

```
>>
```

```
>> The RDM$DEMO directory contains all the files used in building this
>> sample database.
```

If the command procedure executes successfully, you will find the following files in the directory you specified:

- **PERSONNEL.RDB**

This is the file where data and data definitions (metadata) are stored. **Metadata** is data that is used to describe other data. Data definitions are sometimes referred to as metadata.

- **PERSONNEL.SNP**

This is the snapshot file that provides read-only access to the database. A snapshot file exists to improve database performance when there are many concurrent database users.

You can create alternative versions of the sample personnel database by using the three available parameters on the **PERSONNEL** command line. These parameters specify the following choices:

- 1 Use of either SQL or RDO as the data definition language. (Enter SQL or RDO.)
- 2 Creation of either a single-file or multifile personnel database. (Enter S or M.)
- 3 Whether or not to store data definitions in the data dictionary. (Enter CDD or NOCDD.)

To specify the second or third parameters, you must include any previous parameters.

The multifile version of the personnel database is **MF_PERSONNEL**. To build the **MF_PERSONNEL** database using SQL data definitions, enter the following command:

```
$ @RDM$DEMO:PERSONNEL SQL M NOCDD
```

This example will create the **MF_PERSONNEL** database using SQL. Data definitions will not be stored in the data dictionary.

The MF_PERSONNEL database consists of the following files:

- MF_PERSONNEL.RDB

This is the database root file where data definitions (metadata) are stored.

- Several storage area files (file type RDA)

Storage area files are where data is stored in a multifile database.

- Several snapshot files (file type SNP); one for each storage area file

Rdb/VMS uses snapshot files during read-only transactions to ensure each transaction a consistent view of the database, and to improve overall performance when there are many concurrent users of the database.

If you encountered an error while running the command procedure, check the file PERSONNEL.LOG. You may have entered a dictionary or VMS directory specification incorrectly. In this case, you may have produced an incomplete version of the PERSONNEL or MF_PERSONNEL database. To remedy the situation, delete any data dictionary node or database files the command procedure has created. (The command procedure first creates the data dictionary node, then creates the database files, and finally stores data in the database. Unless an error, such as a system failure, occurs when data is being stored, you will have to delete either a dictionary node or database files, but not both.)

You can delete a database by invoking interactive SQL and entering a DROP SCHEMA FILENAME statement that includes a file specification. If you need to delete the database files and a dictionary node, use the PATHNAME clause of the DROP SCHEMA statement. For the VMS file specification and dictionary path name specified in the following examples, you would substitute the VMS file specification and dictionary path name that identify your incomplete copy of the PERSONNEL database:

```
SQL> DROP SCHEMA PATHNAME SYS$COMMON:[CDDPLUS]D32T.FIELDMAN.PERSONNEL;  
SQL> DROP SCHEMA FILENAME DISK02:[FIELDMAN]PERSONNEL;
```

You can then run the PERSONNEL.COM command procedure again with a corrected path name or directory specification.

The PERSONNEL.COM command procedure consists of several smaller command files. These command files are located in the directory RDM\$DEMO. You might want to use these files as models in creating your databases. Table 1-5 lists the command files used to create the sample databases with the SQL interface. For information on the command files used with the RDO interface, see the *VAX Rdb/VMS Guide to Using RDO, RDBPRE, and RDML*.

Table 1-5 SQL Files to Create Sample Databases

File Name	Explanation
PERSONNEL.COM	<p>Builds a single-file or multifile version of the sample personnel database; you can use SQL or RDO definitions and you can define the database by file name or data dictionary path name.</p> <p>For the SQL single-file version, the procedure invokes SQL command files to create domains, tables, views, constraints, indexes, and triggers. In addition, it invokes programs to store most of the data.</p> <p>For the SQL multifile version, the procedure invokes SQL command files to create domains, tables, views, constraints, indexes (both sorted and hashed), and to spread tables and indexes across multiple files. In addition, it invokes programs to store most of the data.</p>
BUILDPERS.SQL	Creates domains, tables, views, and constraints, and stores data.
SQL_CREATE_STORAGE.SQL	Creates storage areas for the multifile database.
MF_BUILDPERS.SQL	Creates domains, tables, views, constraints, hashed indexes, storage maps, and one sorted index, and stores data.
INDEXES.SQL	Creates the indexes (sorted) for the single-file database.
MF_INDEXES.SQL	Creates the remaining sorted indexes for the multifile database.
TRIGGERS.SQL	Creates triggers.
LIST.SQL	Loads rows of the RESUMES table.

The definitions of domains, tables, views, constraints, and triggers are the same for the single-file and multifile versions of the sample database. The definitions for the multifile database also include hashed indexes, storage area files, and other structures associated with the multifile implementation.

Note *You can use either the SQL or the RDO interface to work with the resulting databases, regardless of whether the SQL or RDO command files were used to create them.*

There are, however, some differences between the databases of the same name depending on whether you used the SQL-based procedure or the RDO-based procedure to create the database. The significant differences are as follows:

- The SQL-based procedures define default values for certain domains, whereas the RDO procedures define missing values for the corresponding global fields. The *VAX Rdb/VMS Guide to Database Design and Definition* discusses the distinction between a default value and a missing value.
- The SQL-based procedures define constraints to enforce the referential integrity of the database as part of the CREATE TABLE statements. In the RDO-based procedures, such constraints are defined in separate DEFINE CONSTRAINT statements.

1.4 Accessing Rdb/VMS Using SQL

Once you have created the PERSONNEL or MF_PERSONNEL databases, you can practice accessing them using SQL, the primary interface to Rdb/VMS. Digital recommends use of SQL instead of the proprietary RDO, RDML, or RDBPRE interfaces. Using SQL, you can access Rdb/VMS databases in several ways:

- Interactively
- Through programs
 - SQL module language
 - Precompiled SQL
 - Dynamic SQL
- From systems other than VMS (using SQL/Services)

This section presents the various SQL interfaces and discusses the various sample online programs available with Rdb/VMS.

1.4.1 Using Interactive SQL

This section introduces interactive SQL and demonstrates some simple retrieval statements you can use. As you develop application programs, you can invoke interactive SQL to learn the SQL language and to test statements that you plan to include in a program. For database administration, you may prefer the convenience of interactive SQL for creating or changing database definitions and protection. You can also use interactive SQL for queries against the database. While using interactive SQL, you can access online help at any time for information about SQL syntax. See Section 1.2 for more information about accessing online help.

For more detailed information on interactive SQL, see the *VAX Rdb/VMS Guide to Using SQL*.

Note *Interactive SQL can be used by people who are not data processing professionals. However, you might prefer to prepare applications at your site that are specially tailored to meet user needs. Generally, products such as DECdecision, TEAMDATA, or applications developed using RALLY or traditional programming languages, are the recommended relational database interfaces for most end users.*

1.4.1.1 Accessing Interactive SQL To use interactive SQL, you must run the SQL executable image. Define a symbol at the DCL level and include the symbol in your LOGIN.COM file. Define the symbol SQL as a foreign command. This command lets you execute an image by entering only the symbol name. (The command is **foreign** because it is unknown to DCL.) In the following example, the symbol SQL is defined to execute the image SQL\$:

```
$ SQL ::= $SQL$
```

After you enter the symbol SQL, the SQL prompt (SQL>) indicates that you are at the SQL interface command level and can interactively enter SQL statements.

```
$ SQL  
SQL>
```

To exit interactive SQL, press CTRL/Z or type EXIT.

1.4.1.2 Retrieving Data Using SQL SELECT Statements Before you can access the database, you must establish a connection by attaching to it. Enter the following statement to attach to the sample PERSONNEL database (without using data dictionary definitions):

```
SQL> DECLARE SCHEMA FILENAME PERSONNEL;
```

If you are using the multifile MF_PERSONNEL database, specify MF_PERSONNEL rather than PERSONNEL on the command line.

To retrieve data from the database, you can use various combinations of SQL SELECT statements. (The SELECT statement that you use in programs varies slightly from the interactive SELECT statement. See the *VAX Rdb/VMS Guide to Using SQL* for details.) These SELECT statements create result

tables. A **result table** is the set of columns and rows that fits the criteria for a specific retrieval statement. For example, the following is a result table produced by a simple **SELECT** expression that retrieves employee identification (ID) numbers and last names from the **EMPLOYEES** table:

```
SQL> SELECT EMPLOYEE_ID, LAST_NAME FROM EMPLOYEES;
EMPLOYEE_ID  LAST_NAME
00164        Toliver
00165        Smith
00166        Dietrich
00167        Kilpatrick
.
.
.
```

1.4.1.3 Using Indexes for Retrieving Data A particularly efficient way to retrieve data is through the use of an index. A search for data based upon an indexed value is fast and efficient. For example, in the **EMPLOYEES** table the index is sorted by the **EMPLOYEE_ID** column. Therefore, use of an **EMPLOYEE_ID** column value will speed the retrieval of data from the **EMPLOYEES** table because the query optimizer will choose to use the index to locate the data. The following example demonstrates an **SQL SELECT** statement that retrieves employee records using the **EMPLOYEE_ID** column value 00164:

```
SQL> SELECT EMPLOYEE_ID, LAST_NAME, FIRST_NAME, ADDRESS_DATA_1,
cont> CITY, STATE, POSTAL_CODE
cont> FROM EMPLOYEES
cont> WHERE (EMPLOYEE_ID = "00164");
EMPLOYEE_ID  LAST_NAME  FIRST_NAME  ADDRESS_DATA_1
CITY         STATE      POSTAL_CODE
00164        Toliver    Alvin       146 Parnell Place
Chocorua     NH         03817

1 row selected
```

If necessary, you can also perform a search on a non-indexed record. Suppose you have an employee's last name, but not his or her employee ID number. You could perform the following search:

```
SQL> SELECT EMPLOYEE_ID, LAST_NAME
cont> FROM EMPLOYEES
cont> WHERE (LAST_NAME = "Toliver");
EMPLOYEE_ID  LAST_NAME
00164        Toliver

1 row selected
```

Rdb/VMS finds the information, but does not perform the search as efficiently as when an index has been defined on the column being searched.

1.4.1.4 Joining Data from Multiple Tables You can combine data from two or more different tables using retrieval statements. Such operations are called joins. **Joins** differ from views in that they operate only for a specific retrieval statement, as opposed to existing as a permanent logical table. The following SQL statement asks for the names and current departments of all employees, and illustrates the use of a join:

```
SQL> SELECT LAST_NAME,
cont>         FIRST_NAME,
cont>         DEPARTMENT_NAME
cont> FROM EMPLOYEES, DEPARTMENTS, JOB_HISTORY
cont> WHERE ((JOB_HISTORY.EMPLOYEE_ID = EMPLOYEES.EMPLOYEE_ID)
cont>         AND
cont>         (DEPARTMENTS.DEPARTMENT_CODE = JOB_HISTORY.DEPARTMENT_CODE)
cont>         AND (JOB_END IS NULL))
cont> ORDER BY LAST_NAME, FIRST_NAME, DEPARTMENT_NAME;
EMPLOYEES.LAST_NAME   EMPLOYEES.FIRST_NAME   DEPARTMENTS.DEPARTMENT_NAME
Ames                  Louie                   Northeastern US Sales
Andriola              Leslie                  Telecommunications Industries
Babbin                Joseph                  European Sales
Bartlett              Dean                    Southern U.S. Sales
Bartlett              Wes                     Corporate Sales
```

In the preceding statement, the columns `LAST_NAME`, `FIRST_NAME`, and `DEPARTMENT_NAME` are retrieved from the `EMPLOYEES` and `DEPARTMENTS` tables, respectively. The `EMPLOYEES` and `DEPARTMENTS` tables do not share a column. Therefore, the join includes the `JOB_HISTORY` table to determine the correct department name for each employee. As the `SELECT` statement indicates, the employee identification number in the `JOB_HISTORY` table must equal the employee identification number in the `EMPLOYEES` table, and the `DEPARTMENT_CODE` column value in the `DEPARTMENTS` table must equal the `DEPARTMENT_CODE` column value in the `JOB_HISTORY` table.

Note that the preceding example qualified table names when the same column was used from more than one table. In SQL, qualifying or identifying the table is necessary only when using multiple tables. To save typing, you can define

an alias to qualify the table from which a particular column is selected. (An **alias** is a qualifier for column names that is used in place of the table name in an SQL SELECT statement.) In the following example, the E after EMPLOYEES, the D after DEPARTMENTS, and the JH after JOB_HISTORY are all aliases:

```
SQL> SELECT LAST_NAME,
cont>         FIRST_NAME,
cont>         DEPARTMENT_NAME
cont> FROM EMPLOYEES E, DEPARTMENTS D, JOB_HISTORY JH
cont> WHERE ((JH.EMPLOYEE_ID = E.EMPLOYEE_ID)
cont>         AND
cont>         (D.DEPARTMENT_CODE = JH.DEPARTMENT_CODE)
cont>         AND (JOB_END IS NULL))
cont> ORDER BY LAST_NAME, FIRST_NAME, DEPARTMENT_NAME;
E.LAST_NAME      E.FIRST_NAME    D.DEPARTMENT_NAME
Ames              Louie            Northeastern US Sales
Andriola          Leslie           Telecommunications Industries
Babbin            Joseph           European Sales
Bartlett          Dean             Southern U.S. Sales
Bartlett          Wes              Corporate Sales
.
.
.
```

1.4.1.5 Isolating Unique Data Values It is also possible to isolate unique values for a particular column. You can create a result table that consists of just one row for any given value of a single column (that is, additional rows with the same column value are not returned to you). For example, the EMPLOYEES table contains the column STATE. Suppose you want to list all the states in which employees live, listing each state only once regardless of the number of employees living there. To do so, you would use the DISTINCT clause with the SELECT statement:

```
SQL> SELECT DISTINCT STATE
cont> FROM EMPLOYEES;
STATE
CT
MA
NH
3 rows selected
```

1.4.2 Using SQL Statements in Programs

Executing SQL statements in programs, rather than from interactive SQL, should be the primary use of SQL for database access at your site. Program access provides a number of advantages, including processing speed. This section describes the ways in which you can use SQL in programs, and outlines the online sample programs available with Rdb/VMS. You can find more detailed information on using SQL statements in programs in the *VAX Rdb/VMS Guide to Using SQL*.

1.4.2.1 Using SQL Module Language You can use the SQL module language from *any* host language that supports the VMS Procedure Calling Standard. The SQL module language allows host language programs to execute SQL statements contained in an SQL module file. The module file includes one or more procedures, each of which contains parameter (variable) declarations and at least one SQL statement. The SQL module procedures can be called from any host language program, including those not supported by the SQL precompiler. A call to a procedure in an SQL module causes the SQL statement in the procedure to be executed.

The SQL module language offers these advantages to precompiled SQL:

- Languages not supported by the SQL precompiler can be used with the SQL module language instead.
- Isolation of all SQL statements in SQL modules improves modularity and avoids the use of two languages in the same source file.
- Programs can work around restrictions of the SQL precompiler by calling SQL modules.
- Programs written in languages for which there is an ANSI/ISO standard can avoid embedding nonconforming code by isolating SQL statements in SQL modules.

For any host language, you can use SQL module language to create a module of SQL procedures. In your host language source file you can then specify calls to these SQL procedures. Process your program by running:

- 1 The SQL module processor to create an SQL object file
- 2 The host language compiler to create a host language object file
- 3 The VMS Linker to create an executable image from the preceding object files
- 4 The image created by the linker

1.4.2.2 Using Precompiled SQL SQL statements can be embedded directly in certain host language source files. These source files can then be processed by the SQL precompiler. The precompiler converts embedded SQL statements to a form understandable by the host language compiler. The precompiler also invokes the host language compiler and creates an intermediate object file ready for linking.

If your host language is VAX Ada, VAX C, VAX COBOL, VAX FORTRAN, VAX Pascal, or VAX PL/I, you can embed SQL statements directly in a host language source file and process your program by running:

- 1 The SQL precompiler (which indirectly uses your host language compiler) to create an object module
- 2 The VMS Linker to create an executable image
- 3 The executable image created by the linker

1.4.2.3 Using Dynamic SQL Dynamic SQL is a set of SQL statements and data structures that allows programs to formulate and process SQL statements at run time. Supported by both the SQL module processor and the SQL precompiler, dynamic SQL offers programs some advantages normally available only in an interactive environment. For example, applications that use dynamic SQL can:

- Enter SQL statements from a terminal
- Accept SQL statements from a front-end program
- Translate statements in another language into SQL
- Act as a generic Rdb/VMS server for applications running on remote systems other than VMS systems

1.4.2.4 Using the SQL Online Program Examples Online SQL source program examples are available for your use in the directory defined by the logical SQL\$SAMPLE. These program examples define and query the sample personnel database. You can create an executable form (image) of any source program example for which your system has language support and then run the image to see how the program works. You can also create hardcopy listings of program sources and use them as references when you create your own applications.

The online program examples are presented in several languages and include:

- **SQL\$ALL_DATATYPES**
Illustrates INSERT and UPDATE statements, variations in techniques of handling null values, and the correspondence between SQL host language data types and their conversions.
- **SQL\$DIST_TRANS**
Illustrates how to use distributed transactions. This program updates two Rdb/VMS databases, which contain information about the employees in a company's two divisions, and one VAX DBMS database, which contains central information about all employees in the company.

- **SQL\$DYNAMIC**
Illustrates dynamic SQL. This program consists of two files, **SQL\$DYNAMIC** and **SQL\$DYNAMIC_DRIVER**. The Ada module language version of this program provides a comprehensive example of a general dynamic SQL application. The PL/I version provides a more limited example.
- **SQL\$INSERT_DEGREES**
Illustrates SQL module language. This program adds information to the database about an employee's college degrees.
- **SQL\$LOAD_(table-name)**
Loads indicated table from a data file. Specific tables are used to illustrate various techniques. For instance, loading the **EMPLOYEES** table illustrates how to set a **CHAR** column to null, based on a column value in the data file; and loading the **JOB_HISTORY** table illustrates how to set a **DATE** column to null, based on a column value in the data file.
- **SQL\$MULTI_STMT_DYN**
Illustrates how to use extended dynamic cursors to process any number of dynamic SQL statements.
- **SQL\$REPORT**
Writes a salary report with control breaks on the department code and job code columns to print the average salary per job and total salary per department, and writes the grand total of department salary totals.
- **SQL\$RESUMES**
Illustrates how to insert segmented strings (lists) into the personnel database.
- **SQL\$TERMINATE**
Illustrates how to update the personnel database when an employee is terminated.

To find out what is available for precompiled SQL programs in a specific language, enter the **DIRECTORY** command at the DCL prompt as follows. Specify the asterisk wildcard character (*) for the file name and the

appropriate language-related file type. These file types are summarized in Table 1-6. For example, to display the programs written in COBOL, enter:

```
$ DIR SQL$SAMPLE:*.SCO
```

Table 1-6 Program File Types

Language	File Type
Ada	SQLADA
C	SC
COBOL	SCO
FORTRAN	SFO
Pascal	SPA
PL/I	SPL

To locate SQL module language programs, enter the following DCL DIRECTORY command:

```
$ DIR SQL$SAMPLE:*.SQLMOD
```

1.4.3 Accessing Rdb/VMS from Systems Other Than VMS Using SQL/Services

Application programs on various types of systems other than the VMS system can access an Rdb/VMS database through SQL/Services, a client/server system that consists of:

- An Application Programming Interface (API) library of callable routines that provides functions similar to the dynamic SQL statements. The data structures and SQL statement syntax accepted by SQL/Services are identical to that accepted by dynamic SQL. In addition, the API library provides routines to establish server connections and to define postprocessing of result tables.
- A server process that is present on all systems running Rdb/VMS Version 3.1 or higher. The server accesses Rdb/VMS databases through dynamic SQL.

The SQL/Services API library communicates with the server process in a manner that is virtually transparent to the application. To do that, SQL/Services uses the same kind of structure as SQL: the SQL Descriptor Area (SQLDA) and the SQL Communications Area (SQLCA). Other than ensuring that DECnet is installed on the client and server systems, no knowledge of networking is required to develop SQL/Services applications.

1.5 Accessing Rdb/VMS Using RDO, RDML, and RDBPRE

Rdb/VMS supplies several other interfaces in addition to SQL. This section discusses the other Rdb/VMS interfaces supplied with your kit and outlines the online program examples located in the RDM\$DEMO directory that use those interfaces.

1.5.1 Introducing RDO, RDML, RDBPRE, and Other Interfaces

Other interfaces to Rdb/VMS include:

- **RDO/Callable RDO**

The Relational Database Operator (RDO) interface consists of data definition and manipulation statements that you can use either interactively or in application programs. The Callable RDO interface lets you access an Rdb/VMS database from a program written in any host language supported by the VAX Procedure Calling Standard.

Note *Callable RDO uses more resources than either RDBPRE or RDML. If you prefer not to use SQL, and you are programming in BASIC, C, COBOL, FORTRAN, or Pascal, you should use the RDBPRE or RDML interfaces rather than Callable RDO. However, you will have to use Callable RDO to perform certain data definition tasks.*

- **RDML**

The Relational Data Manipulation Language (RDML) consists of data manipulation statements that you can embed in C and Pascal application programs. These programs are processed by the RDML preprocessor. After preprocessing, you can submit the resulting source code to the host language compiler.

- **RDBPRE**

The RDBPRE preprocessor lets you access an Rdb/VMS database from BASIC, COBOL, or FORTRAN programs. You can embed statements directly in your host language program.

In addition, DATATRIEVE, RALLY, DECdecision, TEAMDATA, and VIDA are products that can be used to access Rdb/VMS databases.

1.5.2 Using the Online Program Examples for RDO, RDML, and RDBPRE

Online sample programs are available for use with the RDBPRE and RDML preprocessors and with Callable RDO. These programs are in the directory specified by the logical name RDM\$DEMO, and they are referred to extensively in the language-specific chapters of the *VAX Rdb/VMS Guide to Using RDO, RDBPRE, and RDML*. You can create an executable form (image) of any source program example for which your system has language support and then run the image to see how the program works. You can also create hardcopy listings of program sources and use them as references when you create your own applications.

The exact file names for the programs vary depending on the source language and the interface being used. File names containing the following strings are available:

- **SAMPLE**
A program that performs numerous data manipulation operations on the sample PERSONNEL database
- **CALL_OTHER**
A module called from the SAMPLE program that passes that program information for preprocessing and returns control to the SAMPLE program
- **ERROR**
A module that handles run-time errors for the SAMPLE program
- **DEPTFOR.FOR**
A program that performs several data manipulation operations on the PERSONNEL database
- **F_DDL_STMNT.FOR**
A program that demonstrates how to perform data definition tasks from an RDBPRE FORTRAN program using Callable RDO

Table 1-7 lists the available programs with their supported languages, and indicates the applicable interface.

Table 1-7 Online Program Examples for RDBPRE and RDML

Language	Program	Interface
BASIC	B_SAMPLE.RBA	RDBPRE
	B_CALL_OTHER.RBA	RDBPRE
	B_CALLABLE_ERROR_HANDLER.BAS	Callable RDO
	B_ERROR_HANDLER.BAS	RDBPRE
COBOL	COB_SAMPLE.RCO	RDBPRE
	COB_CALL_OTHER.RCO	RDBPRE
	COB_CALLABLE_ERROR_HANDLER.RCO	Callable RDO
FORTRAN	F_SAMPLE.RFO ¹	RDBPRE
	F_CALLABLE.FOR	Callable RDO
	F_CALLABLE_ERROR_HANDLER.FOR	Callable RDO
	DEPTFOR.FOR	Callable RDO
	F_DDL_STMNT.FOR	RDBPRE/Callable RDO
C	C_SAMPLE.RC	RDML
	C_CALL_OTHER.RC	RDML
	C_ERROR.RC	RDML
Pascal	P_SAMPLE.RPA	RDML
	P_CALL_OTHER.RPA	RDML
	P_ERROR.RPA	RDML
PL/I	PLI_SAMPLE.PLI	Callable RDO
	PLI_CALL_OTHER.PLI	Callable RDO

¹Calls many other programs

1.6 Maintaining and Tuning Rdb/VMS Databases with RMU

Rdb/VMS provides the Rdb/VMS Management Utility (RMU) to aid in the administration and maintenance of the database. This utility provides tools to perform the following functions:

- Analyze data storage patterns

The RMU/ANALYZE command collects and displays information on how data is being stored in the database within both storage areas and logical areas, and describes in detail the index structure or structures for the database.

- **Back up and restore the database**

The **RMU/BACKUP** command creates a full or partial (incremental) backup file of the database or the after-image journal file. The backup operation can be performed on line or off line. The **RMU/BACKUP** utility is **multithreaded** (that is, multiple tape devices can be used simultaneously) to allow quick and efficient performance. The **RMU/RESTORE** command restores the database to the condition it was in at the time of the last full or incremental backup operation performed with the **RMU/BACKUP** command. You can specify a new after-image journal file for the database while restoring it. You can use the **RMU/VERIFY** command before backing up your database to ensure that the database is not corrupt.

- **Open and close the database**

The **RMU/OPEN** command manually opens the database. The **RMU/CLOSE** command manually closes the database, letting you perform specific maintenance activities.

- **Display the internal contents of various database files**

The **RMU/DUMP** command displays the contents of the database root file, storage area files, journal files, snapshot files, the backup file, and information about the current users of the database, including all users in a VAXcluster environment.

- **Monitor the database**

The **RMU/MONITOR** command starts or stops the Rdb/VMS database monitor process on your system. You can create a new version of the monitor log file for the database monitor running on your node. An Rdb/VMS monitor process must be running on each node on which you use Rdb/VMS.

- **Load and unload tables and views from the database**

The **RMU/LOAD** command loads specific tables of an Rdb/VMS database from specially structured files or from sequential record management services (VAX RMS) files. The **RMU/UNLOAD** command extracts the data from tables or views within an Rdb/VMS database into either sequential RMS files or specially structured files. These commands are useful to perform an initial load operation, to restructure specific tables, to create archival databases, or to access data from applications using RMS files.

- **Display information about your Rdb/VMS database**

The **RMU/SHOW** command displays information about the Rdb/VMS system on the node from which you issue the **RMU/SHOW** command. You can display information about database statistics, database users, the database system being used, and the version of Rdb/VMS running on your system.

- **Recover the database**
The `RMU/RECOVER` command recovers the database from a restored version. This command uses the after-image journal file to roll forward the database to its current state.
- **Verify the integrity of the database**
The `RMU/VERIFY` command checks and verifies the internal integrity of data structures. It can also verify logical areas and indexes. It does not check for invalid data. Use this command before backing up your database.

1.7 Rdb/VMS Product Kits

Different sites require different capabilities from an Rdb/VMS database. For your convenience, Rdb/VMS provides the following three kits:

- **Rdb/VMS Full Development Kit**
The Rdb/VMS full development kit lets you create and modify databases, create programs that use databases, and run programs that use databases. It lets you access Rdb/VMS databases through SQL/Services client applications. The full development kit is necessary if you will be accessing Rdb/VMS using programs written in standard programming languages (3GLs, that is, third-generation languages).
- **Rdb/VMS Interactive Kit**
The Rdb/VMS interactive kit lets you use existing Rdb/VMS databases and build new Rdb/VMS databases using the same full-function SQL and RDO interactive interfaces that are supplied with the full development kit. It contains the SQL/Services server component. However, it does not let you preprocess data manipulation language (DML) programs; it does not include any precompilers, so you cannot develop programs.
- **Rdb/VMS Run-Time Kit**
The Rdb/VMS run-time kit lets you use the Rdb/VMS database built with the full development kit or interactive kit. It contains the SQL/Services server component. However, it does not let you create new databases, modify data definitions of existing databases, or preprocess DML programs. The run-time kit is useful if you use prepared application solutions based on Rdb/VMS, or if you access Rdb/VMS databases from DECdecision, TEAMDATA, DATATRIEVE, or RALLY applications.

1.8 Using Rdb/VMS with Other Digital Products

You can use Rdb/VMS with other Digital products, including the following:

- **ACMS**

An application development system that implements and manages transaction processing applications. Use ACMS when you must manage large transaction processing applications whose data is stored in an Rdb/VMS database.

- **CDD/Plus**

A central storage facility for data definitions. Use CDD/Plus to centrally store the data definitions your application programs will be using.

- **Data Distributor**

A product that gives a database administrator control of the process of distributing database access across remote nodes in a network. Use Data Distributor when your Rdb/VMS database must be distributed across remote nodes.

- **DATATRIEVE**

An interactive query language that includes the capability to generate reports and graphics. Use DATATRIEVE as an interface to your Rdb/VMS database. DATATRIEVE is also useful for generating attractive reports based on data in your Rdb/VMS database. You can use DATATRIEVE to get data in a single query from Rdb/VMS, VAX DBMS, and RMS files.

- **DECdecision and TEAMDATA**

An information management and decision support system that you can use as an interface to your Rdb/VMS database. Use DECdecision in DECwindows environments, or TEAMDATA in character-cell environments.

- **DECtrace**

A product that collects and reports data and performance information from databases on an event basis (as opposed to products that collect on a timer basis). You can have DECtrace collect application workload information, and then have RdbExpert import that information.

- **RALLY**

A fourth-generation language (4GL) environment that constructs complete applications quickly. Use RALLY applications as end-user interfaces to your Rdb/VMS database.

- **RdbExpert**

A product that you can use to optimize the physical design of your Rdb/VMS database. Using RdbExpert, you specify information about the application workload, data volume, and system environment of the database. RdbExpert applies its design rules to the schema and to the information you have supplied. It generates several design reports, as well as a command procedure that (with minimal edits) you can run to create a new database with an optimal physical design. This procedure also unloads any existing data and reloads it in the new database.

- **TDMS**

A forms package that manages the display of forms and the movement of data to and from the terminal screen. Use TDMS to generate the forms your application programs will use to permit users to enter data interactively.

- **VAX DBMS**

The Digital CODASYL-compliant database. VAX DBMS is intended for large, highly complex applications.

- **VIDA**

A data access software product. A VIDA database contains data that resides on an IBM system. You can use SQL to retrieve data through VIDA but not to define or update a VIDA database.



Rdb/VMS Documentation Directory

This chapter briefly describes the manuals in the Rdb/VMS full development license documentation set. The introductory manual is listed first, followed by more advanced user guides and reference manuals, and finally by the release notes and installation guide.

All manuals other than the *VAX Rdb/VMS SQL Quick Reference Guide* are available on CDROM.

Titles dealing specifically with the RDO or RDML interfaces retain terminology associated with the Digital proprietary RDO interface (for example: relation, record, field). All other titles use terminology associated with the industry-standard SQL interface (for example: table, row, column).

VAX Rdb/VMS Introduction and Master Index (this manual)

Audience: All users

Content: Introduces VAX Rdb/VMS, the Digital relational database management system for VMS software environments. Explains major terms and concepts. Includes a glossary, a directory of Rdb/VMS documentation, and a master index that combines entries from all the Rdb/VMS manuals.

VAX Rdb/VMS Guide to Using SQL

Audience: All users

Content: Describes and illustrates the use of Rdb/VMS features with the SQL industry-standard interface. Shows how to retrieve, modify, and delete data stored in a database. Explains how to use transactions, select expressions, joins, and views. Also explains how to use SQL within programs. You can use this guide as a tutorial for learning the major features of SQL.

VAX Rdb/VMS Guide to Using SQL/Services

Audience: Application programmers

Content: Describes how to develop application programs that use SQL/Services, a client/server software component of Rdb/VMS that allows programs, from various remote computers running the Macintosh, MS-DOS, OS/2, ULTRIX, ULTRIX for RISC, or VMS operating systems to access Rdb/VMS or VIDA databases on a VMS server system.

VAX Rdb/VMS Guide to Distributed Transactions

Audience: Programmers

Content: Describes the two-phase commit protocol and distributed transactions, explains how to start and complete distributed transactions using SQL, RDBPRE, and RDML, and how to recover from unresolved transactions using RMU commands.

VAX Rdb/VMS Guide to Database Design and Definition

Audience: Database designers and administrators

Content: Explains how to design a database and how to set up definitions of database entities. Also guides you from the analysis of your organization's information needs, through logical and physical database design processes, and through the actual creation of the database. Discusses database security and use of the VAX CDD/Plus data dictionary.

VAX Rdb/VMS Guide to Database Maintenance and Performance

Audience: Database administrators and operators

Content: Shows how to use the Rdb/VMS database maintenance utilities to keep the database running and to keep its data consistent. Explains how to perform backup and restore operations, how to handle database journaling, how to evaluate and improve your database performance, and how to optimize the way in which your database uses system resources.

VAX Rdb/VMS Guide to Database Tuning

Audience: Database administrators

Content: Introduces the concept of tuning, and explores how tuning the system, the database, and the application can affect database performance. Outlines a series of steps to follow in identifying, analyzing, isolating, and solving a performance problem, and in monitoring the resulting solution. Includes a set of decision trees that provide an organized approach to solving some common database tuning problems.

VAX Rdb/VMS Guide to Using RDO, RDBPRE, and RDML

Audience: All users

Content: Describes and illustrates the use of Rdb/VMS features with the Relational Database Operator (RDO) Digital-proprietary interface. Shows how to retrieve, modify, and delete data stored in a database. Explains how to use transactions, record selection expressions, relational joins, and views. Also explains how to use RDO, RDML, and Callable RDO within programs. You can use this guide as a tutorial for learning the major features of the RDO interactive utility.

VAX Rdb/VMS SQL Reference Manual

Audience: All users

Content: Contains complete reference information on features of Rdb/VMS that you can specify using the industry-standard SQL interface. Gives syntax and usage for all data definition statements, data manipulation statements, database maintenance utility statements, statements for setting up the interactive environment, and VAX Data Distributor statements.

VAX Rdb/VMS SQL Quick Reference Guide

Audience: All users

Content: Summarizes the information in the *VAX Rdb/VMS SQL Reference Manual*.

VAX Rdb/VMS RDO and RMU Reference Manual

Audience: All users

Content: Contains complete reference information on features of Rdb/VMS that you can specify using the Relational Database Operator (RDO) utility. Describes major Rdb/VMS terms and concepts. Gives syntax and usage for all data definition statements, data manipulation statements, database maintenance utility statements, VAX Data Distributor statements, and statements for setting up the interactive environment. Also includes a chapter on the Rdb/VMS Management Utility (RMU) for administrative and maintenance tasks.

RDML Reference Manual

Audience: Programmers

Content: Describes the syntax and usage of the Relational Data Manipulation Language (RDML), which can be embedded in VAX C or VAX Pascal programs to access Rdb/VMS or Rdb/ELN databases.

VAX Rdb/VMS Release Notes

Audience: All users

Content: Includes information about new features, problems, restrictions, and sometimes other important information about the current Rdb/VMS release. Sometimes contains material that became available too late for inclusion in the other Rdb/VMS manuals.

VAX Rdb/VMS Installation Guide

Audience: System managers

Content: Explains how to install Rdb/VMS and run the Installation Verification Procedure (IVP).

Glossary

This glossary defines terms used in the documentation for VAX Rdb/VMS and related products.

access control entry (ACE)

ACL-style protection. An individual entry on an access control list.

See also access control list (ACL) and ACL-style protection.

access control list (ACL)

ACL-style protection. A list that defines which users can access a schema, table, or view and what operations they can perform. The SQL statements GRANT and REVOKE create, modify, and delete entries in an ACL. Position of an entry on an access control list can affect the privileges allowed.

See also access control entry (ACE), ACL-style protection, and privilege.

access mode

Another term for access mode is lock type.

See lock type and reserving option.

access privilege set (APS)

The generic term used to refer to either ACL-style or ANSI-style protection schemes.

See also ACL-style protection and ANSI-style protection.

access rights

Other terms for access rights are rights or privileges.

ACE

See access control entry (ACE).

ACL

See access control list (ACL).

ACL-style protection

A protection scheme that involves access control lists in which the position of an access control entry is critical.

See also access control entry, access control list, access privilege set, and ANSI-style protection.

ACMS

The VAX ACMS, an application control and management system, is a transaction processing monitor that is layered on VMS and is used to define, run, and control transaction processing applications. You can use ACMS to control existing applications and applications developed with ACMS. It provides a task implementation method that uses high-level definitions to replace complex application code. These definitions reduce the programming time and maintenance costs of traditional application programs that do comparable work. ACMS supports offloaded terminal processing, which allows the execution of ACMS tasks on remote nodes.

See also distributed transaction processing, online transaction processing, and transaction processing.

actual parameter

A parameter declaration in a host language program used in a call statement to a procedure in an SQL module file. Another term for call parameter.

after-image journal (AIJ)

A file that contains copies of records after they have been updated. You can use the after-image journal to reconstruct a saved backup file of a database up to the last successfully completed transaction.

See also before-image journal.

aggregate expression

Another term for function.

AIJ

See after-image journal (AIJ).

alarm

See security alarm.

alias

A qualifier for column names that is used in place of a full table or view name in an SQL SELECT statement. Users can qualify a column name with its table or view name, or with an arbitrary alias they specify in the FROM clause of an SQL statement. SQL requires aliases as qualifiers instead of table or view names in statements that join a table with itself. Another term for alias is context variable.

allocation

The number of pages allotted to each storage area or snapshot file of the database in the SQL CREATE SCHEMA statement.

See also extent, multifile database (MFDB), and storage area.

allow mode

Another term for allow mode is share mode. *See* reserving option.

anchor

A VMS directory that contains all the files that describe a CDO dictionary and directory system.

See also CDD/Plus and Common Dictionary Operator (CDO) utility.

ANSI

American National Standards Institute, the organization that approved a standard, ANSI X3.135-1989, for SQL database languages. That standard specifies the syntax and semantics of interfaces to define and query databases using SQL. ANSI is also developing an extended SQL standard, SQL2, in conjunction with the International Standards Organization (ISO).

See also ISO.

ANSI-style protection

A protection scheme that involves a set of privileges defined at both the system level and the individual level. In contrast to ACL-style protection, the position of the user within the access privilege set does not affect the privileges granted to that user.

See also access privilege set (APS) and ACL-style protection.

API

See Application Programming Interface (API).

Application Control and Management System (ACMS)

See ACMS.

application program

A sequence of instructions and routines, not part of the basic operating system, designed to serve the specific needs of a user. An application program can use a database system, a fourth-generation language like DATATRIEVE, or RMS to access data.

See also run unit.

Application Programming Interface (API)

In SQL/Services, a library of callable routines that are linked into the client executable image.

See also client, server, and SQL/Services.

APS

See access privilege set (APS).

area

See storage area.

area bit map (ABM)

A database page found in storage areas with uniform page format in both single-file and multifile databases. The first page of a logical area is an area bit map. The area bit map can be used to speed up sequential scans of table rows.

See also area inventory page (AIP) and uniform page format.

area inventory page (AIP)

A database page that maintains a queue of database area pointers to logical area bit maps. The area inventory page points to the area bit map for each table.

arithmetic expression

A value expression formed by combining one or more numeric value expressions with arithmetic operators.

array

A data structure that consists of more than one data item, in which all data items have the same data types and are referred to by the same variable name.

ascending order

A sorting order that starts with the lowest value of a sort key and proceeds to the highest value, in accordance with the rules for comparing data items.

See also descending order and sort key.

ASCII

A computer character set and collating sequence. The acronym stands for American Standard Code for Information Interchange. The ASCII character set occupies the first 128 positions of the DEC Multinational Character Set (MCS) that is integral to VAX systems.

ASSOCIATE_STR

A data structure passed to the SQL/Services API routine that allows an application to modify association characteristics.

See also Application Programming Interface (API), association, and SQL/Services.

association

A communications link and related context between the SQL/Services client and database server processes. In this case, the context refers to the data structures that the software creates and uses (on both ends of the link) to keep track of what is happening.

See also client, server, and SQL/Services.

attach (to a database)

The operation by Rdb/VMS that establishes a connection between your program or interactive session and a database. This connection is identified by a database handle that Rdb/VMS assigns as part of establishing the connection. The DECLARE SCHEMA statement in SQL and the INVOKE DATABASE statement in RDO attach to the specified database(s). The term *attachment* is sometimes used instead of the term *attach*.

See also database handle, and detach.

attachment

A connection between a host program or interactive utility and the database it will access. The term *attach* is sometimes used instead of the term *attachment*.

attribute

See field attribute.

audit

See security audit.

audit event

See security event.

audit trail

The list of audit activities (security events) that Rdb/VMS writes to a database security audit journal when the audit feature is enabled for the database by the RMU/SET AUDIT command.

See also security audit, security event and security audit journal.

authorization identifier

In SQL, a name specified in a DECLARE SCHEMA or CREATE SCHEMA statement that identifies the schema in subsequent SQL statements.

See also database handle.

backup operation

An operation that creates a backup copy of the database. A backup database is used to restore the database after a hardware or software failure.

There are two types of backup operations:

- **RMU/BACKUP**

Use the RMU/BACKUP command from the DCL level to create a database backup file (default file type RBF) that can be restored using the RMU/RESTORE command.

- **EXPORT**

Use the EXPORT statement to migrate a database from one database management system to another, or to restructure a database.

Migration changes include taking an existing Rdb/VMS database, backing it up with an EXPORT statement, and then using the IMPORT statement to convert the internal data structures so that they are compatible with a newly installed version of Rdb/VMS. Restructuring changes include taking a single-file database, backing it up using the EXPORT statement, and then using the IMPORT statement to define one or more storage areas in a new multifile database.

See also export operation, import operation, and restore operation.

base table

A table that is physically stored in the database.

See also view.

batch processing (Rdb/VMS)

A mode of computer operation in which the SQL or RDO statements and the data that control the actions of the computer are entered by a programmed script rather than by a person at a terminal.

See also interactive processing (Rdb/VMS).

batch-update transaction

A transaction mode that executes without the overhead, or security, of a recovery-unit journal. The batch-update transaction is intended for initial loads of databases. Should *any* error occur during the batch-update transaction, the entire database is permanently corrupted.

See also lock type, reserving option, share mode, and transaction.

before-image journal

A file that contains copies of records before they have been updated. Rdb/VMS uses before-image journaling to automatically remove updates to a database when a transaction is rolled back. Another term for before-image journal is recovery-unit journal (RUJ).

See also after-image journal (AIJ) and idempotent.

before-image journaling

In Rdb/VMS databases, the method of keeping track of changes you make to the database in the before-image journal file. It is this journaling that lets you issue a ROLLBACK statement to remove changes you made to the database.

binary language representation (BLR)

The binary language representation of requests to the database. In other words, the binary form of data manipulation language. A request in this context is a sequence of instructions given to the database system by a host program for compilation and possible execution.

See also data manipulation language (DML), metadata binary language representation (MBLR), and request.

bind

Another term for attach.

bitvector

A structure that is part of the area bit map (ABM) page that contains set bits that indicate the corresponding space area management pages that have entries for a logical area.

block structure

A programming construct that defines the lexical scope of program variables. For example, C is a block structured language where a left brace ({) denotes the beginning of a block and a right brace (}) denotes the end. Blocks are often nested.

BLR

See binary language representation (BLR).

Boolean expression

Another term for predicate or conditional expression.

See also Boolean operator, conditional operator, and predicate.

Boolean operator

A keyword that lets you join two or more predicates to form a complex predicate. Boolean operators negate a predicate (NOT), specify a combination of predicates (AND), or specify a list of alternative predicates (OR). Another term for Boolean operator is logical operator.

broadcast message

A text line that lets you know of a system event, such as a system shutdown or the receipt of mail.

B-tree

A sorted index structure for a specified table (system or user). Specified by using the TYPE IS SORTED clause in the DEFINE INDEX statement.

See also hashed index, index, and sorted index.

built-in function

Another term for function.

cache

The process of storing blocks in memory for future use; used to minimize physical transfer of data between mass storage devices and memory.

Callable RDO

An interpretive call interface that consists of a single external routine that accepts an Rdb/VMS data manipulation language (DML) or data definition language (DDL) statement as a parameter. You can call this routine from any language that adheres to the VAX Procedure Calling Standard. Callable RDO lets a program use Rdb/VMS even if no preprocessor exists for the language.

Callable RDO supports embedded DML statements in VAX BASIC, VAX C, VAX COBOL, VAX FORTRAN, and VAX Pascal source programs. To embed DDL statements or to use dynamic strings in Rdb/VMS DML statements, you must use Callable RDO.

See also Relational Database Operator (RDO).

call interface

A mechanism by which a program accesses a software product.

call parameter

Another term for actual parameter.

candidate key

Any column or set of columns that uniquely identifies the individual rows of a table. Candidate key is similar to a primary key. For example, in a table of employee information, the employee identification number is a candidate key. When more than one primary key exists, the alternatives are called candidate keys. Of the two or more candidate keys, a primary key is selected, with the remaining keys becoming secondary keys.

See also foreign key, primary key, and secondary key.

cardinality

The number of rows in a table or the number of indexed entries in a defined index.

See also index, row, and table.

Cartesian product

For multiple tables that are joined by a select expression in SQL, the Cartesian product is a result table that consists of every possible combination of all the rows and columns of each table. SQL forms the Cartesian product when it evaluates a FROM clause that names more than one table. Another term for Cartesian product is cross product.

See also join.

cascading delete

See cascading update.

cascading update

An update operation performed by a trigger definition that will update appropriate rows from any necessary table. For example, in the sample personnel database, you could have a cascading update specify that whenever an employee leaves the company, the row for that employee will be deleted not only from the EMPLOYEES table, but from the JOB_HISTORY, SALARY_HISTORY, and DEGREES tables as well. Such an operation would be called a cascading delete operation.

See also table-specific constraint and trigger.

CDD/Plus

An abbreviated name of VAX CDD/Plus, a VMS software component through which users can define, organize, and control access to the data and forms definitions required by applications. Users can access an Rdb/VMS database through a database definition they store in CDD/Plus. Rdb/VMS documentation sometimes refers to CDD/Plus simply as the data dictionary.

The data dictionary lets you ensure the integrity of the shared metadata and provides procedures you need to analyze, maintain, manage, and design your business' database; it provides a useful centralized repository for information management.

CDD/Plus supports metadata created in an appropriate format for both the Dictionary Management Utility (DMU) and the Common Dictionary Operator (CDO).

Support for CDD/Plus by Rdb/VMS enables you to:

- Define global column and row definitions in a shareable dictionary
- Directly copy the column or row definition(s) or both from the dictionary to an Rdb/VMS database when you define new Rdb/VMS columns or tables in SQL or RDO
- Receive informational messages about the CDD/Plus column and row definitions that are used by other Rdb/VMS databases
- Integrate shareable dictionary definitions into an Rdb/VMS database (the INTEGRATE SCHEMA FILENAME statement) or integrate database definitions to a CDD/Plus dictionary directory (the INTEGRATE SCHEMA PATHNAME statement)

Rdb/VMS does not require CDD/Plus.

CDO

See Common Dictionary Operator (CDO) utility.

CDROM

Compact disk read-only memory. New type of disk that is used as a distribution medium for software and documentation.

character string

A string of characters (bytes) that is identified by an address and a length.

checksum

A sum of digits or bits used to verify that a number or an operation is valid. For a database page, the sum of all bits on the page.

client

A service requester. In SQL/Services, an application program that uses the Application Programming Interface to request services from the SQL/Services server.

See also Application Programming Interface (API), SQL/Services, and server.

clump

A set of adjacent pages within a storage area when uniform page format is specified. These pages are devoted to specific logical areas which either store rows from a specific table or nodes from one or more sorted indexes on that table by the area's logical ID.

clumplet

A portion of the RDBVMS\$TRIGGER_ACTIONS segmented string field that contains a trigger action for a specified trigger. The RDBVMS\$TRIGGER_ACTIONS field is part of the RDBVMS\$TRIGGERS system relation.

See also trigger.

cluster

See VAXcluster.

clustered index

Hashed indexes whose hash buckets reside on the same page of a storage area file as their associated table rows. Clustering hash buckets with rows may make it possible for one input/output operation to load the hash bucket and rows into memory.

See also hash bucket and hashed index.

CODASYL

An acronym for the Conference on Data Systems Languages, the committee that designed the COBOL language and provided the guidelines used in the development of VAX DBMS, the Digital network database.

CODASYL-compliant

Any database system that conforms to the guidelines set by the Conference on Data Systems Languages. VAX DBMS is CODASYL-compliant.

collating sequence

The sequence in which characters are ordered for sorting, merging, and comparing.

See also National Character Set (NCS) utility.

column

The vertical dimension of a table. A column has a name and a data type, and all values in a column have that same data type. Another term for column is field.

See also data item and domain.

column select expression

A select expression that specifies a one-column result table.

SQL accepts column select expressions as arguments to IN and quantified predicates, and more generally as value expressions:

- As arguments to IN and quantified predicates, column select expressions specify a collection of values to which SQL compares a value expression. Thus, column select expressions used as arguments to those predicates can return one or more values.
- As a type of value expression, column select expressions specify a single value. Thus, a column select expression used as a value expression should not return more than one value. If it does, SQL evaluates the value expression *based on the first value returned* by the column select expression. For the same reason, a column select expression used as a value expression cannot include GROUP BY or HAVING clauses.

Other terms for column select expression are record selection expression (RSE) and subquery.

See also record stream and select expression.

comment character

A character that begins a line of descriptive text in a program or procedure; it does not affect program or procedure execution. Comment lines begin with a comment character reserved by the language you are using. Some typical comment characters are the exclamation point (!), the asterisk (*), and the letter C. The exclamation point is the comment character used by SQL and the RDO utility. You can also use

a double hyphen (--) in interactive SQL and SQL module language. Comment lines end with a carriage return.

commit

To make permanent any changes to the database made during the current transaction. Use the COMMIT statement to commit changes.

See also roll back and transaction.

commit phase

The second phase of the two-phase commit protocol. During the commit phase, the coordinator of the distributed transaction instructs each resource manager to make permanent all changes to the database, that is, to commit the changes. The resource managers commit the changes and the distributed transaction is completed.

See also coordinator, distributed transaction, prepare phase, resource manager, and two-phase commit protocol.

common data dictionary

See CDD/Plus.

Common Dictionary Operator (CDO) utility

The user interface to CDD/Plus. With CDO, you can access any CDO or DMU dictionary to which you have privileges, create or delete field and record definitions, and run all CDO commands.

See also Dictionary Management Utility (DMU).

communication server

In SQL/Services, a multithreaded process on a VMS server system that accepts Application Programming Interface (API) requests, assigns them to execution server processes, and asynchronously returns execution results to client applications. The communication server sets up the process pool during the initializing phase at system startup time, in preparation for handling requests during the working phase.

See also Application Programming Interface (API), client, execution server, initializing phase, process pooling, server, and working phase.

complex predicate

A predicate that combines any number of predicates using the Boolean operators AND, OR, or NOT.

compress

To reduce in size. When data is compressed, free space is eliminated as much as possible.

compressed index

An index in which information in the index node is compressed so that data takes up less space.

See also compress, index.

COMPUTED BY columns

Virtual columns that appear in a table or view definition, but not in the physical row. Because the value of a COMPUTED BY column is computed as part of a statement, it occupies no space in the row.

concurrency

The simultaneous use of a database by more than one user.

See also consistency.

conditional expression

Another term for predicate or Boolean expression.

See also conditional operator and predicate.

conditional operator

A keyword that specifies how you want to compare value expressions in a predicate. SQL uses these conditional operators:

=	(equal to)
<>	(not equal to)
<	(less than)
<=	(less than or equal to)
>	(greater than)
>=	(greater than or equal to)

BETWEEN

CONTAINING

EXISTS

IN

IS
LIKE
STARTING
WITH
UNIQUE

Another term for conditional operator is relational operator.

configuration file

In SQL/Services, a file that the communication server reads at system startup time to set up the process pool. The SQLSRV\$CONFIG.DAT file in the SYS\$STARTUP system directory contains one or more definitions, each consisting of a set of parameters. Each definition describes the characteristic of a subpool within the process pool. An initial version of the file is stored on the system during installation. The system manager can modify the file at any time to better suit application requirements.

See also communication server, process pooling, and subpool.

consistency

The level to which a database system guarantees that records being read by a user cannot be changed simultaneously by other users.

See also concurrency.

constant

Another term for literal.

constraint

A condition that restricts the values that can be stored in a table. When you insert and update column values, the constraint checks the values against the conditions specified by the constraint. If a value violates the constraint, an error message is generated and the insert or update statement fails. You specify constraints in SQL CREATE and ALTER TABLE statements. Optionally, you supply a name for the constraints following the SQL DIAGNOSTIC keyword.

context file

A special SQL command procedure that contains SQL-specific declarations such as DECLARE SCHEMA and DECLARE TRANSACTION statements. SQL applies the declarations in a context file to the precompilation and execution of a host language program.

A context file is a way to manage schema and transaction contexts for the program without coding these declarations in source code. Context files help limit SQL statements in programs to generic SQL syntax. They let the program be more easily transported for use with a variety of database products. The SQL precompiler accepts a context file specification as an optional argument when it processes a host language module.

context structure

A data structure in SQL that contains a distributed transaction identifier (TID).

See also distributed transaction identifier (TID).

context variable

Another term for alias.

control break

Partitions between groups of rows with the same value in an intermediate result table. Specifies control breaks with the GROUP BY clause.

See also intermediate result table.

coordinator

A component of a distributed transaction. The coordinator is the transaction manager on the node where the application started a distributed transaction. The coordinator orchestrates the distributed transaction.

See also DECdtm services, distributed transaction, resource manager, transaction manager, and two-phase commit protocol.

correlated reference

Another term for outer reference.

correlated subquery

A subquery (or column select expression) that contains an outer reference.

correlation name

Another term for alias.

cross operation

Another term for join.

cross product

Another term for Cartesian product.

cursor

A result table defined by the select expression in an SQL DECLARE CURSOR statement. Unlike other result tables, a cursor can exist throughout execution of more than one statement. Host language programs require cursors because programs must perform operations one row at a time and therefore may execute statements more than once to process an entire result table.

See also list cursor, result table, and table cursor.

database

A collection of interrelated data on one or more mass storage devices. The collection is organized to facilitate efficient and accurate inquiry and update. In a database, more than one user can access the data at the same time. Data integrity and security are provided by the database. Specifically, database refers to the data definitions, user data, and the database system files associated with a schema.

See also CODASYL, hierarchical database, network database, relational database, and schema.

Database Control System (DBCS)

The component that, together with the VMS operating system, provides run-time control of database processing.

database execution server

In SQL/Services, one of two types of execution servers (database and generic execution servers) that execute Application Programming Interface (API) requests for the communication server. Database servers are already attached to an Rdb/VMS database at system startup time and provide access to Rdb/VMS databases in a specific user environment. Having servers already attached to a database eliminates the need for each API request to make a separate database attach.

See also Application Programming Interface (API), communication server, execution server, generic execution server, and server.

database handle

Another term for authorization identifier. RDO and RDML use *database handle* instead of the term authorization identifier.

See also attach and detach.

database key

See dbkey.

database management system (DBMS)

A system for creating, maintaining, and accessing a collection of interrelated data records that can be processed by one or more applications without regard to physical storage. Data is described independently of application programs, providing ease in application development, data security, and data visibility.

Note that the abbreviation “DBMS” is often used in general technical literature and in the trade press to mean “database management system.” Do not confuse this generic use of the term DBMS with references to the specific Digital software product VAX DBMS, a CODASYL-compliant database management system.

See also CODASYL, database, hierarchical database, network database, and relational database.

database name

A name that a person, program, or product must supply to identify and access a database.

database page

The structure used to store and locate data in an Rdb/VMS database. Rdb/VMS database pages consist of 1 or more disk blocks of 512 bytes each. The default size of a database page is 2 blocks, or 1024 bytes.

database root file

In a single-file database, the database root file contains data, metadata, and system information. In a multifile database, this file contains only system information. The metadata is stored in the RDB\$SYSTEM storage area. Data is stored in the RDB\$SYSTEM or other storage area files. The database root file has the file type RDB.

See also storage area.

database tuning

The process of adjusting system or database parameters to attain optimum resource use for database applications without using unnecessary or excess central processing unit, memory, or input/output capacity.

See also process pooling, query optimizer, SPAM page, and threshold.

data compression

A technique that reduces data to its smallest possible size to optimize data storage. Enabled by default, data compression can be disabled by specifying the `CREATE STORAGE MAP . . . DISABLE COMPRESSION` statement or the `ALTER STORAGE MAP . . . DISABLE COMPRESSION` statement for a given table.

data definition language (DDL)

A set of statements that lets you define the structure and characteristics of stored data. You use data definition language to describe databases characteristics, storage area files, columns, tables, views, indexes, and constraints. Both the SQL and RDO interactive interfaces include Rdb/VMS data definition language.

See also CDD/Plus and metadata binary language representation (MBLR).

Data Distributor

A Digital product that lets you make complete or partial copies of databases. These database copies can be made available to users on the source node or on remote nodes. Data Distributor can copy the databases of three relational database systems: Rdb/ELN, Rdb/VMS, and VIDA. Database copies are in Rdb/VMS format.

See also extraction, Rdb/ELN, Rdb/VMS, replication, source database, source node, target database, and target node.

data element

Another term for data item.

data item

The smallest unit of data that you can retrieve from a row. A data item occupies a single column in a row. Another term for data item is data element.

See also column and domain.

data manipulation language (DML)

A set of statements that lets you store, retrieve, modify, and erase data from a database. There are several methods of manipulating data using SQL:

- Use a high-level language program to execute the SQL statements in an SQL module file.
- Embed the data manipulation statements in a high-level language, such as Ada, BASIC, C, COBOL, FORTRAN, Pascal, or PL/I. SQL has a precompiler for Ada, C, COBOL, FORTRAN, Pascal, and PL/I.

- Issue the data manipulation statements interactively.

The Rdb/VMS data manipulation languages (RDO, SQL) can be used to access a VIDA database as well as an Rdb/VMS database.

See also binary language representation (BLR).

data table

See table.

DATATRIEVE

A VAX query language and data management tool for manipulating, storing, and modifying records from RMS data files, VAX DBMS, Rdb/VMS, and Rdb/ELN databases. DATATRIEVE also generates reports and graphs from data stored in RMS files and VAX DBMS, Rdb/VMS, VIDA, and Rdb/ELN databases. SQL provides DATATRIEVE support clauses in the CREATE DOMAIN, CREATE TABLE, and CREATE VIEW statements, and in the ALTER DOMAIN and ALTER TABLE statements.

DATATRIEVE is callable from a variety of high-level languages.

data type

The characteristic of columns and host language variables that controls how Rdb/VMS interprets and stores values.

DBCS

See Database Control System (DBCS).

dbkey

A unique value that points to specific table rows in a database, identifying the address of the table row. The Database Control System (DBCS) assigns the value when a record is stored in the database. Application programs can use the DBKEY keyword in SQL statements to refer to the database key for a table row. SQL statements that retrieve rows by specifying their dbkeys offer faster access and reduced locking over indexed or sequential searches. Another term for dbkey is database key.

See also scope.

```
DECLIT AA VAX KY66C
```

DBMS

See database management system.

```
VAX Rdb/VMS introduction  
and master index
```

DBR

The name of the process that performs database recovery. It is created by the Rdb/VMS monitor in case of:

- A (hard) executable image termination, where the usual exit mode is bypassed; this can occur when the user terminates an executable image by entering CTRL/Y and then invokes the DCL command STOP or any other command that calls a new image.
- A VMS system failure
- A VAX node exit in a VAXcluster environment

DCL

DIGITAL Command Language. A set of commands that provides an operating system interface for users. When users log in to a VMS system, they typically enter the DCL environment. In this environment, they can create files and directories and use a variety of programs and utilities to get their work done.

Interactive SQL and RDO let users issue DCL commands with the DCL Invoke (\$) statement.

DCL command procedure

A sequence of DIGITAL Command Language (DCL) commands stored in a file; sometimes referred to as a DCL procedure.

See also RDO command procedure and SQL command procedure.

DDL

See data definition language (DDL).

deadlock

A situation in which two or more processes request the same set of resources and there is no method for resolving the conflict other than to reject all but one of the processes. For example, if process A has row 1 locked and requests row 2 while process B has row 2 locked and is requesting row 1, a deadlock occurs between the two processes.

DECdtm services

The VMS utility that coordinates distributed transactions by providing the underlying operating system support for distributed transactions. DECdtm services

supports the two-phase commit protocol, and guarantees consistent execution of the distributed processing of applications on the VMS operating system.

See also coordinator, distributed transaction, resource manager, transaction manager, and two-phase commit protocol.

DECnet

The Digital software facility that implements the DIGITAL Network Architecture (DNA) to let a user access information on a remote computer through telecommunications lines. A DECnet-VAX network enables a VMS system to function as a network node and, for instance, allows programs running on one node to remotely access an Rdb/VMS database on another node in the same DECnet-VAX network.

DECtrace

A product that collects and reports data and performance information from databases on an event basis (as opposed to products that collect on a timer basis). You can have DECtrace collect application workload information, and then have RdbExpert import that information.

See also event and RdbExpert.

default

A value that is assumed unless or until you specifically indicate another value.

default dictionary directory

The CDD/Plus directory assigned to you when you invoke an image that uses the CDD/Plus dictionary. This directory becomes the first directory listed within path names. You can define a directory as the default by assigning a path name to the logical name CDD\$DEFAULT. If you do not, the default directory is the equivalent of the CDD\$COMPATIBILITY logical name. By default, CDD\$COMPATIBILITY is defined as SYS\$COMMON:[CDDPLUS].

See also anchor, dictionary directory, and path name.

default directory

The directory from which the VMS system reads and to which it writes all files that you create unless you explicitly name a directory.

See also directory.

default schema

The database declared without an explicit authorization identifier in the **DECLARE SCHEMA** statement. Table and view names in SQL statements that refer to the default schema do not have to be qualified by an authorization identifier.

In SQL module language, the authorization identifier specified in the module header is the one that designates the default schema. In precompiled SQL programs and in interactive SQL, the special authorization identifier **RDB\$DBHANDLE** designates the default schema. In all environments, omitting an explicit authorization identifier is the same as specifying the authorization identifier that designates the default schema.

If a user or program does not issue an explicit **DECLARE SCHEMA** statement, SQL attempts to declare a default database using the file specification assigned to the logical name **SQL\$DATABASE**.

See also authorization identifier and schema.

default value

The data value that is stored in the database if an insert or update operation on a row specifies no data value for that column. In SQL, you can define a default value for a domain or for a column. A default value using the SQL interface is not the same as the missing value that you can define using the RDO interface.

See also missing value.

deferred snapshots

A process by which update transactions write to the snapshot files only when a read-only transaction is in progress. Deferred snapshots eliminate the unnecessary writing of updated records to the snapshot file.

See also snapshot.

degree (of a table)

The number of columns in a table definition.

descending order

A sorting order that starts with the highest value of a key and proceeds to the lowest value, in accordance with the rules for comparing data items.

See also ascending order and sort key.

descriptor

A data structure used for passing argument types, addresses, and other information between SQL and host language programs. SQL module procedures can specify that parameters be passed by descriptor. Also, SQL uses a descriptor called the SQL Descriptor Area (SQLDA) to communicate with programs about dynamically executed SQL statements (however, note that the SQLDA is not a standard VMS descriptor).

See also SQLDA.

detach (from a database)

The operation by Rdb/VMS that closes the database stream to a database and releases resources associated with the terminated connection. The FINISH statement in both SQL and RDO detaches from the database (in addition to performing other operations).

See also attach.

dictionary

In the most general sense: an overall hierarchical storage facility that includes dictionary directories, subdictionaries, and objects. Dictionary facilities are also called repositories. In the Rdb/VMS documentation, dictionary refers to the CDD/Plus dictionary product.

As a keyword used with the SET and SHOW statements, DICTIONARY has the more limited meaning of the current location within the CDD/Plus dictionary used by the invoked database.

See also CDD/Plus.

dictionary directory

The structure for organizing data descriptions stored in the CDD/Plus dictionary. Dictionary directories are similar in function to VMS directories. They “own” other dictionary directories or dictionary objects.

See also default dictionary directory.

Dictionary Management Utility (DMU)

The utility used with VAX CDD (prior to CDD/Plus) to create and maintain the dictionary directory hierarchy and its associated access control and history lists. DMU is still available with CDD/Plus, but only to work with dictionary definitions

that existed before CDD/Plus was installed on your system. To share definitions among more than one Rdb/VMS database, the metadata must reside in the CDO-format dictionary.

See also Common Dictionary Operator (CDO) utility.

dictionary object

A data definition stored in the CDD/Plus dictionary. Examples of objects include:

- ACMS definitions
- CDD/Plus record definitions
- DATATRIEVE domains, records, procedures, plots, and tables
- VAX DBMS schemas, areas, sets, and records
- Rdb/VMS database entry
- TDMS forms, requests, and request library definitions

Data Distributor does not store any definitions for a target database in a CDD/Plus dictionary; however, you can use the INTEGRATE statement to copy the database definitions of extractions and replications to a CDD/Plus dictionary yourself.

See also CDD/Plus, extraction, replication, and target database.

DIGITAL Command Language (DCL)

See DCL.

directory

A file that catalogs a set of files stored on disk or tape. The directory includes the name, type, and version number of each file in the set.

See also default directory and dictionary directory.

DISTINCT clause

Used in a select expression to find unique values for a column and to eliminate repeating records.

distributed transaction

A transaction that groups more than one database or more than one database attachment together into one transaction even if the databases are located on different nodes. Rdb/VMS uses the two-phase commit protocol of DECdtm services to guarantee that if one operation in a transaction cannot be completed, none of the operations is completed.

See also coordinator, DECdtm services, resource manager, transaction manager, and two-phase commit protocol.

distributed transaction identifier (TID)

An identifier generated by DECdtm services to keep track of a distributed transaction and of the resource managers and transaction managers that are involved in a particular distributed transaction.

See also coordinator, distributed transaction, resource manager, and transaction manager.

distributed transaction processing

The processing of ACMS tasks on two separate nodes, a front end for forms processing and a back end for processing against the database. An ACMS user or task submitter logged in to an ACMS system on one node can select tasks in an application on an ACMS system on another node. ACMS uses the DECnet network to communicate transparently between nodes without rewriting applications or tasks, and can distribute processing between nodes in a VAXcluster, a local area network, a wide area network, or any combination of the three.

See also ACMS and transaction processing.

DML

See data manipulation language (DML).

DMU

See Dictionary Management Utility (DMU). *See also* Common Dictionary Operator (CDO) utility.

domain

The set of values that a table column can have. The CREATE DOMAIN statement specifies the set of values by associating a data type with a domain name. CREATE TABLE statements can use the domain in column definitions.

See also column and data item.

dynamic cursor

A cursor used in dynamic SQL programs. You explicitly specify the cursor name at compile time; however, you do not explicitly specify the **SELECT** statement. Instead, you specify the name of a prepared statement. In other words, the cursor name is known at compile time, but the **SELECT** statement is not known until run time.

See also cursor, dynamic SQL, and extended dynamic cursor.

dynamic interface

Another term for dynamic SQL.

dynamic SQL

A set of special SQL statements (**PREPARE**, **DESCRIBE**, **EXECUTE**, **EXECUTE IMMEDIATE**, and **RELEASE**) and data structures (**SQLCA** and **SQLDA**) that let programs accept or generate SQL statements for which no executable form exists until the program runs. Unlike statements embedded in programs or contained in SQL module files, such dynamically executed SQL statements are not part of any source code but are created while the program runs. Dynamic SQL is useful when programs cannot predict before they execute the type of SQL statement they will need to process.

To dynamically execute statements, programs either embed the special dynamic SQL statements in host language source files for precompilation or call SQL module files that contain procedures with the special statements.

See also client, dynamically executable statement, executable statement, nonexecutable statement, server, and SQL/Services.

dynamically executable statement

An SQL statement that can be formulated and executed at run time.

See also dynamic SQL, executable statement, and non executable statement.

edit string

A character or group of characters that controls how **DATATRIEVE** displays data in a column or domain.

embedded DML

DML statements that are embedded directly in a host language program.

embedded RDO

RDO statements that are embedded directly in a host language program rather than being issued in the interactive RDO utility. Programs that contain embedded RDO must be processed by the preprocessor to convert the embedded RDO statements to a form that is understandable by the host language compiler.

embedded select

Another term for singleton select.

embedded SQL

SQL statements that are embedded in a host language program rather than in an SQL module language procedure or issued in the interactive SQL utility. Programs that contain embedded SQL statements must be processed by the SQL precompiler before compilation and execution to convert the embedded SQL statements to a form that is understandable by the host language compiler.

equijoin

A join operation that matches values in a column from one table with those in a corresponding column in another table.

event

An occurrence of some activity within a facility. DECTrace tracks two types of events: duration and point. **Duration events** have logical beginning and ending points. **Point events** occur instantaneously. DECTrace allows you to collect and report on data based on a set of predefined Rdb/VMS events.

See also DECTrace and security event.

executable image

An image that can be run in a process. When run, an executable image is read from a file for execution in a process.

executable statement

In host language programs, SQL statements that undergo processing during precompilation or module compilation, but do not actually execute until the program runs. In interactive SQL, statements whose operation is processed immediately by SQL.

See also dynamic SQL, dynamically executable statement, and nonexecutable statement.

execution server

A process that executes Application Programming Interface (API) requests for the communication server. There are two classes of execution servers that the communication server can create: generic or database. By default, the communication server uses generic execution servers when a request does not explicitly name a database execution server.

See also communication server, database execution server, generic execution server, and server.

export operation

Makes a copy of a database in an intermediate form. Use an import operation to rebuild an Rdb/VMS database from the intermediate (RBR) file created by the export operation.

See also backup operation, import operation, and restore operation.

expression

Another term for value expression.

extended dynamic cursor

A cursor used in dynamic SQL programs. You supply parameters for the cursor name and the SELECT statement. In other words, the cursor name and select statement are not known until run time. Extended dynamic cursors let you use one set of cursor-related statements to process any number of dynamically generated statements.

See also cursor, dynamic cursor, and dynamic SQL.

extent

The number of pages by which a storage area file or a snapshot file is extended each time its space limit has been reached. The SQL CREATE SCHEMA statement specifies the initial number of pages for each storage area.

See also multifile database (MFDB), snapshot, and storage area.

extraction

In Data Distributor, the process by which complete or partial copies of a source database are transferred to a directory. Data Distributor can transfer a database to a directory on the same node as the source database or to directories on remote

nodes. The source database for a Data Distributor extraction can be any of these database systems: Rdb/VMS, Rdb/ELN, and VIDA. Database copies are in Rdb/VMS format. The target database for an extraction process is not periodically updated
See also Data Distributor, replication, source database, and source node.

field

Another term for column or domain.

See also data item.

field attribute

In RDO, a condition or characteristic of a field in a record.

file

A collection of related records treated as a unit; often referred to by a logical name.

file name

The name you choose to identify a file. The file name can have from 1 to 39 characters selected from the letters A through Z, the numbers 0 through 9, and the underscore (_) or the dollar sign (\$). When you name files, you can use any names that are meaningful to you.

file specification

A name that uniquely identifies a file. A full file specification identifies the node, device, directory name, file name, file type, and version number under which a file is stored.

file type

The part of a file specification that describes the nature or class of file. The file type follows a period after the file name and consists of 1 to 39 characters. The VMS operating system and VAX software products recognize many default file types used for special purposes.

In an Rdb/VMS database, the possible default file types are as follows:

- RDB

The database root file, where information about the characteristics and physical structure of the entire database is maintained. There is one database root file per Rdb/VMS database. In a single-file database, the database root file also contains all the user's data.

- **RDA**

One or more storage area files, where the user's data is located in a multfile database. System relations are stored in the default storage area. The RDA file type applies only to multfile databases.

- **SNP**

One snapshot (SNP) file for each storage area (RDA) file in a multfile database when snapshots are enabled for the database. In a single-file database, there is a single snapshot file.

- **RUJ**

One recovery-unit journal file created for each user who starts a read/write transaction. The recovery-unit journal file resides, by default, in the user's SYS\$LOGIN directory. The RUJ file is deleted after the user detaches from the database.

- **AIJ**

One after-image journal file per database if after-image journaling is enabled.

- **RBF**

A backup file of the database created by the RMU/BACKUP command.

- **RBR**

A backup file of the database created by the RDO EXPORT statement.

- **RRD**

A file containing data definitions for a table or view created by the RMU/UNLOAD command.

- **UNL**

A file containing data created by the RMU/UNLOAD command.

See also multfile database (MFDB).

foreign key

A column in a table that does not uniquely identify rows in that table, but is used as a link to matching columns in other tables.

See also candidate key, primary key, and secondary key.

formal parameter

A parameter declaration in an SQL module procedure that corresponds to an actual parameter in a calling program. The SQL statement in the module procedure uses the formal parameter name to refer indirectly to the actual parameter named in the host language call to the module procedure.

See also actual parameter.

free space

The portion of a database page that is not filled with data.

See also database page.

full path name

A name that uniquely identifies a dictionary directory, subdictionary, or object in the CDD/Plus hierarchy. The full path name is a concatenation of the given names of directories and objects, beginning with the anchor name, ending with the given name of the object or directory you want to specify, and including the given names of the intermediate subdictionaries and directories. The names of the directories and objects are separated by periods.

`SYS$COMMON:[CDDPLUS]CORP.ACCOUNTING.PERSONNEL` is a full path name that uniquely identifies the object `PERSONNEL`, an entry for an Rdb/VMS database definition.

See also logical path name and relative path name.

function

A keyword that calculates a single value based on all the values in a column of a result table or group. SQL uses these functions:

<code>COUNT</code>	Number of rows in a result table
<code>SUM</code>	Sum of the values in a column
<code>AVG</code>	Average of the values in a column
<code>MAX</code>	Largest value in a column
<code>MIN</code>	Smallest value in a column

Other terms for function are aggregate expression, built-in function, and statistical expression.

generic execution server

In SQL/Services, one of two types of execution servers (database and generic servers) that execute Application Programming Interface (API) requests for the communication server. Generic execution servers are assigned by default to execute API requests. Unlike database execution servers, generic servers are not pre-attached to an Rdb/VMS or VIDA database at system startup time. Generic servers, however, require no application or server system modification, as database execution servers require.

See also Application Programming Interface (API), communication server, execution server, database execution server, and server.

global aggregate

An expression that uses field values from one relation to group records from another. A function is then used to calculate a value for the group. For example, you can group salary records in a SALARY_HISTORY relation according to the DEPARTMENT_CODE field in the DEPARTMENT relation. Then you can use the AVERAGE function to find the average salary for each department.

global transaction

See distributed transaction.

handle

See database handle.

hash bucket

A data structure that maintains information about an index key, and a list of internal pointers (or dbkeys) to the page and row that contain the value of the index key.

Another term for hash bucket is hashed index node.

hashed index

In a hashed index, the index key value is converted mathematically to a relative page number in the storage area of a particular table. On that page is a hash bucket or hashed index node that contains pointers (or dbkeys) that point to where the row is actually stored. To find a row using the hashed index, the database system searches the hash bucket, finds the appropriate dbkey, and fetches the table row. Hashed indexes are more effective for random, direct access when the query supplies the entire index key. Hashed index structures are created when a row is stored and a hashed index has been defined for that table.

A hashed index is defined in Rdb/VMS by specifying the **TYPE IS HASHED** clause within the **CREATE INDEX** statement. Hashed indexes are available only in a multifile database.

See also hashing, hash bucket, index, multifile database (MFDB), multisegmented index, and sorted index.

hashed index node

Another term for hash bucket.

hashing

The conversion of a column's primary key value (for example, an **EMPLOYEE_ID** of 00167) into a database page number on which the row will be stored; subsequent retrieval operations that specify the key column value use the same hashing algorithm and can locate the row directly. Hashing provides fast retrieval for data that contains a unique key value.

See also hashed index.

hierarchical database

A type of database that organizes the relationships between record types as a tree structure (usually depicted "upside-down," with branches growing downward and out). A hierarchical database stores related records on the same branch of the tree to make data retrieval efficient.

See also database, network database, relational database, and schema.

host language program

All the host language source files that go together to make up an entire program.

host language source file

A file containing host language source code. Such files can use SQL statements in one of the following ways:

- By using host language calls to procedures contained in an SQL module file compiled by the SQL module processor, independently of the host language source file.
- By embedding SQL statements directly in the file. The statements are flagged by special delimiters. Such host language source files must be processed by the SQL precompiler before the host language statements in them are compiled.

host language variable

A variable declared in a program that the program can refer to in an embedded SQL statement.

See also parameter.

host structure

A host language variable that corresponds to a group construct of several host language variables. Once a program declares a host structure, it can refer to the host structure in some SQL statements instead of listing the host language variables that comprise it.

host variable

See host language variable.

idempotent

A state of being in which database recovery using before-image journal (RUJ) files should succeed following one or more system failures as if the recovery were done completely and successfully the first time.

See also before-image journal and recovery.

import operation

Creates an Rdb/VMS database from an intermediate RBR file. You use the import operation with an export operation to make changes to Rdb/VMS databases that cannot be made any other way.

See also backup operation, export operation, and restore operation.

index

A structure within a file or database that lets you quickly locate particular records based on key column values.

In Rdb/VMS, you can use any column or combination of columns from a row as an index key. You can also define more than one index for a given table. Rdb/VMS has two types of indexes, sorted and hashed. Hashed indexes are available only in a multifile database.

See also cardinality, hashed index, multisegmented index, and sorted index.

index fill factor

A parameter that controls the initial fullness percentage of each index node. Generally, the performance of query-intensive applications benefit from index structures with large, full nodes, while the performance of update-intensive applications can benefit from index structures with small, partially filled nodes.

See also index, index node, and sorted index.

index key

A column of a row in an indexed file or database that determines the order of search and retrieval. You can use any column or combination of columns from a row as an index key. You can also define more than one index for a given table.

See also hashed index, index, and sorted index.

index node

A sorted index data structure that contains key values and pointers (dbkeys) to rows in the database, as well as to other nodes and rows in this sorted index structure. You can control the size of a node in a sorted index by using the `NODE SIZE IS` clause within the `SQL ALTER INDEX`, `SQL CREATE INDEX`, and `IMPORT` statements.

See also hashed index, index, and sorted index.

indicator array

Another term for indicator structure.

indicator parameter

A parameter whose value indicates whether its associated main parameter has been assigned a null value, or whether or not the text string passed from a database has been truncated.

See also main parameter.

indicator structure

A one-dimensional array of host language variables declared with the signed word data type that programs use as indicator variables for host structures. Indicator structures are the only way to specify indicator parameters for host structures.

indicator vector

Another term for indicator structure.

initializing phase

The period beginning at SQL/Services startup time in which the communication server reads the configuration file and creates the process pool of execution server processes. The initializing phase ends when the communication server receives its first message request from a client application at the outset of the working phase.

See also communication server, configuration file, execution server, process pooling, SQL/Services, and working phase.

input parameter

Another term for parameter marker.

Installation Verification Procedure (IVP)

A test procedure that determines whether or not a software product has been installed correctly.

integrity

The correctness of information in an Rdb/VMS database. There are three general types of integrity control:

- Integrity constraints make sure that database information remains correct when users try to modify it incorrectly.
- Concurrency control lets only one user at a time update a file while allowing many users simultaneous access to the database.
- Recovery restores a database to the state it was in before a system failure.

interactive processing (Rdb/VMS)

A mode of computer operation in which the SQL or RDO statements and the data that control the actions of the computer are entered by a person at a terminal.

See also batch processing (Rdb/VMS).

interactive SELECT

Another term for the SELECT statement.

intermediate result table

A temporary result table created by SQL as it evaluates the clauses of a select expression. After each clause, SQL logically produces an intermediate result table that is used to evaluate the next clause.

interpretive call interface

See Callable RDO.

interval

The number of data pages between space area management (SPAM) pages.

See also SPAM page and threshold.

ISO

International Standards Organization. ISO maintains an SQL standard, ISO 9075:1989, and is developing an extended SQL standard, SQL2, in conjunction with ANSI.

IVP

See Installation Verification Procedure (IVP).

join

An operation in which Rdb/VMS retrieves data from more than one table based on matching column values. To process a join operation, Rdb/VMS first forms the Cartesian product of the tables.

See also Cartesian product, equijoin, and reflexive join.

journal file

A file that contains all records modified by a run unit or transaction. The journal file allows reconstruction of the database in the event of corruption due to system or program failures.

See also after-image journal (AIJ), before-image journal, and before-image journaling.

journaling

The process of recording, on a recoverable resource, information about operations on a database. The type of information recorded depends on the type of journal being created.

See also after-image journal (AIJ), before-image journal, and before-image journaling.

key

A column in a row that you use to locate one or more specific rows. Using keys, Rdb/VMS can locate rows in the table directly, without searching sequentially. Identifying keys increases the speed of some database operations.

See also candidate key, foreign key, hashed index, index key, key value, primary key, and sorted index.

key value

The values supplied in a data manipulation language operation to identify a specific row for access.

keyword

A word reserved for use in certain specified syntax formats, usually in a command or a statement. In Rdb/VMS syntax diagrams, keywords are shown in capital letters *and* are underlined.

License Management Facility (LMF)

A software facility that automatically checks for licenses prior to installation of Digital software.

See also Product Authorization Key (PAK) and Service Update PAK (SUP).

line index

A dynamic section of a database page that acts as a directory to data on the page by indexing page offsets of individual data segments.

See also database page.

linker

A program that creates an executable program, called an image, from one or more object modules produced by a language compiler or assembler. Programs must be linked before they can be executed.

list

A large data object with a segmented internal structure, used for storage and retrieval of unstructured data such as graphics data, large amounts of text, or long strings of binary data. The entire list appears as the value of a column. A list has

the **LIST OF BYTE VARYING** or **LIST OF VARBYTE** data type. In RDO, a list is called a segmented string and has the **SEGMENTED STRING** data type.

See also list cursor, **LIST OF BYTE VARYING** data type, **RDB\$LENGTH**, and **RDB\$VALUE**.

list cursor

A result table defined by the **SELECT** expression in an **SQL DECLARE LIST CURSOR** statement in order to scan the individual elements of an SQL list. A list cursor enables programs to perform operations on each element of a list. Each element of a list is stored as the value of an SQL column. In order to reference elements within a row, a list cursor must reference a table cursor because the table cursor provides the row context. A list cursor is used to find the value or length of the segments of unstructured data, such as graphics data, large amounts of text, or long strings of binary data in the **LIST OF BYTE VARYING** data type. There are two types of list cursors: read-only and insert-only. See the sample program **SQL\$RESUMES** for more information on how to request segment length in SQL programs.

In RDO and RDML, segmented strings are retrieved by creating a record stream that finds the contents of the value expressions **RDB\$LENGTH** and **RDB\$VALUE**.

See also list, **LIST OF BYTE VARYING** data type, positioned insert, **RDB\$LENGTH**, **RDB\$VALUE**, record stream, result table, and table cursor.

LIST OF BYTE VARYING data type

A special data type for the storage and retrieval of unstructured data, such as graphics data, large amounts of text, or long strings of binary data. List data type is the SQL term; the comparable RDO term is segmented string data type.

See also list cursor, **RDB\$LENGTH**, and **RDB\$VALUE**.

literal

A value expression that directly specifies a value. Literals can be numeric, character string, or date.

Another term for literal is constant.

LMF

See License Management Facility (LMF).

locking

A mechanism for protecting transactions from interference by other concurrently executing transactions. The mechanism that controls the allocation and deallocation of a resource, such as a record or a process. Rdb/VMS allows locks on individual rows and on all tables in one or multiple storage area files. The **RESERVING** clause of the **SET TRANSACTION** or **DECLARE TRANSACTION** statements can limit the extent of the locking to specific portions of the database.

See also lock type, reserving option, and share mode.

lock type

Part of the reserving option in a **SET TRANSACTION**, **DECLARE TRANSACTION**, or **START_TRANSACTION** statement. The reserving option, if specified, consists of a share mode and a lock type.

The possible share modes are exclusive, protected, and shared. The possible lock types are read and write. For example, a **SET_TRANSACTION** statement might reserve a table, specifying a reserving option of **PROTECTED WRITE**. Another term for lock type is access mode.

See also locking, reserving option, and share mode.

logical area

An internal partitioning of a storage area with uniform page format. Logical areas are used for the storage of tables and indexes. You can think of a logical area as a set of pages with common characteristics. There will be different logical areas within each storage area. Also, if a table is partitioned over multiple storage areas, it will have one logical area per physical storage area. This is most noticeable with the dbkey, which uses the logical area portion to denote which partition contains the record.

logical name

A user-specified name for any portion or all of a file specification. Logical name assignments are maintained in logical name tables for each process, each group, and the system.

See also the **RDM\$. . .** and **RDMS\$. . .** entries for descriptions of some logical names used by Rdb/VMS.

logical operator

Another term for Boolean operator.

logical path name

A logical name that uniquely identifies a dictionary directory, subdictionary, or object in the CDD/Plus dictionary. The logical path name is a name you define for a full or relative path name.

macro

In certain programming languages, such as C, a command, series of commands, or text string that is assigned to a key or word, similar to a DCL logical name.

main parameter

A parameter that contains the value to be used in an SQL statement.

See also indicator parameter.

map

See storage map.

MBLR

See metadata binary language representation (MBLR).

message file

A file that contains a table of message symbols and their associated text.

message protocol

The internal message-packet protocol used in SQL/Services client/server network communications.

metadata

Data that is used to describe other data. Data definitions are sometimes referred to as metadata. Examples of metadata include schema, table, and column definitions.

See also data definition language (DDL).

metadata binary language representation (MBLR)

The binary language representation of metadata manipulation. MBLR specifies the actions that the database must perform to change the metadata. In other words, the binary form of data definition language.

See also binary language representation (BLR), data definition language (DDL), and metadata.

MFDB

See multfile database (MFDB).

missing value

In RDO syntax, a special value retrieved when a column of a row in a table is null. Missing values signify in displays and to programs that there is no value stored.

SQL does not use missing values. However, you can specify default values. If you do not specify a default value, NULL is displayed in interactive SQL when it encounters null values. In programs, SQL requires the use of indicator parameters to handle the possibility of null values. Default values in SQL are not the same as missing values in RDO.

See *also* default value and null value.

mixed page format

A storage area format that allows rows from more than one table to reside on or near a particular page of the storage area file.

See *also* interval, SPAM page, threshold, and uniform page format.

module file

Another term for SQL module file.

module language

Another term for SQL module language.

module procedure

Another term for an SQL module procedure.

module processor

Another term for SQL module processor.

multifile database (MFDB)

A database in which the database root file (default file type RDB) contains only metadata and system information; all data is stored in one or more storage area files. The files in a multifile database can be on the same disk or on multiple disks. (Careful placement of files on different disk drives can reduce input/output contention and substantially improve application performance.)

See *also* allocation, extent, file type, hashed index, storage area, and storage map.

multisegmented index

An index with a key comprised of more than one column.

See also hashed index, index, and sorted index.

multithreaded

In SQL/Services, a characteristic of a server process that enables it to handle multiple Application Programming Interface (API) requests simultaneously. The communication server can process multiple API requests simultaneously because incoming requests can be processed before the execution of previous requests is complete.

National Character Set (NCS) utility

A utility that provides a way to define collating sequences, register them as definition modules in an NCS library, and make them locally accessible to programmers. NCS collating sequences are selective subsets of the DEC Multinational Character Set, and are frequently used in international applications. For example, a Spanish collating sequence would resolve sorting characters you might encounter when processing strings from the Spanish language.

The SQL and RDO interfaces to Rdb/VMS use the NCS collating sequences to specify collating sequences and languages.

See also collating sequence.

NCS

See National Character Set (NCS) utility.

network database

A database model that establishes relationships between records using sets. A single record can participate in any number of sets, so you can relate it to any other record in the database, not just those above and below it in a hierarchy.

Network databases are also called CODASYL databases. VAX DBMS is a network database.

See also CODASYL, database, hierarchical database, relational database, and schema.

nonexecutable statement

In host language programs, statements that SQL processes completely when it precompiles a program (DECLARE statements, INCLUDE and WHENEVER) or compiles an SQL module (DECLARE statements only). In interactive SQL, statements for which the operation controlled by the statement does not occur until SQL encounters an executable statement (DECLARE TRANSACTION and DECLARE CURSOR).

See also executable statement.

normalization

The process that reduces a database structure to its simplest form and eliminates data redundancy. Normalization physically separates related concepts in the database into separate tables or rows. A data item is stored only once and requires only one update operation to change it.

null value

The absence of a value. If a particular column of a row in a table is null, that means there is no value stored. In RDO syntax, a null value is called a missing value.

numeric data type

A characteristic assigned to a column that indicates column values are to be considered numbers rather than text.

object

A passive repository of information to be protected. In Rdb/VMS, an object is a database or schema, table, view, or column.

See also dictionary object and security.

OLTP

See online transaction processing (OLTP).

online transaction processing (OLTP)

A technique for organizing multiuser, high-volume, online applications that provides control over user access and updates of data.

See also ACMS and transaction processing.

optimizer

See query optimizer.

outer query

The top-level select expression.

See also column select expression.

outer reference

A reference in a subquery to a table specified in an outer query that contains the column select expression (or subquery).

output parameter

In dynamic SQL, another term for a select list item.

See also select list.

page header

A fixed-length section at the beginning of the database page that contains page and storage area information.

See also database page.

PAK

See Product Authorization Key (PAK) and Service Update PAK (SUP).

parameter

A variable declared in a host language program that is associated with an SQL statement. The meaning of parameter encompasses host language variables named directly in embedded SQL statements, actual and formal parameters in programs that use SQL module language, and parameter markers in the statement string of a PREPARE statement.

See also actual parameter, formal parameter, host language variable, indicator parameter, main parameter, and parameter marker.

parameter marker

A question mark (?) in the statement string of a PREPARE statement. In dynamic SQL, the question marks in the statement string of a PREPARE statement serve the same purpose as host language variables in an SQL statement embedded in a program (or as formal parameters in a SQL statement that is part of an SQL module procedure). SQL writes information about the number and data types of any

parameter markers to the SQLDA when it executes a DESCRIBE . . . MARKERS statement. The program uses the information to declare host language variables corresponding to the markers and puts values in the variables that SQL uses when it dynamically executes the prepared statement.

parameter specification

Another term for parameter.

partial path name

See relative path name.

path name

In the data dictionary, a name that begins with the user's CDD/Plus anchor and ends with the given name of a dictionary directory or object. The path name includes names of intervening dictionary directories, for example:

```
SYS$COMMON:[CDDPLUS]PERSONNEL.DEPARTMENTS
```

You can have full, logical, and relative path names.

See also full path name, logical path name, and relative path name.

placeholder

Used in VAX Language-Sensitive Editor (LSE) templates, placeholders represent the SQL syntax you need to define SQL elements and data dictionary objects, and to use SQL. When you expand a placeholder, LSE provides the required syntax or indicates optional elements. You can expand a placeholder into:

- The required SQL syntax elements that are appropriate for that context
- Optional elements
- Tokens that represent appropriate keywords or information to be supplied
- Other placeholders

PLACEMENT VIA index option

A row placement option in which Rdb/VMS uses the index value in determining the database page on which to store rows. The PLACEMENT VIA index option is declared in the storage map definition.

See also hashed index, sorted index, and storage map.

plan file

Another term for context file.

platform

The combination of computer hardware and operating system software.

See also server.

pointer

A place marker that identifies a row's address in a storage area.

See also dbkey.

pointer variable

A variable that provides indirect access to storage by storing the address of data instead of directly storing data in the variable. To take full advantage of dynamic SQL, host languages must support pointer variables.

positioned insert

A special form of the SQL INSERT statement used for putting a row into a table cursor. You must use a positioned insert to prepare for storing data values in a list. When you store the row data, the positioned insert places the cursor on the table row where the list is stored, so that you can use subsequent INSERT statements to insert data into the individual list elements.

See also list cursor.

precompiled SQL

Another term for embedded SQL.

precompiler

Another term for SQL precompiler.

predicate

A condition that SQL evaluates as either true, false, or unknown. Predicates compare value expressions or result tables with different conditional operators. In SQL statements, predicates follow the WHERE or HAVING keywords. In RDO, a predicate is called a conditional expression. Other terms for predicate are Boolean expression, conditional expression, and search condition.

See also conditional operator.

prepare phase

The first phase of the two-phase commit protocol. During the prepare phase, the coordinator of the distributed transaction asks each resource manager involved in the distributed transaction whether or not it is prepared to commit the changes to the databases. If the coordinator receives yes responses from *all* the resource managers, the coordinator instructs the participants in the distributed transaction to enter the commit phase.

See also commit phase, coordinator, distributed transaction, resource manager, and two-phase commit protocol.

prepared statement

An SQL statement that has been processed with the PREPARE statement and that can be dynamically executed.

preprocessor

Another term for precompiler.

primary key

A column in a table whose value uniquely identifies its row in the table. It cannot be a null value.

See also candidate key, foreign key, and secondary key.

print list

One or more value expressions (including the names of elementary and group fields) whose values you want Rdb/VMS to display.

privilege

The ability to access a file or other resource for a certain purpose.

See also access privilege set (APS), ACL-style protection, and ANSI-style protection.

procedure

A general-purpose routine, entered by means of a call instruction, that uses an argument list passed by a calling program and only local variables for data storage. A procedure is entered from and returns control to the calling program. An example of a procedure is an SQL module procedure.

An SQL module procedure contains a series of SQL statements. These statements can be executed with the execute (@) command.

See also SQL module procedure.

procedure parameter

Another term for formal parameter.

process

An environment from which a VMS user can issue commands. These commands can be issued either in interactive or noninteractive mode. A user automatically creates an interactive process when he or she logs in to a VMS system. The user can then create subordinate processes, such as batch jobs, that execute independently of the interactive process. A user authorization file (UAF), usually maintained by a system manager, specifies characteristics that are associated with any process created by that user. Among those characteristics are device and VMS directory defaults, and a set of privileges and resource quotas that define what the process can use on the system.

process pooling

In SQL/Services, the feature that allows Application Programming Interface (API) requests on the client system to be executed on the server system by the execution server processes that comprise the process pool. The communication server starts a predefined set of execution server processes during the initializing phase at SQL/Services system startup time. The number of execution server processes created for the process pool depends on the number of definitions specified in the configuration file. Process pooling decreases API request response time and reduces overall system resource utilization.

See also Application Programming Interface (API), communication server, configuration file, and execution server.

Product Authorization Key (PAK)

A unique paper key provided to customers with the necessary information needed to register and use a layered product. Rdb/VMS requires a key for successful installation and use.

See also License Management Facility (LMF) and Service Update PAK (SUP).

project operation

See reduction operation.

qualifier

A portion of a command string that modifies a command verb or command parameter. A qualifier follows the command verb or parameter to which it applies and has the following format:

/qualifier[=option]

query header

A substitute column header that you define to replace the field name when DATATRIEVE displays values from a field. For example, you might want to define the query header "Status" to appear at the top of the column of values from the field EMPLOYEE_STATUS. You can specify a query header with the RDO CHANGE FIELD or DEFINE FIELD statement.

query name

A synonym you give to a DATATRIEVE field name in order to make input easier to type and remember. For example, to make it easier to write DATATRIEVE statements about the field SECTION_NUMBER, you can define the query name NUM and substitute it for the full field name. You can specify a query name with the RDO CHANGE FIELD or DEFINE FIELD statement.

query optimizer

The software component of Rdb/VMS that evaluates each query and determines the most efficient means of accessing the data requested.

query specification

Another term for select expression.

quiet point

A time when a run unit is not accessing any database areas. Quiet points occur between transactions.

See also run unit and transaction.

RdbExpert

A product you can use to optimize the physical design of your Rdb/VMS database. Using RdbExpert, you specify information about the application workload, data volume, and system environment of the database. RdbExpert applies its design rules to the schema and to the information you have supplied. It generates several design reports, as well as a command procedure that (with minimal edits) you can

run to create a new database with an optimal physical design. This procedure also unloads any existing data and reloads it in the new database.

Rdb/ELN

A Digital relational database management product designed for real-time applications on systems running in the VAXELN run-time application environment. Rdb/ELN uses the relational model of database organization.

Rdb/VMS

A relational database management system. The SQL interface is a component of Rdb/VMS that allows users to define and query an Rdb/VMS database using standard SQL syntax.

See also SQL.

Rdb/VMS Management Utility (RMU)

A DCL-level Rdb/VMS utility that allows database administrators to:

- Analyze how disk space is being used by the database
- Convert a previous version of the database to the current version's format
- Display the contents of Rdb/VMS database files
- Control the Rdb/VMS monitor process
- Display information about current Rdb/VMS database users and database activity statistics
- Back up and restore Rdb/VMS databases
- Truncate the after-image journal (AIJ) file and create a secondary, backup copy of the AIJ
- Load and unload database files
- Verify the integrity of Rdb/VMS databases
- Open and close an Rdb/VMS database

RDB\$LENGTH

A special Rdb/VMS value expression for the length in bytes of a segment of an RDO segmented string. RDB\$LENGTH can be referenced as LENGTH in the RDML precompiler.

See also list, list cursor, LIST OF BYTE VARYING data type, and RDB\$VALUE.

RDB\$VALUE

A special Rdb/VMS value expression for the value stored in a segment of an RDO segmented string. RDB\$VALUE can be referenced as VALUE in the RDML precompiler.

See also list, list cursor, LIST OF BYTE VARYING data type, and RDB\$LENGTH.

RDM\$BIND_BUFFERS

A logical name Rdb/VMS recognizes that lets you specify a NUMBER OF BUFFERS to be used at run time that is different from the defined default. This can be a very powerful tool for tuning specific applications. For example:

```
$ DEFINE RDM$BIND_BUFFERS 100
```

RDM\$BUGCHECK_DIR

A logical name Rdb/VMS recognizes that lets you redirect the location of bugcheck files from the SYS\$LOGIN directory to another location. This can be useful when SYS\$LOGIN does not have enough space for bugcheck files. For example:

```
$ DEFINE/SYSTEM/EXEC RDM$BUGCHECK_DIR DISK12:[BUGCHECK_DIR]
```

RDML

See Relational Data Manipulation Language (RDML).

RDMS\$BIND_SEGMENTED_STRING_BUFFER

A logical name Rdb/VMS recognizes that may let you increase the efficiency of applications that manipulate segmented strings by increasing the buffer space for segmented strings. The default value is 10,000 bytes. For example:

```
$ DEFINE RDMS$BIND_SEGMENTED_STRING_BUFFER 20000
```

An adequate buffer size is needed to store large segmented strings (using segmented string storage maps) in storage areas other than the default RDB\$SYSTEM storage area.

See also list.

RDMS\$BIND_SORT_WORKFILES

A logical name that specifies how many work files the VMS Sort utility (SORT) is to use if work files are required. The default is 2 (the SORT default) and the maximum is 10.

The work files can be individually controlled by the SORTWORK n logical names (where n is from 0 to 9).

RDMS\$BIND_WORK_FILE

A logical name Rdb/VMS recognizes that lets you redirect the location of the temporary tables that Rdb/VMS sometimes creates for use in matching operations to a disk structure other than SYS\$LOGIN (the default). For example:

```
$ ! Assign the work area to another disk with read/write access:  
$ DEFINE RDMS$BIND_WORK_FILE WORK$DISK:[RDB.WORK]  
$ DIRECTORY/PROTECTION RDMS$BIND_WORK_FILE
```

```
Directory WORK$DISK:[RDB.WORK]
```

```
LATENT.FIL;1 (RWED,RWED,RWED,RWED)
```

```
Total of 1 file.
```

RDMS\$BIND_WORK_VM

A logical name Rdb/VMS recognizes that may let you reduce the overhead of disk input/output on matching operations when temporary tables are used to perform this operation. By redefining this logical, you can specify the amount of virtual memory (VM) that will be allocated to your process for use in matching operations. Once the allocation is exhausted, additional data values will be written to a temporary file on disk (SYS\$LOGIN, if RDMS\$BIND_WORK_FILE is undefined). The default VM is 10,000 bytes. For example:

```
$ DEFINE RDMS$BIND_WORK_VM 20000
```

RDMS\$KEEP_PREP_FILES

A logical name Rdb/VMS recognizes that lets you prevent the RDBPRE preprocessor from deleting the intermediate (file type MAR) and language files. This can be helpful when you are trying to debug an RDBPRE program and need to see the language files to do so. For example:

```
$ DEFINE RDMS$KEEP_PREP_FILES YES
```

RDMS\$RUJ

A logical name Rdb/VMS recognizes that lets you locate the recovery-unit journal (RUJ) file on a different disk and directory from the default (SYS\$LOGIN). This can be helpful in reducing contention in that directory. For example:

```
$ DEFINE RDMS$RUJ USERS1:[CLIENTS.JOURNAL]
```

RDO

See Relational Database Operator (RDO).

RDO command procedure

A sequence of Rdb/VMS statements stored in a text file; sometimes referred to as an RDO procedure.

See also DCL command procedure and SQL command procedure.

record

Another term for row.

record locking

See row locking.

Record Management Services (RMS)

A set of VMS operating system procedures that programs can call to process files and records within files. RMS lets programs issue GET and PUT requests at the record level (record input/output) as well as read and write blocks (block input/output). RMS is an integral part of the VMS system software and is used by high-level languages, such as VAX COBOL and VAX BASIC, to implement their input and output statements.

record selection expression (RSE)

In RDO and RDML, a set of conditions that individual records (rows) of a relation (table) must meet before being included in a record stream (result table).

See also column select expression, record stream, result table, select expression, and select operation.

record stream

In RDO and RDML, a temporary group of related records (rows), formed by a record selection expression (RSE), from a relation (table). Streams are used in an application program or with RDO to retrieve one record at a time for manipulation. In SQL terminology, streams correspond to result tables that consist of entire rows from their source tables. Another term for record stream is stream.

See also record selection expression (RSE).

recovery

The process of restoring a database to a known condition after a system or program failure.

See also after-image journal (AIJ), before-image journal, idempotent, journal file, journaling, and transaction.

recovery-unit journal (RUJ)

See before-image journal.

reduction operation

In RDO, an operation that finds the unique values for a field or group of fields and eliminates repeated records. You use the **REDUCED TO** clause to perform the operation. The **REDUCED TO** clause is the RDO equivalent to the SQL **DISTINCT** clause. Another term for reduction operation is project operation.

referential integrity

A database state where every foreign key value matches some value for its associated primary key.

See also table-specific constraint and trigger.

reflexive join

An operation that joins a relation to itself.

See also Cartesian product.

relation

Another term for table. The term *relation* is used in RDO and RDML documentation.

relation-specific constraint

See table-specific constraint.

relational database

A database model that represents data as a set of independent tables. Within a table, data is organized in columns and rows, with no more than one data item occupying each intersection. Relationships between tables depend on values within the tables. In RDO and RDML documentation, tables are called relations, columns are called fields, and rows are called records.

See also database, hierarchical database, network database, and schema.

Relational Database Operator (RDO)

An interactive utility for maintaining Rdb/VMS databases, creating and modifying definitions of database elements, and storing and manipulating data. The RDO utility can also be used to access a VIDA database and to perform Data Distributor operations. The SQL interface to Rdb/VMS provides largely parallel features to RDO.

See also Callable RDO.

Relational Data Manipulation Language (RDML)

A data manipulation language for VAX C and VAX Pascal programs that access Rdb/VMS and Rdb/ELN databases. Programs written in RDML are processed by the RDML preprocessor, which converts the RDML statements into a series of calls to the database. Following successful preprocessing, the programmer can submit the resulting source code to the host language compiler. The SQL interface to Rdb/VMS provides largely parallel features to RDML.

relational join operation

See join.

relational operator

Another term for conditional operator.

relative path name

The shortened form of a dictionary path name. It includes only the parts of the path name that follow the default CDD/Plus directory name. You can use either the full path name or the relative path name to refer to directories, subdictionaries, and objects in a CDD/Plus dictionary.

remote server

The part of Rdb/VMS that lets you access data on other computers on your DECnet network. If, for example, you are using the computer VACKS1 and you type `DECLARE SCHEMA FILENAME "TRIXIE::DISK1:[TOP] PERSONNEL"`, Rdb/VMS instructs the remote server to log in to an account on TRIXIE, which then processes your statements.

replication

In Data Distributor, the process by which a copy of a complete or partial database is first transferred from a source database on the source node to a target database either on the source node or on a remote node. Then, the database in the target directory is periodically updated when any changes are made to that portion of the source database that was originally copied to the target database. The source replications database must be an Rdb/VMS database. Note that the target databases for replications are periodically updated, while those for extractions are not.

See also Data Distributor, extraction, source database, source node, target database, and target node.

request

An optional clause of certain data manipulation language (DML) statements. The request clause lets you specify the transaction or request to be affected by the DML statement. A request can include a transaction handle, a request handle, or both.

See also data manipulation language (DML), request handle, and transaction handle.

request handle

A variable that uniquely identifies a request.

reserving option

The sharing and locking characteristics that Rdb/VMS applies to row access operations during a transaction. The reserving option consists of a share mode (exclusive, protected, or shared) and a lock type (read or write). The reserving option is specified or defaulted when the transaction is started. Another term for share mode is allow mode. Another term for lock type is access mode.

See also lock type and share mode.

resource manager

A component of a distributed transaction. The resource manager is a database management system, such as Rdb/VMS. The resource manager is responsible for maintaining and recovering its own resources. From the perspective of the application, the resource manager is a single attachment to an Rdb/VMS database. In addition, other database products that support the two-phase commit protocol can be resource managers.

See also coordinator, DECdtm services, distributed transaction, transaction manager, and two-phase commit protocol.

restore operation

An operation that rebuilds a database from a saved backup file or copy. Usually, you restore a saved backup file or copy after a hardware or software failure.

There are two types of restore operations in Rdb/VMS:

- **RMU/RESTORE**

Use the **RMU/RESTORE** command from DCL level to restore a database backup file (default file type RBF) created by the **RMU/BACKUP** command.

- **IMPORT**

Use the **IMPORT** statement in SQL to migrate a database from one database management system to another, or to restructure a database.

Migration changes include taking an existing Rdb/VMS database, backing it up, and then using the **IMPORT** statement to convert the internal data structures so that they are compatible with a newly installed version of Rdb/VMS.

Restructuring changes include taking a single-file database, backing it up using the **EXPORT** statement, and then using the **IMPORT** statement to define one or more storage areas in a new multifile database.

See also backup operation, export operation, and import operation.

restrict

See select operation.

result table

A temporary table of values derived from columns and rows of one or more tables that meet conditions specified by a select expression.

RMS

See Record Management Services (RMS).

RMU

See Rdb/VMS Management Utility (RMU).

roll back

To cancel any changes to the database made during the current transaction. Use the **ROLLBACK** statement to roll back changes.

See also commit and transaction.

rollforward

The process of using an after-image journal to restore a database to a known state. This process replaces updates to the database that were lost because a system, program, or disk failure required the installation of backup media.

See also recovery.

root file

See database root file.

row

The horizontal dimension of a table composed of a set of columns that contain one data item each.

row locking

A process by which a database management system reserves a row or set of rows for use by one user. This process can limit or prevent access to those rows by other users. Row locking helps guarantee the consistency of data.

RSE

See record selection expression (RSE).

RUJ

Recovery-unit journal.

See before-image journal.

run unit

An execution of a single program that accesses a database.

See also quiet point and transaction.

scalar expression

Another term for value expression.

scale factor

The power of 10 to multiply by when storing a value with a fixed number of decimal places in an exact numeric data type. For example, the scale factor for a dollar amount such as \$999.99 is two.

schedule definition

In Data Distributor, a definition that specifies when a particular transfer definition is executed by the transfer monitor. You can also define the frequency of transfer if you need to transfer records or updates to existing records on, for example, a daily or weekly basis. You create schedule definitions using either the **CREATE SCHEDULE** statement of interactive SQL or the **DEFINE SCHEDULE** statement of interactive RDO.

See also Data Distributor, Relational Database Operator (RDO), transfer database, transfer definition, and transfer monitor.

schema

The data definitions that comprise a database created with SQL data definition statements. The SQL **CREATE SCHEMA** statement lets you specify in a single SQL statement all data and privilege definitions for a new schema (you can also add definitions to the schema later).

See also database, hierarchical database, network database, and relational database.

schema element

Any **CREATE** statement or a **GRANT** statement embedded within a **CREATE SCHEMA** statement.

schema name

A name that users or programs supply to identify a schema and the database system files associated with it.

The schema name is a VMS file specification or a data dictionary path name.

scientific notation

A way of expressing a very large or very small number as a constant multiplied by the appropriate power of 10. For example:

.000000009	.9E-8 (9 times 10 to the power of -8)
9000000.	.9E 7 (9 times 10 to the power of 7)

scope

The scope of a transaction refers to the statements beginning with the executable statement that starts the transaction and ending with the **COMMIT** or **ROLLBACK** statement that completes the transaction.

The scope of a dbkey refers to how long the database system guarantees that a particular row's dbkey will point to that row only and not be reused even if the row is deleted. Scope is also used as a standard term from programming languages that refers to the part of a program where particular declarations can be seen.

See also dbkey and transaction.

search condition

Another term for predicate.

secondary key

Any key other than the primary key.

See also candidate key, foreign key, and primary key.

security

The protection of the information stored in a database against unauthorized reading, writing, or deleting.

See also access privilege set (APS), access control list (ACL), ACL-style protection, ANSI-style protection, object, privilege, and subject.

security alarm

One-time notifications of Rdb/VMS security events that are sent to all terminals enabled as security operators. Security alarms are triggered by the occurrence of an event previously designated as worthy of the alarm because of its security implications.

See also security audit, security event, and security operator terminal.

security audit

In Rdb/VMS, the record or recorded history of Rdb/VMS security events that are written to the security audit journal file. The writing of security audits are triggered by the occurrence of an event previously designated as worthy of the audit because of its security implications. Security audit is also referred to as audit trail.

See also security event and security audit journal.

security audit journal

The record of audit activities (security events) that is created by Rdb/VMS when the audit feature is enabled for the database by the RMU/SET AUDIT command. The security audit journal is sometimes referred to as the audit journal, audit trail, or security journal.

See also security alarm, security audit, and security event.

security event

Some operation (in SQL, RDO, or RMU) that affects the security of the objects in the database. A security event is sometimes referred to as an event or as an audit event.

See also security audit and security audit journal.

security manager

The person or persons responsible for protecting the security of the database. This role is sometimes performed by the same person who functions as the database administrator. It requires the same skills as the DBA, but includes additional privilege (the SECURITY privilege) as well as knowledge of the security features provided with an Rdb/VMS database.

See also privilege and security.

security operator terminal

A class of terminal that has been enabled to receive security alarms. Normally, such a terminal is a hardcopy terminal in a protected room. The output provides a log of security-related events and details that identify the source of the event.

See also security alarm and security event.

segmented string

The RDO term for a list.

See also list, RDB\$LENGTH, and RDB\$VALUE.

segmented string data type

A special RDO data type designed to handle large pieces of unstructured data, such as graphics data, large amounts of text, or long strings of binary data, with a segmented internal structure. The maximum size of a string segment is 64K bytes.

See also list, RDB\$LENGTH, and RDB\$VALUE.

segmented string identifier

A number that points to the location of the segmented string or list. This number, rather than the contents of the list, is stored in the table column when you create a table that contains a list. In SQL, you must use cursors to retrieve or store the contents of a list. In RDO and RDML, you must use a FOR loop to retrieve or store the values for RDB\$LENGTH and RDB\$VALUE.

See also list.

select expression

The fundamental element in SQL syntax that is the basis of the SELECT, DECLARE CURSOR, CREATE VIEW, and INSERT statements. Select expressions specify a result table derived from some combination of the tables or views identified in the FROM clause of the expression.

select list

A list of column names and value expressions in a select expression. The select list specifies the columns in the result table for the select expression.

select operation

An operation that chooses from tables those rows that satisfy a conditional expression. For example, if you want to display employee names with salaries greater than \$20,000, a selection operation eliminates employee rows with salaries less than or equal to \$20,000 from the output.

See also record selection expression (RSE).

SELECT statement

A select expression with an optional ORDER BY clause.

sequential file

An RMS file whose records appear in the order in which they were originally written. A sequential file does not have an index.

server

A service provider. In SQL/Services, the server is a process present on all VMS systems running Rdb/VMS V3.1 (or higher) that provides access to databases with application programs running on various platforms.

See also Application Programming Interface (API), client, dynamic SQL, platform, and SQL/Services.

Service Update PAK (SUP)

A PAK for customers who already have service contracts with Digital.

See also License Management Facility (LMF) and Product Authorization Key (PAK).

shadowing

For table rows, a storage strategy to optimize input/output operations that is used when a parent table has a much smaller cardinality than the related child table. This storage strategy uses a hashed index placement index to place parent rows with its hash buckets in one storage area along with the hash buckets of the second table. When the rows of the child table are placed in the second storage area, they are clustered (shadowed) in the vicinity of where the hash buckets in the first storage area are located. When sufficient space is allocated for storing the child rows and the child rows are stored in a uniform storage area, then free space will be available to maintain the clustering effect of the child rows and result in a minimum of two input/output operations to retrieve a child row. Shadowing serves to physically group rows together to minimize input/output operations and optimize query strategies.

See also hash bucket.

share mode

The degree of sharing that Rdb/VMS will permit when another user attempts to access a table that you have reserved for a transaction. The share mode is part of the reserving option, which is specified or defaulted for each table at the start of the transaction. The possible share modes are exclusive, protected, and shared. Another term for share mode is allow mode.

See also lock type and reserving option.

single-file database

Databases that have a combined root and storage area file. By default, a CREATE SCHEMA statement creates a single-file database. The absence of a CREATE STORAGE AREA clause in a CREATE SCHEMA statement is what makes the created database single file.

singleton select

A SELECT statement that must specify a result table no larger than one row. A singleton select includes an additional clause, INTO, to assign the values in the row to host language variables in a program. SQL allows singleton select statements only within a host language program. Another term for singleton select is embedded select.

snapshot

A consistent view of the data within the database at a selected time. Initiated in an Rdb/VMS database when a user starts a read-only transaction.

See also deferred snapshots.

sort key

The columns or value expressions in an ORDER BY clause according to which SQL sorts an intermediate result table. The first column or value expression in the ORDER BY clause is the major sort key. Subsequent columns or value expressions are minor sort keys. SQL groups all rows with the same value for a sort key together.

See also ascending order and descending order.

sorted index

A tree structure (B-tree) of nodes that the database system navigates by reading nodes on progressively lower-level branches of the tree until it finds the record that contains the location (or dbkey) of a particular row.

See also B-tree, dbkey, hashed index, index, key, key value, and multisegmented index.

SORTWORKn (where n = 0 to 9)

Logical names VMS recognizes that let you increase the efficiency of Rdb/VMS sort operations, which use the VMS Sort utility (SORT), by assigning the location of the temporary sort work files to different disks. These assignments are made by using the logical names SORTWORK0, SORTWORK1, and so forth. Normally, SORT places work files in the user's SYS\$SCRATCH directory. By default, SYS\$SCRATCH is the same device and directory as the SYS\$LOGIN location.

Example:

```
$ DEFINE SORTWORK0 $222$DUA10:[SORT.TEMP]
$ DEFINE SORTWORK1 $222$DUA11:[SORT.TEMP]
```

source database

In Data Distributor, a single, centralized database that is intended to support the information requirements of users throughout a network.

See also Data Distributor, extraction, replication, source node, target database, and target node.

source node

In Data Distributor and Rdb/VMS, the node in a network on which the source database resides.

See also Data Distributor, extraction, replication, source database, and target node.

space area management page

See SPAM page.

SPAM page

A special database page (common to both uniform page format and mixed page format storage areas) that stores the fullness thresholds of each data page in the storage area, for a specified interval of pages. The space area management (SPAM) page and also contains a list of clumps and the logical areas to which the clumps belong.

See also clump, interval, mixed page format, threshold, and uniform page format.

spooling

The way in which output to slow devices is placed into queues on faster devices to wait for transmission to the slower devices.

SQL\$DATABASE

The logical name SQL uses to declare the default schema. If a user or program does not issue an explicit DECLARE SCHEMA statement, SQL attempts to declare a default schema using the file specification assigned to SQL\$DATABASE.

SQL

Structured query language. A data definition and data manipulation language for relational databases. Variations of SQL are offered by most major vendors for their relational database products. The SQL interface to Rdb/VMS provides a user interface to other database products. ANSI Standard X3.135-1989 specifies the syntax and semantics approved for SQL by the American National Standards Institute.

SQL2

An extension to the SQL standard under development by the American National Standards Institute and the International Standards Organization.

SQLCA

SQL Communications Area. A host structure SQL uses to provide information about the execution of SQL statements to application programs. The SQLCA shows whether or not a statement was successful, and for some errors, the particular error when a statement is not successful.

SQLCODE

An integer parameter whose value indicates the error status returned by the most recently executed SQL statement. A positive value indicates a warning, a negative value an error, and zero indicates successful execution. The SQLCA contains an SQLCODE field.

SQLDA

SQL Descriptor Area. A collection of host language variables used only in dynamic SQL programs. The SQLDA provides information about dynamic SQL statements (the number and data type of a prepared statement's parameter markers and select list items) to the program, and information about memory allocated by the program (the addresses of host language variables that correspond to parameter markers and select list items) to SQL.

See also descriptor.

SQL command procedure

A sequence of SQL statements stored in a text file; sometimes referred to as an SQL procedure.

See also DCL command procedure and RDO command procedure.

SQL module file

The file containing SQL module language elements to be compiled by the SQL module processor. An SQL module file includes one or more procedures that contain an SQL statement. The procedure can be called from a host language program.

SQL module language

Special keywords and syntax that allow procedures that contain SQL statements to be called from host languages that are not supported by the SQL precompiler.

SQL module procedure

A construct within an SQL module file named in host language calls to the module. SQL module procedures contain formal parameter declarations and an SQL statement that executes when the procedure is called by a host language program.

SQL module processor

The component of SQL that compiles SQL module files. The SQL module processor produces object (OBJ) files that can be linked with object files produced by host language compilers.

SQL precompiler

The SQL utility that converts embedded SQL statements in host language programs to a form understandable by the host language compiler. The precompiler also invokes the host language compiler and creates an intermediate object file ready for linking.

SQL/Services

A client/server component of Rdb/VMS that allows application programs running on various types of computers to access databases that use the dynamic interface to SQL.

See also client, dynamic SQL, server, and SQL.

static SQL

Precompiled SQL statements that do not use dynamic SQL.

statistical expression

Another term for function.

storage area

A subdivision of the database, named in the CREATE STORAGE AREA clause within a CREATE SCHEMA statement. This clause also determines the physical characteristics of the storage area or areas. Subsequent CREATE STORAGE MAP statements associate the storage area with particular tables in the schema. Only multifile Rdb/VMS databases contain storage areas as physical VMS files separate from the database root file (default file type RDB). Each storage area has a default file type of RDA, and each storage area has a corresponding snapshot file with a default file type of SNP.

See also allocation, extent, multifile database (MFDB), and storage map.

storage map

A definition that associates a table with one or more storage areas in a multifile database. The CREATE STORAGE MAP statement specifies a storage map that controls which rows of a table are stored in which storage areas.

When you create a storage map, you specify the table name and the storage area or areas in which rows from the table are to be stored; you can also specify whether or not an index will be used to choose the location for storing the row and whether or not data compression will be enabled when rows are stored in the table.

See also multifile database (MFDB) and storage area.

stream

Another term for record stream.

structure

In the *VAX Rdb/VMS Guide to Using SQL* and *VAX Rdb/VMS SQL Reference Manual*, refers to a parameter that contains other fields. (Unlike the case of repeating items that make up one dimension in an array, the basic elements in a structure do not need to be identically defined.) In some host languages, the term *record* and the term *structure* refer to the same declaration. In other languages, such as FORTRAN, you declare structures and records separately. If the language you are using declares records separately from structures, you specify record names in precompiled SQL statements. In this case, the fields listed or implied by the record declaration constitute the structure to SQL.

subdirectory

A list of files that is grouped one or more levels below the top-level or main VMS directory.

subject

An active entity that gains access to information on behalf of people. In Rdb/VMS, a subject is an attachment to the database. An attachment can be either an interactive user or an application program.

See also object and security.

subpool

In SQL/Services, a set of either generic or database execution server processes. Multiple subpools comprise the process pool. Definitions in the configuration file specify the characteristics of a subpool. The communication server creates one subpool for every definition in the configuration file.

See also communication server, configuration file, database execution server, execution server, and generic execution server.

subquery

Another term for column select expression.

subselect

Another term for select expression.

substring

A value expression made up of a character string that has a specified start position and an optionally specified length. The characters in the substring are numbered beginning with the number 1.

See also value expression.

system manager

A VMS user responsible for the overall operation of an ACMS or VMS system. Duties of the system manager include authorizing all users of the system, setting access requirements for all system resources, and running all procedures necessary to ensure the correct and timely operation of the system.

system relation

A relation (table) that holds information (such as table or column names) needed for the operation of a database management system.

Systems Network Architecture (SNA)

A description of network structure, configuration and operation control, and protocols (rules) developed by IBM.

table

A set of data elements that has a horizontal dimension (row) and a vertical dimension (column) in a relational database system. A table has a specified number of columns but can have any number of rows. Rows stored in a table are structurally equivalent to records from sequential files in that they must not contain repeating columns. Another term for table is relation.

See also cardinality.

table cursor

A result table defined by the `SELECT` expression in an `SQL DECLARE CURSOR` statement in order to scan the individual rows of a table. A table cursor enables programs to perform operations on each row of a table. It can also be positioned on a row that contains a list, so that a list cursor can access the individual elements of that list. There are three types of table cursors: read-only, insert-only, and update.

See also cursor, list cursor, and result table.

table-specific constraint

A constraint that indicates that if a certain update action is taken on one table, appropriate update actions will be taken on other specified tables as well. Another term for table-specific constraint is relation-specific constraint.

See also cascading update, referential integrity, and trigger.

target database

In Data Distributor, a database into which Data Distributor copies data. A target database can be created by an extraction or replication. You can create a target database on the same node as the source database or on a node at a remote site. The target database is referred to as either an extraction database or replication database, depending on the process that created it.

See also Data Distributor, extraction, replication, source database, source node, and target node.

target node

In Data Distributor, the receiving node for data transferred from the source database on the source node. A copy of a database can be transferred to a directory on the source node or to a directory on a remote node.

See also Data Distributor, extraction, replication, source database, source node, and target database.

TDMS

A Digital product that uses forms to collect and display information on the terminal. TDMS stands for Terminal Data Management System. TDMS provides data independence by allowing data used in an application to be separated from the application program. ACMS multiple-step tasks use TDMS services to manage terminal input and output.

Terminal Data Management System (TDMS)

See TDMS.

threshold

A value that indicates the fullness of a data page. Rdb/VMS checks the space area management (SPAM) pages to determine the fullness thresholds of data pages.

See also interval and SPAM page.

TID

See distributed transaction identifier (TID).

TP

See transaction processing.

transaction

A group of statements whose changes can be made permanent or undone only as a unit. A transaction ends with a COMMIT or ROLLBACK statement. If it ends with a COMMIT statement, all the changes made to the database by the statements are made permanent. If the transaction ends with ROLLBACK, none of the statements takes effect.

See also commit, quiet point, recovery, and roll back.

transaction handle

A host language variable that specifies the name of a transaction so that it can be uniquely identified.

transaction manager

A component of a distributed transaction. The transaction manager coordinates the actions of the resource managers that are located on the same node as the transaction manager. DECdtm services is a transaction manager.

See also coordinator, DECdtm services, distributed transaction, resource manager, and two-phase commit protocol.

transaction processing

An environment that addresses large, corporate-level applications that support many users for critical business functions. In an OLTP application, there are usually many users simultaneously performing predefined (query and update) functions to a collection of shared data, generally a database. Results are usually expected immediately (real time). In contrast to standard transaction processing,

when an OLTP transaction is completed, all data is fully updated during the request and output stages. OLTP applications can be used for tasks such as order processing, banking, accounting, or inventory control.

See also ACMS and online transaction processing (OLTP).

transfer database

In Data Distributor, the storage location of all transfer definitions, schedule definitions, and record selection criteria. The transfer database is created by the transfer monitor.

See also Data Distributor, schedule definition, transfer definition, and transfer monitor.

transfer definition

In Data Distributor, a definition that uses record selection expressions to identify subsets of the source database that the transfer monitor can copy to one or more target databases. You create transfer definitions with SQL or RDO.

See also Data Distributor, record selection expression (RSE), source database, target database, and transfer monitor.

transfer monitor

The program that oversees all of the functions provided by Data Distributor. In general, the transfer monitor regulates the execution of data transfers from all source databases to target databases (directories) that are located either on the same node as the source database or on remote nodes.

See also Data Distributor, extraction, replication, and transfer definition.

trigger

A statement that causes certain updating actions to be performed before or after an update of a table in order to maintain the referential integrity of the database. For example, a trigger would specify that when an employee's record is deleted from the EMPLOYEES table, that same employee's record will be automatically deleted from other specified tables within the database.

See also cascading update, table-specific constraint, and referential integrity.

truth table

A method of showing how predicates combined with the Boolean operators AND, OR, and NOT, are evaluated.

tuple

Relational database terminology for a record or row. In SQL, the term row is used. In RDO and RDML, the term record is used.

two-phase commit protocol

A protocol provided by DECdtm services and the VMS utility. The two-phase commit protocol coordinates the activity of components of a distributed transaction to ensure that every required operation is completed before the distributed transaction is made permanent.

See also commit phase, coordinator, DECdtm services, distributed transaction, prepare phase, resource manager, and transaction manager.

UIC

See user identification code (UIC).

uniform page format

The default storage area page format. A format that consists of groups of n pages, called clumps, where n equals the buffer size divided by the page size. A set of clumps forms a logical area.

See also clump, logical area, and mixed page format.

UNION operation

The merger of the values of columns in one table with the values of columns in other tables.

unit of recovery

Another term for transaction.

usage mode

See reserving option.

user identification code (UIC)

A code that uniquely identifies a VMS user by a group number or name and a member number or name. The UIC is enclosed in brackets ([]). A UIC can be in numeric or alphanumeric format.

user name

A designation assigned to a VMS user to identify that user. Also, the name a terminal user types to log in to VMS and ACMS, or for VIDA, to log in to an IBM system.

validation

The process of checking data on entry to ensure that it meets previously established requirements.

value expression

A symbol or string of symbols used to calculate a value. If you use a value expression in a statement, SQL or RDO calculates the value associated with the expression and uses that value when executing the statement. Types of value expressions include literals, arithmetic expressions, functions, and column select expressions.

variable

A symbol declared in a program that stands for a memory storage location. A variable has a name and a data type.

VAXcluster

A highly integrated organization of VMS systems that communicate over a high-speed communications path. VAXcluster systems have all the functions of single-node VMS systems, plus the ability to share CPU resources, queues, and disk storage. Like a single-node system, the VAXcluster organization provides a single security and management environment. Member nodes can share the same operating environment or serve specialized needs.

vector

A one-dimensional array in a host language program. SQL can refer to vectors only as indicator structures.

VIDA

A database and internetworking facility that gives VMS users read-only access to data managed by an IBM database system.

view

A logical table whose data is not physically stored. Views refer to rows stored in other tables. You define a view to:

- Access a subset of the columns stored in a row
- Access a set of columns stored in different rows
- Avoid creating a redundant copy of data that is already stored

VMS

Virtual Memory System. The major operating system for Digital VAX processors.

VMS user

A person or account authorized by a VMS system manager to access a VMS system. The user is assigned a user name, a password, a user identification code (UIC), a default VMS directory, a default command language, quotas, limits, and privileges.

wildcard character

A symbol, such as the asterisk (*) or percent sign (%), that you use in place of all or part of a file specification.

working phase

The period beginning when the communication server receives its first message request from the client system and ending when the SQL/Services system fails or is manually shut down. During the working phase, the communication server accepts incoming client message requests, dispatches them to execution server processes for processing, receives in return execution results, and sends results back to the waiting client application.

See also client, communication server, execution server, initializing phase, and SQL/Services.

A

Rdb/VMS Master Index

The Rdb/VMS master index uses the abbreviations in Table A-1 for the manuals in the Rdb/VMS documentation set.

Table A-1 Abbreviations for the Manuals Included in the Master Index

Abbreviation	Manual Title
INTRO	VAX Rdb/VMS Introduction and Master Index
GUSQL	VAX Rdb/VMS Guide to Using SQL
DESIGN	VAX Rdb/VMS Guide to Database Design and Definition
MAINT	VAX Rdb/VMS Guide to Database Maintenance and Performance
GURRR	VAX Rdb/VMS Guide to Using RDO, RDBPRE, and RDML
SQLSRV	VAX Rdb/VMS Guide to Using SQL/Services
DIST_TRANS	VAX Rdb/VMS Guide to Distributed Transactions
TUNING	VAX Rdb/VMS Guide to Database Tuning
SQLRM	VAX Rdb/VMS SQL Reference Manual
RDORM	VAX Rdb/VMS RDO and RMU Reference Manual
RDMLRM	RDML Reference Manual

Index

A

Abort

- network link
 - sqlsrv_abort routine, *SQLSRV*, 9-6

Access

- costs, *MAINT*, 17-51
- displaying with *RDMS\$DEBUG_FLAGS*, *MAINT*, 17-20
- improving
 - further normalization, *MAINT*, 16-77
- saving output with *RDMS\$DEBUG_FLAGS_OUTPUT*, *MAINT*, 17-20
- strategies, *MAINT*, 17-51
- to table rows
 - using the query optimizer, *MAINT*, 17-2
 - using the optimizer, *MAINT*, 16-47
- Access conflicts, *GURRR*, 2-17t
- Access control entry (ACE), *DESIGN*, 6-2, 6-3; *RDORM*, 9-47, 9-141
- Access control list (ACL), *RDORM*, 9-43, 9-132; *SQLRM*, 6-359, 6-455
 - creating, *DESIGN*, 6-8, 6-11
 - deleting entries, *DESIGN*, 6-11
 - modifying, *DESIGN*, 6-11
 - organizing, *DESIGN*, 6-17

Access control list (ACL) (Cont.)

- See also* Privilege; Protection, *DESIGN*, 6-1

Accessing a database

- user identification code (UIC), *RDORM*, 9-44

Accessing DCL

- DCL invoke statement (\$), *RDORM*, 9-81

Access privilege set, *DESIGN*, 6-2;

SQLRM, 6-359, 6-379

- See also* Privilege; Protection
- creating, *DESIGN*, 6-11

Access rights

- See also* Privilege; Protection
- CHANGE PROTECTION* statement, *RDORM*, 9-45
- displaying, *RDORM*, 9-392
- displaying all ACL entries
 - SHOW PROTECTION* statement, *RDORM*, 9-395
- for distributed transaction, *DIST_TRANS*, 3-6

ACL

- See* Access control list

ACL-style privilege, *DESIGN*, 6-2;

SQLRM, 6-359, 6-455

Ada

- See also* Embedded SQL; Program

Ada (Cont.)

- calling procedure in SQL module, *GUSQL*, 7-11
 - declaring parameter in, *GUSQL*, 10-26
 - precompiled program, *GUSQL*, 8-4
 - supported variable declarations in precompiled SQL, *SQLRM*, 5-11
 - using parameter in, *GUSQL*, 10-26
 - ADD CONSTRAINT clause, *DESIGN*, 5-7
 - Adjusting storage and memory use parameters, *DESIGN*, 5-27t
 - After-image journal (AIJ) file
 - backing up, *MAINT*, 9-12; *RDORM*, 6-37
 - from a batch job, *MAINT*, 9-12
 - contents, *MAINT*, 9-14
 - displaying output, *RDORM*, 6-59
 - displaying unresolved transactions, *RDORM*, 6-60
 - After-image journaling, *MAINT*, 9-3, 16-79
 - AIJ file lost due to disk problem, *MAINT*, 9-17
 - displaying output, *RDORM*, 6-59
 - for database recovery, *RDORM*, 6-91
 - modifying during restore, *MAINT*, 8-54
 - RDO CHANGE DATABASE
 - statement, *MAINT*, 9-4
 - recovery, *MAINT*, 9-18
 - re-creating lost AIJ file, *MAINT*, 9-17
 - SQL ALTER SCHEMA statement, *MAINT*, 9-4
 - statistics, *MAINT*, 15-25
 - transaction sequence number, *MAINT*, 9-16
- AIJ file
- See After-image journal (AIJ) file; After-image journaling

Alarms

- interpreting AUDIT type, *MAINT*, 4-22
- interpreting DACCESS type, *MAINT*, 4-22
- interpreting PROTECTION type, *MAINT*, 4-24
- interpreting RMU type, *MAINT*, 4-24
 - security, *MAINT*, 4-3, 4-20; *RDORM*, 6-127
- Alias, *GUSQL*, 3-10; *SQLRM*, 3-15
 - naming, *SQLRM*, 3-15
 - outer references, *SQLRM*, 3-17
- ALL keyword, *GUSQL*, 3-54
- ALLOCATION clause
 - importance for hashed index, *DESIGN*, 4-12
- ALTER DOMAIN statement, *DESIGN*, 5-6; *SQLRM*, 6-2
- ALTER INDEX statement, *MAINT*, 16-50; *SQLRM*, 6-11
- Altering
 - database corruption, *MAINT*, 7-1
 - See also Changing; Modifying, *DESIGN*, 5-20
 - triggers, *DESIGN*, 5-20
- ALTER SCHEMA statement, *MAINT*, 10-7; *SQLRM*, 6-15
 - journaling, *MAINT*, 9-4, 10-18
- ALTER STORAGE MAP statement, *SQLRM*, 6-29
- ALTER TABLE statement, *DESIGN*, 5-7; *SQLRM*, 6-35
- ANALYZE statement
 - See also RMU/ANALYZE command
 - gathering statistics, *RDORM*, 9-2
- Analyzing databases, *RDORM*, 6-5
- Anchor
 - of data dictionary, *DESIGN*, 7-2
- AND expression, *RDMLRM*, 3-3
- AND operator, *GUSQL*, 3-28, 3-47

ANSI/ISO standard
 flagging extension, *GUSQL*, 7-27, 8-12
 format with COBOL program, *GUSQL*, 8-5

ANSI/ISO-style privilege, *DESIGN*, 6-3; *SQLRM*, 6-379, 6-465

ANY expression, *RDMLRM*, 3-9

ANY operator, *GUSQL*, 3-34; *GURRR*, 3-14

API (Application Programming Interface), *SQLSRV*, 1-2
 Macintosh, *SQLSRV*, 1-4
 MS-DOS, *SQLSRV*, 1-4
 OS/2, *SQLSRV*, 1-4
 ULTRIX, *SQLSRV*, 1-4
 ULTRIX for RISC, *SQLSRV*, 1-4
 VMS, *SQLSRV*, 1-4

API routines, *SQLSRV*, 9-1, 9-6 to 9-75
 association, *SQLSRV*, 2-8 to 2-9
 based on SQL statements, *SQLSRV*, 2-9 to 2-10
 C format, *SQLSRV*, 9-3
 data structures used, *SQLSRV*, 2-12 to 2-13
 documentation format, *SQLSRV*, 9-1
 embedding in C programs, *SQLSRV*, 1-7
 functional interface, *SQLSRV*, 2-11 to 2-12
 naming, *SQLSRV*, 9-2
 overview, *SQLSRV*, 2-8 to 2-12
 result table, *SQLSRV*, 2-10
 return values, *SQLSRV*, 9-2t
 utility, *SQLSRV*, 2-10 to 2-11
 VAX format, *SQLSRV*, 9-3

Application
 running as detached process, *GURRR*, 9-2

Archiving information, *DESIGN*, 2-13

Area
See Storage area

Area bit map
 uniform page format, *MAINT*, 11-63e

Area inventory page
 uniform page format, *MAINT*, 11-64e

AREA . . . PAGE command
 (RdbALTER), *RDORM*, 7-3

Area qualifiers
See Parameter qualifiers

Arithmetic expression, *RDMLRM*, 2-4; *RDORM*, 3-21

Arithmetic operator, *GUSQL*, 3-5; *SQLRM*, 3-75

Ascending index, *RDORM*, 9-122

Ascending value
 ordering row in, *GUSQL*, 3-19

ASC keyword
 in ORDER BY clause, *GUSQL*, 3-19

ASSOCIATE_STR data structure, *SQLSRV*, 2-12, 10-3

Association
 creating and releasing in SQL/Services, *SQLSRV*, 4-6

Asterisk (*)
 wildcard character, *GURRR*, 3-7

ASTLM parameter
 values, *MAINT*, 16-140

AT END clause
 error handling, *RDORM*, 9-276

At sign (@) command
See Execute statement (@)

ATTACH command (RdbALTER), *MAINT*, 7-3; *RDORM*, 7-5

Attaching to a database, *MAINT*, 5-5; *GUSQL*, 2-3; *SQLRM*, 6-255
 failure, *GUSQL*, 11-38
 in RdbALTER, *MAINT*, 7-3
 INVOKE DATABASE statement, *RDORM*, 9-329

Attribute
 global, *RDORM*, 5-1
 local, *RDORM*, 5-2

Audit event
See also Security; Security auditing

Audit event (Cont.)

AUDIT event type

enabling or disabling, *MAINT*,
4-4

default security auditing, *MAINT*,
4-2

disabling DACCESS, *MAINT*, 4-5,
4-13; *RDORM*, 6-128

disabling event information, *MAINT*,
4-16

enabling DACCESS, *MAINT*, 4-5,
4-13; *RDORM*, 6-128

enabling event information, *MAINT*,
4-16

Auditing

See also Security; Security auditing
audit event information

enabling or disabling, *MAINT*,
4-16

DACCESS level security auditing,
MAINT, 4-11, 4-13

disabling, *RDORM*, 6-128

displaying characteristics, *RDORM*,
6-143

enabling, *RDORM*, 6-127, 6-128

event level security auditing,
MAINT, 4-11, 4-16

four levels of security auditing,
MAINT, 4-11

monitoring resources, *MAINT*, 4-3
specific objects, *MAINT*, 4-13;

RDORM, 6-130

specific object types, *MAINT*, 4-13;
RDORM, 6-129

starting, *MAINT*, 4-16; *RDORM*,
6-133

stopping, *MAINT*, 4-11, 4-16;
RDORM, 6-133

top level security auditing, *MAINT*,
4-11, 4-16

use of RMU commands, *MAINT*,
4-10; *RDORM*, 6-132

user level security auditing, *MAINT*,
4-11, 4-12

Audit journal, *MAINT*, 4-3, 4-25;
RDORM, 6-127

reviewing, *MAINT*, 4-28

Audit journal records

defining database table for storing,
MAINT, 4-25; *RDORM*, 6-71

defining relation for storing, *MAINT*,
4-25; *RDORM*, 6-71

Audit trail

in RdbALTER, *MAINT*, 7-24

Authorization identifier, *DESIGN*,
3-12; *GUSQL*, 2-9; *SQLRM*, 3-10

in SQL module, *GUSQL*, 9-4

AUTHORIZATION keyword

in SQL module, *GUSQL*, 7-12

AUTOGEN parameters, *MAINT*,
16-132

Auto-locking, *GURRR*, 2-13

AVERAGE function, *RDMLRM*, 5-4;
RDORM, 3-3, 3-11

AVERAGE statistical expression,
RDORM, 3-14

AVG function, *GUSQL*, 3-54; *SQLRM*,
3-70

B

Backing up a database, *MAINT*, 8-1,
8-23e, 9-35e

EXPORT statement, *RDORM*, 9-271;
SQLRM, 6-349

full, *MAINT*, 8-13

incremental, *MAINT*, 8-18

online, *MAINT*, 8-10

options, *MAINT*, 8-22

proper timing for changing AIJ file,
MAINT, 9-20

read-only storage areas, *MAINT*,
8-25e

RMU/BACKUP command, *RDORM*,
6-20

specifying by area incremental
backup, *MAINT*, 8-32

Backing up a database (Cont.)

- specifying complete incremental backup, *MAINT*, 8-32
- specifying file protection, *MAINT*, 8-9
- tradeoffs
 - incremental backup versus multiple AIJ files, *MAINT*, 9-8
- Backup file
 - displaying output, *RDORM*, 6-62
- BACKUP statement
 - See EXPORT statement
 - obsolete, *RDORM*, F-1
- BASED ON clause, *RDMLRM*, 6-4
- BASIC
 - data type equivalents, *GURRR*, 8-7t
 - declaring host language variables, *GURRR*, 9-2
 - designing program in RDO, *GURRR*, 7-3
 - developing an Rdb/VMS program in, *GURRR*, 13-1 to 13-56
 - INCLUDE directive, *GURRR*, 12-7
- Basic predicate, *SQLRM*, 3-81
- Batch-update transaction, *GUSQL*, 2-24, 2-28; *GURRR*, 2-7, 2-9; *DIST_TRANS*, 2-1, 4-3
- Before-image journal
 - See Recovery-unit journal
- BEGIN DECLARE statement, *SQLRM*, 6-52
- BETWEEN expression, *RDMLRM*, 3-13
- BETWEEN operator, *GUSQL*, 3-35; *SQLRM*, 3-81
- Binary data
 - and C, *GUSQL*, 7-24, 8-4, 10-33
- BIOLM parameter
 - values, *MAINT*, 16-140
- Blocked database, *DIST_TRANS*, 6-4
- Boolean operator, *GUSQL*, 3-28, 3-45; *SQLRM*, 3-82

B-tree

See Sorted index

Buffers

flushing, *MAINT*, 16-92

Bugcheck dump

- causes, *MAINT*, 12-6, 12-8
- contents of, *MAINT*, 12-9
- database recovery (DBR), *MAINT*, 12-3
- (DBR) RDMDBRBUG.DMP file, *MAINT*, 12-2
- error message meaning, *MAINT*, 12-4
- examining, *MAINT*, 12-9
- example error messages, *MAINT*, 12-5
- exception
 - disk quota exceeded, *MAINT*, 12-6
 - insufficient page file quota (PGFLQUOTA), *MAINT*, 12-7
 - insufficient virtual memory (VIRTUALPAGECNT), *MAINT*, 12-7
 - invalid dbkey, *MAINT*, 12-8
 - lock queue limit (ENQLM) too low, *MAINT*, 12-7
- exception code, *MAINT*, 12-9
- locations, *MAINT*, 12-3
- (monitor)
 - SYS\$SYSTEM:RDMBUGCHK.DMP file, *MAINT*, 12-2
- Rdb/VMS Management Utility (RMU), *MAINT*, 12-3
- Rdb/VMS run-time services, *MAINT*, 12-3
- (RMU) RMUBUGCK.DMP file, *MAINT*, 12-2
- (Run-time services) RDSBUGCK.DMP file, *MAINT*, 12-2
- types of, *MAINT*, 12-2
- typical message, *MAINT*, 12-3
- when to report, *MAINT*, 12-8

By area restore
 purpose of, *RDORM*, 6–110
BY DESCRIPTOR clause
 of SQL module language, *GUSQL*,
 7–22
BYTE VARYING data type, *SQLRM*,
 3–37
BYTLM parameter
 values, *MAINT*, 16–139

C

C

See also Embedded SQL; Program
and binary data, *GUSQL*, 7–24, 8–4,
 10–33
data types, *GURRR*, 8–8t, 8–9t
data types generated by RDML,
 RDMLRM, A–1
declaring
 host language variables, *GURRR*,
 9–2
 parameter in, *GUSQL*, 10–33
 typedef, *RDMLRM*, 6–4
designing program in RDO, *GURRR*,
 7–3
functions for I/O operations,
 RDMLRM, B–1
null-terminated string, *GUSQL*,
 7–24, 8–4, 10–33
placement in program using
 DATABASE statement,
 RDMLRM, 6–16
precompiled program, *GUSQL*, 8–4
programming for SQL/Services,
 SQLSRV, 1–7
supported variable declarations in
 precompiled SQL, *SQLRM*, 5–18
using RDML statements in, *GURRR*,
 17–1 to 17–53
Callable RDO, *GURRR*, 7–1, 19–2
 data definition statements, *GURRR*,
 19–29

Callable RDO (Cont.)

 data manipulation statements,
 GURRR, 19–6
 designing program in, *GURRR*, 7–3
 error handling, *GURRR*, 19–34
 linking programs, *GURRR*, 11–12
 processing programs, *GURRR*, 11–12
 using with distributed transaction,
 DIST_TRANS, 5–1
Callable RDO statement
 in BASIC programs, *GURRR*, 13–37
 in COBOL programs, *GURRR*, 14–38
 in C programs, *GURRR*, 17–36
 in FORTRAN programs, *GURRR*,
 15–40
 in Pascal programs, *GURRR*, 18–35
Cascading delete, *DESIGN*, 3–38
Case sensitivity
 and conditional operator, *GUSQL*,
 3–23
CDD
 See Data dictionary
CDD\$COMPATIBILITY logical name,
 DESIGN, 7–6, 7–7; *MAINT*, 18–19
CDD/Plus
 See Data dictionary
CDO (Common Dictionary Operator),
 DESIGN, 7–2
CHANGE DATABASE statement
 after-image journaling process,
 RDORM, 9–16
 database shutdown process, *RDORM*,
 9–23
 examples, *RDORM*, 9–23
 journaling, *MAINT*, 9–4
 modifying database parameters,
 RDORM, 9–9
CHANGE FIELD statement
 changing field definitions, *RDORM*,
 9–27
CHANGE INDEX statement
 changing index definitions, *RDORM*,
 9–36

CHANGE PROTECTION statement
 modifying access rights, *RDORM*,
 9-45

CHANGE RELATION statement
 modifying fields, *RDORM*, 9-48

CHANGE STORAGE MAP statement
 changing storage map definitions,
RDORM, 9-61

Changing
See also Modifying

- access control list, *DESIGN*, 6-11
- column definitions, *DESIGN*, 5-7
- columns, *SQLRM*, 6-35
- constraints, *DESIGN*, 5-17;
RDORM, 9-48; *SQLRM*, 6-35
- data, *RDORM*, 9-338; *GURRR*, 6-4
- database characteristics, *MAINT*,
 10-6 to 10-28
- database definitions
 with INTEGRATE statement,
DESIGN, 7-15
- database files, *DESIGN*, 5-25
- database page contents, *MAINT*,
 7-13
- databases, *DESIGN*, 5-27t;
RDORM, 9-9; *SQLRM*, 6-15
- dictionary definitions
 by replacing definitions,
DESIGN, 7-14
- domains, *DESIGN*, 5-6; *SQLRM*,
 6-2
- fields, *RDORM*, 9-27, 9-48
- indexes, *RDORM*, 9-36; *SQLRM*,
 6-11
- lists, *GUSQL*, 5-12
- memory usage for data access,
DESIGN, 5-25
- metadata, *DESIGN*, 7-13
- radix in RdbALTER, *MAINT*, 7-13,
 7-23
- RDO parameters
 SET statement, *RDORM*, 9-362
- referential integrity, *DESIGN*, 5-20
- relations, *RDORM*, 9-48

Changing (Cont.)

- schemas, *SQLRM*, 6-15 to 6-28
- segmented strings, *GURRR*, 9-28
- SQL parameters, *SQLRM*, 6-482
- storage area and schema
 characteristics by restoring,
DESIGN, 5-26
- storage area characteristics with
 RMU/COPY_DATABASE or
 RMU/MOVE_AREA, *DESIGN*,
 5-26
- storage map definitions, *RDORM*,
 9-61
- CHANGE STORAGE MAP
 statement, *RDORM*, 9-61
- storage maps, *RDORM*, 9-61;
SQLRM, 6-29 to 6-34
- storage parameters, *DESIGN*, 5-25
- tables, *DESIGN*, 5-7; *SQLRM*, 6-35
 to 6-51
- triggers, *DESIGN*, 5-20

CHANNELCNT parameter
 values, *MAINT*, 16-133

Character data type, *SQLRM*, 3-34

Characteristics
 security auditing, *RDORM*, 6-143

Character string literal, *SQLRM*, 3-43

CHAR data type, *DESIGN*, 3-18;
SQLRM, 3-34

Checking
 tape labels
 with RMU, *RDORM*, 6-34, 6-66,
 6-120

CHECK option
 for procedure parameter, *GUSQL*,
 7-23

Client/server model
 SQL/Services, *SQLSRV*, 1-1f

CLISYMTBL parameter
 values, *MAINT*, 16-133

CLOSE statement, *GUSQL*, 4-2;
SQLRM, 6-55

See RMU/CLOSE command
 obsolete, *RDORM*, F-3

CLOSE statement (Cont.)

using parameter, *GUSQL*, 12-46

Closing a database, *MAINT*, 5-14;
RDORM, 6-42

Closing cursors, *SQLRM*, 6-55
in distributed transaction, *DIST_*
TRANS, 4-9

Clustering rows

by index values
to improve joins, *DESIGN*, 4-16

Clustering strategies for data
using hashed indexes, *MAINT*, 2-15

COBOL

See also Embedded SQL; Program
COPY FROM DICTIONARY
statement, *GURRR*, 12-8

declaring host language variables,
GURRR, 9-2

declaring parameter in, *GUSQL*,
10-43

designing program in RDO, *GURRR*,
7-3

developing an Rdb/VMS program in,
GURRR, 14-1 to 14-56

precompiled program, *GUSQL*, 8-5
supported variable declarations in
precompiled SQL, *SQLRM*, 5-24

using parameter in, *GUSQL*, 10-43

Collating sequence, *GURRR*, 1-14, 3-9
clause, *RDORM*, 5-19
creating, *SQLRM*, 6-64 to 6-67
defining

DEFINE COLLATING_
SEQUENCE statement,
RDORM, 9-85

deleting, *RDORM*, 9-225; *SQLRM*,
6-295

modifying, *RDORM*, 9-31; *SQLRM*,
6-5

specifying, *DESIGN*, 3-21; *GUSQL*,
1-28

using to sort rows, *DESIGN*, 3-21

Column

changing, *SQLRM*, 6-35

changing COMPUTED BY clause,
DESIGN, 5-12

changing definition of, *DESIGN*, 5-7
constraint, *DESIGN*, 3-31

creating, *DESIGN*, 5-7

creating, *SQLRM*, 6-138

default values, *DESIGN*, 3-30
defining protection for, *DESIGN*,
6-24

definition of, *INTRO*, 1-2

deleting, *DESIGN*, 5-7; *SQLRM*,
6-316

ensuring unique values in, *DESIGN*,
3-42

modifying, *SQLRM*, 6-35

naming, *SQLRM*, 3-13

using the COMPUTED BY clause,
DESIGN, 3-29

Column constraint

See Constraint

Column select expression, *GUSQL*, 3-4;
SQLRM, 3-98, 3-106

as value expression, *GUSQL*, 3-67
performance compared to join,
GUSQL, 3-62

Combining

queries, *GUSQL*, 3-59

tables, *GUSQL*, 3-59

Command line qualifier

for SQL module processor, *GUSQL*,
7-26

for SQL precompiler, *GUSQL*, 8-12

Command procedure, *GUSQL*, 1-21
SQL, *GUSQL*, 6-6

executing, *GUSQL*, 1-21

using to define protection, *DESIGN*,
6-19

Comment

including in

SQL command procedure,
GUSQL, 1-21

SQL module, *GUSQL*, 7-8

COMMENT ON statement, *SQLRM*, 6-57

COMMIT
in RdbALTER, *MAINT*, 7-25

COMMIT command (RdbALTER), *MAINT*, 7-16; *RDORM*, 7-7

Commit phase
of distributed transaction, *DIST_TRANS*, 1-6, 2-12

COMMIT statement, *GUSQL*, 2-46; *RDMLRM*, 6-7; *GURRR*, 6-1; *SQLRM*, 6-60
writing changes to a database, *RDORM*, 9-69

Committing
changes
in RdbALTER, *MAINT*, 7-16
transactions
in RdbALTER, *MAINT*, 7-25

Committing transactions, *SQLRM*, 6-60
distributed, *DIST_TRANS*, 2-6

Common Dictionary Operator (CDO)
See CDO

Compatibility
See Locking

Compiling
host language
input file, *GUSQL*, 8-11t
output file, *GUSQL*, 8-11t
invoking the SQL module processor, *GUSQL*, 7-30
SQL module language source file, *GUSQL*, 7-25

Completing transactions
in RdbALTER, *MAINT*, 7-24

Complex predicate, *SQLRM*, 3-82

Compressed index, *DESIGN*, 3-46; *SQLRM*, 6-79

COMPUTED BY clause, *DESIGN*, 3-29

Concatenated expressions, *RDORM*, 3-25; *SQLRM*, 3-74

Concatenate operator (||) (RDO), *GURRR*, 3-7

Concatenate operator (||) (SQL), *SQLRM*, 3-74

Concatenation operator, *GUSQL*, 3-5, 3-7

Concealed logical name, *DESIGN*, 3-10

Concurrent transactions
lock conflicts, *MAINT*, 16-27

Conditional expression, *RDORM*, 3-36; *GURRR*, 3-7, 3-8
See also Predicate
compound, *GURRR*, 3-10
defining, *RDORM*, 3-1

Conditional operator, *GUSQL*, 3-23, 3-28
See also Conditional expression; Predicate; and Relational operator
range retrieval, *GUSQL*, 3-35

Condition handler, *GURRR*, 10-1

Consistency
locking, *MAINT*, 16-17

Constant
See Literal

Constraint, *GURRR*, 2-19; *SQLRM*, 3-30
ADD CONSTRAINT clause, *DESIGN*, 5-7
and referential integrity, *DESIGN*, 3-37; *GURRR*, 6-16
changing, *DESIGN*, 5-17; *RDORM*, 9-48; *SQLRM*, 6-35
column constraint, *SQLRM*, 6-142
creating, *DESIGN*, 3-31; *SQLRM*, 6-138
default mode, *SQLRM*, 6-485
DEFINE CONSTRAINT statement, *RDORM*, 9-88
defining, *RDORM*, 9-88
definition of, *INTRO*, 1-4
DELETE CONSTRAINT statement, *RDORM*, 9-228
deleting, *DESIGN*, 5-7, 5-17; *RDORM*, 9-228; *SQLRM*, 6-298
displaying information about, *GUSQL*, 1-9; *RDORM*, 9-377

Constraint (Cont.)

- evaluating, *MAINT*, 16–81; *GUSQL*, 5–1, 11–28, 11–29, 11–30
- modifying, *DESIGN*, 5–17; *RDORM*, 9–48; *SQLRM*, 6–35
- naming, *SQLRM*, 3–30
- UNIQUE, *DESIGN*, 3–42
- using, *GURRR*, 6–16
- validation
 - error, *GUSQL*, 11–28
 - violation, *DESIGN*, 5–19; *GUSQL*, 11–28
 - violation of, *GURRR*, 10–12
- CONTAINING expression, *RDMLRM*, 3–16
 - with multinational characters, *RDMLRM*, 3–17
- CONTAINING operator, *GUSQL*, 3–24; *SQLRM*, 3–84
- Contents of
 - after-image journal file, *MAINT*, 9–14
 - area bit map, *MAINT*, 11–63
 - area inventory page, *MAINT*, 11–64
 - data page with hashed index, *MAINT*, 11–42e
 - display and interpret
 - file contents, *MAINT*, 11–1
 - hashed index node record, *MAINT*, 11–44
 - index node record, *MAINT*, 11–39
 - line index, *MAINT*, 11–30
 - locked and unlocked free space, *MAINT*, 11–31
 - page header, *MAINT*, 11–29
 - page tail
 - mixed storage area, *MAINT*, 11–50
 - uniform storage area, *MAINT*, 11–51
 - recovery-unit journal file, *MAINT*, 9–32
 - segmented string, *MAINT*, 11–36
 - snapshot file, *MAINT*, 11–22

Contents of (Cont.)

- space area management page, *MAINT*, 11–61, 11–66
- storage segment, *MAINT*, 11–34
- transaction sequence number, *MAINT*, 11–31
- user-stored records, *MAINT*, 11–35
- Context file, *GUSQL*, 8–10
- Context structure
 - and module language, *DIST_TRANS*, 4–2, 4–10
 - and precompiled SQL, *DIST_TRANS*, 4–3, 4–16
 - format of, *DIST_TRANS*, 4–10
 - using with distributed transaction, *DIST_TRANS*, 4–8
- Context variable, *RDMLRM*, 4–8; *RDORM*, 4–6; *GURRR*, 3–2
- Conversion of data types, *SQLRM*, 3–39
 - by RMU/LOAD operations, *RDORM*, 6–72
 - in precompiled programs, *SQLRM*, 5–10
- Converting databases, *RDORM*, 6–46
 - RMU/RESTORE command, *RDORM*, 6–107
- CONVERT statement
 - See also* RMU/CONVERT command
 - obsolete, *RDORM*, F–5
- Coordinator
 - in distributed transaction, *DIST_TRANS*, 1–4, 2–3
- Copying
 - databases, *RDORM*, 6–50
 - dictionary definitions
 - with INTEGRATE DATABASE, *RDORM*, 9–325
- Copying database, *DESIGN*, 5–26
- Copying declaration
 - See also* FROM dictionary clause
- Corruption, *MAINT*, 6–26e
 - See also* Database corruption
 - causes of database corruption, *MAINT*, 6–3

Corruption (Cont.)

- checking for database corruption,
MAINT, 6-3
- clearing a corruption flag, *MAINT*,
7-3
- clearing an inconsistent flag, *MAINT*,
7-22
- data integrity, *MAINT*, 6-36e
- in a batch-update transaction,
MAINT, 6-3
- page header, *MAINT*, 6-27e
- patching the database, *MAINT*, 7-1
- sorted index, *MAINT*, 6-32e

Costs

- RDMS\$DEBUG_FLAGS*, *MAINT*,
17-51
- COUNT function, *GUSQL*, 3-54;
RDMLRM, 5-8; *RDORM*, 3-3;
SQLRM, 3-69
- COUNT statistical expression, *RDORM*,
3-14
- CPU resource problems, *TUNING*, 2-48
- CREATE COLLATING SEQUENCE
statement, *SQLRM*, 6-64
- CREATE DOMAIN statement, *SQLRM*,
6-68
- CREATE INDEX statement, *DESIGN*,
3-40; *SQLRM*, 6-76
- CREATE SCHEDULE statement,
SQLRM, 6-88
- CREATE SCHEMA statement,
DESIGN, 1-6, 3-4; *SQLRM*, 6-97
- CREATE STORAGE AREA clause,
DESIGN, 3-7, 4-3; *SQLRM*, 6-121
- CREATE STORAGE MAP statement,
DESIGN, 3-10; *SQLRM*, 6-130
- INTERVAL clause, *DESIGN*, 4-19
- partitioning table across storage
areas, *DESIGN*, 4-9
- THRESHOLD clause, *DESIGN*, 4-19
- CREATE TABLE statement, *DESIGN*,
3-24, 3-26; *SQLRM*, 6-138
- using data dictionary, *DESIGN*, 3-24

- CREATE TRANSFER statement,
SQLRM, 6-168
- CREATE TRIGGER statement,
SQLRM, 6-202
- CREATE VIEW statement, *DESIGN*,
3-46; *SQLRM*, 6-219
- CREATE_SEGMENTED_STRING
statement, *RDORM*, 9-74
- Creating
 - See also* Defining
 - access control lists, *DESIGN*, 6-11
 - access privilege sets, *DESIGN*, 6-2,
6-11
 - collating sequence, *SQLRM*, 6-64 to
6-67
 - column constraint, *DESIGN*, 5-7
 - columns, *SQLRM*, 6-35, 6-138
 - constraints, *DESIGN*, 3-31, 5-7;
SQLRM, 6-138
 - databases, *DESIGN*, 3-1; *MAINT*,
11-65; *SQLRM*, 6-97
 - using space management,
MAINT, 11-65
 - using SQL, *DESIGN*, 3-3
 - dictionary definitions, *DESIGN*, 7-15
 - using SQL, *DESIGN*, 7-20
 - domains, *DESIGN*, 3-14; *SQLRM*,
6-68 to 6-75
 - indexes, *DESIGN*, 3-40; *SQLRM*,
6-76 to 6-87
 - new storage area, *MAINT*, 10-28
 - schemas, *DESIGN*, 1-6, 3-4;
SQLRM, 6-97 to 6-120
 - using CDO definitions, *DESIGN*,
7-11
 - snapshot storage areas, *DESIGN*,
3-54
 - storage areas, *DESIGN*, 3-7;
SQLRM, 6-121 to 6-129
 - storage maps, *SQLRM*, 6-130 to
6-137
 - tables, *DESIGN*, 3-23; *SQLRM*,
6-138 to 6-167

Creating

tables (Cont.)

- with data dictionary, *DESIGN*, 3-23, 3-24
- without data dictionary, *DESIGN*, 3-26
- triggers, *DESIGN*, 3-39; *SQLRM*, 6-202 to 6-218
- views, *DESIGN*, 3-46; *SQLRM*, 6-219 to 6-230
- CROSS clause, *RDMLRM*, 4-13; *GURRR*, 4-1, 4-2
 - of record selection expression, *RDORM*, 4-12
- Crossing relations, *GURRR*, 4-2
- CURRENT_TIMESTAMP keyword, *SQLRM*, 3-47
- Cursor, *GUSQL*, 4-1 to 4-21; *SQLRM*, 6-233 to 6-244
 - categories of, *GUSQL*, 4-4
 - closing, *GUSQL*, 4-2; *SQLSRV*, 9-13; *SQLRM*, 6-55
 - in distributed transaction, *DIST_TRANS*, 4-9
 - declaring, *GUSQL*, 4-1; *SQLRM*, 6-233
 - declaring dynamic, *SQLSRV*, 9-15
 - dynamic, *GUSQL*, 4-5, 4-18, 12-19
 - extended dynamic, *GUSQL*, 4-5, 4-19, 12-47
 - insert-only, *SQLRM*, 6-234
 - list, *GUSQL*, 4-5, 4-11, 5-11; *SQLRM*, 6-234
 - naming, *SQLRM*, 3-29
 - opening, *GUSQL*, 4-1; *SQLSRV*, 9-34; *SQLRM*, 6-435
 - read-only, *SQLRM*, 6-234
 - retrieving, *GUSQL*, 4-1; *SQLRM*, 6-352
 - scope and extent, *SQLRM*, 6-233
 - table, *GUSQL*, 4-5; *SQLRM*, 6-234
 - update, *SQLRM*, 6-234
 - updating row, *GUSQL*, 5-13

D

DACCESS audit events

- See also* Security; Security auditing
- column privileges audited, *MAINT*, 4-9
- database privileges audited, *MAINT*, 4-5
- enabling or disabling, *MAINT*, 4-5, 4-13; *RDORM*, 6-128
- table privileges audited, *MAINT*, 4-7

Data

- accessing, *GURRR*, 2-3
- archiving, *DESIGN*, 2-13
- binary
 - storing in database, *SQLRM*, 3-37
- changing, *RDORM*, 9-338; *GURRR*, 6-4
- clustering strategies
 - using hashed indexes, *MAINT*, 2-15
- database pages
 - header, *MAINT*, 11-28
- definition statements, *RDORM*, 2-2; *SQLRM*, 2-2
- deleting, *GURRR*, 6-7; *SQLRM*, 6-283
- estimating size of data row, *MAINT*, 16-64
- loading operations
 - changing database definitions, *MAINT*, 2-45
 - troubleshooting, *MAINT*, 2-19
 - with RMU/LOAD command, *MAINT*, 2-37
- loading strategies, *MAINT*, 2-2
- manipulation statements, *RDORM*, 2-5; *SQLRM*, 2-5
- modifying, *RDORM*, 9-338; *GURRR*, 6-4
- moving, *MAINT*, 7-19

Data (Cont.)

- placement strategies
 - using hashed indexes, *MAINT*, 2-15
 - redundant, *MAINT*, 16-16
 - retrieval methods, *DESIGN*, 1-10
 - retrieving, *SQLRM*, 6-474
 - storage
 - multifile, *MAINT*, 11-2t
 - multifile and single-file, *MAINT*, 11-2
 - single-file, *MAINT*, 11-2t
 - storage area
 - types of pages, *MAINT*, 11-10
 - storing, *GURRR*, 6-1; *SQLRM*, 6-414, 6-560
 - types, *RDORM*, 5-5; *SQLSRV*, 11-1; *SQLRM*, 3-32
 - conversion of by RMU/LOAD operations, *RDORM*, 6-72
 - unloading
 - record definition format, *MAINT*, 2-39
 - with RMU/UNLOAD command, *MAINT*, 2-37
- Data access methods, *DESIGN*, 1-10
- Data and indicator variables
 - freeing, *SQLSRV*, 9-30
- Database
- See also* Schema
- active records, *MAINT*, 16-78
 - adding new storage areas, *MAINT*, 10-28
 - adding records
 - STORE statement, *RDORM*, 9-457
 - adjusting parameters, *MAINT*, 16-86
 - adjusting storage area parameters, *MAINT*, 16-106, 16-107t
 - AIJ file lost, *MAINT*, 9-17
 - ALLOCATION parameter, *MAINT*, 16-110
 - analyzing, *RDORM*, 6-5

Database (Cont.)

- analyzing performance, *MAINT*, 14-1
- and schema context, *GUSQL*, 2-1
- attaching to, *MAINT*, 5-5; *GUSQL*, 2-3
- INVOKE DATABASE statement, *RDORM*, 9-329
- more than one, *GUSQL*, 2-9
- with DECLARE SCHEMA statement, *SQLRM*, 6-255
- backing up, *MAINT*, 8-8, 8-23e; *RDORM*, 6-20
 - using multiple tape drives, *MAINT*, 8-37
 - using single tape drive, *MAINT*, 8-35
- backing up after-image journals, *RDORM*, 6-37
- backup options, *MAINT*, 8-22
- BUFFER SIZE parameter, *MAINT*, 16-89
- causes of corruption, *MAINT*, 6-3
- changing, *DESIGN*, 5-25
 - See also* Database, characteristics, changing
- characteristics, *DESIGN*, 5-27t; *MAINT*, 10-6 to 10-28
- hashed indexes, *MAINT*, 10-41
- indexes, *MAINT*, 10-38
- sorted indexes, *MAINT*, 10-39
- storage areas, *MAINT*, 10-31
- storage maps, *MAINT*, 10-45
- changing page contents, *MAINT*, 7-13
- characteristics, *MAINT*, 3-13
 - changing, *MAINT*, 10-6 to 10-28
 - AIJ allocation size, *MAINT*, 10-24
 - EXTENT IS extent-pages PAGES, *MAINT*, 10-11
 - lock characteristics, *MAINT*, 10-15

Database

characteristics

changing (Cont.)

- maximum number of users,
MAINT, 10-13
- maximum number of
VAXcluster nodes,
MAINT, 10-14
- number of database buffers,
MAINT, 10-16
- number of recovery buffers,
MAINT, 10-17
- OPEN IS option, *MAINT*,
10-10
- requirement for a dictionary,
MAINT, 10-26
- snapshot allocation size,
MAINT, 10-25

controlling

- after-image journaling,
MAINT, 10-18
- snapshot allocation size,
MAINT, 10-22
- snapshots, *MAINT*, 10-18
- making snapshots deferred,
MAINT, 10-23
- making snapshots immediate,
MAINT, 10-24

checking for corruption, *MAINT*, 6-3

choosing between single-file and
multifile, *DESIGN*, 3-8, 3-9t

closing, *MAINT*, 5-7; *RDORM*, 6-42
for maintenance operations,
MAINT, 5-9

using qualifiers, *MAINT*, 5-14

converting, *MAINT*, 9-22; *RDORM*,
6-46

from V2.3 to V3.0 or V3.1 to V4.0,
MAINT, 9-24

from V2.3 to V4.0, *MAINT*, 9-22
from V3.0 or V3.1 to V4.0,

MAINT, 9-26

to VAXclusters, *MAINT*, 18-26

copying, *MAINT*, 10-73

Database

copying (Cont.)

using RMU/COPY_DATABASE,
DESIGN, 5-26

corruption, *MAINT*, 6-26e

data integrity, *MAINT*, 6-36e

page header, *MAINT*, 6-27e

sorted index, *MAINT*, 6-32e

creating, *DESIGN*, 3-4; *MAINT*,
11-65; *SQLRM*, 6-97

in VAXclusters, *MAINT*, 18-19

creating a duplicate of, *DESIGN*,
5-23; *RDORM*, 6-50

creating index, *DESIGN*, 3-40

creating table, *DESIGN*, 3-23, 3-26

creating view, *DESIGN*, 3-46

database recovery (DBR) process,
MAINT, 9-2

data clustering strategies

using hashed indexes, *MAINT*,
2-15

data loading operations

troubleshooting, *MAINT*, 2-19

data loading strategies, *MAINT*, 2-2

data placement strategies

using hashed indexes, *MAINT*,
2-15

declaring, *GURRR*, 9-3

defining, *RDORM*, 9-93

defining protection for, *DESIGN*, 6-1

definitions

ALTER SCHEMA statement,
SQLRM, 6-15

CHANGE DATABASE statement,
RDORM, 9-9

CREATE SCHEMA statement,
SQLRM, 6-97

DEFINE DATABASE statement,
RDORM, 9-93

DELETE DATABASE statement,
RDORM, 9-232

DROP SCHEMA statement,
SQLRM, 6-311

Database (Cont.)

- deleting
 - indexes, *MAINT*, 10-38, 10-44
 - storage areas, *MAINT*, 10-31
 - storage maps, *MAINT*, 10-45, 10-51
- design concept, *DESIGN*, 3-1
 - See also* Database design
 - creation parameters, *MAINT*, 11-65
 - logical, *DESIGN*, 1-4, 2-1
 - snapshot storage area, *DESIGN*, 3-54
- detaching from, *GUSQL*, 2-16;
GURRR, 9-4
FINISH statement, *RDORM*, 9-280; *SQLRM*, 6-357
- displaying and interpreting
 - file contents, *MAINT*, 11-1
- displaying information about,
GUSQL, 1-5, 1-13; *RDORM*, 6-54, 6-142, 9-372, 9-379;
SQLRM, 6-521
- displaying pages, *MAINT*, 7-11
- displaying security auditing
 - characteristics, *MAINT*, 4-2;
RDORM, 6-143
- displaying user information, *MAINT*, 3-10
- duplicating metadata, *DESIGN*, 5-23
- duplicating without data, *SQLRM*, 6-350
- examples of running verify operations,
MAINT, 6-14
- exporting, *DESIGN*, 5-26; *RDORM*, 9-271; *SQLRM*, 6-349
- exporting and archiving, *MAINT*, 10-69
- exporting and containing no data,
MAINT, 10-66
- exporting in VAXcluster environment,
MAINT, 18-31
- failure to attach to, *GUSQL*, 11-38

Database (Cont.) (Cont.)

- field value expression, *RDMLRM*, 2-9
- handle, *RDORM*, 3-4, 9-99
- handling bugcheck dumps, *MAINT*, 12-2
- importing, *RDORM*, 9-299; *SQLRM*, 6-391
- importing and changing device specifications, *MAINT*, 10-68
- importing and containing no data,
MAINT, 10-66
- importing and using TRACE,
MAINT, 10-66
- importing from Rdb/ELN databases,
MAINT, 2-54
- importing in VAXcluster environment,
MAINT, 18-31
- improving performance, *TUNING*, 2-3
 - by normalizing, *MAINT*, 16-11
- improving verify performance,
MAINT, 6-13
- inactive records, *MAINT*, 16-78
- information
 - SHOW ALL* statement, *RDORM*, 9-375
 - SHOW COLLATING_SEQUENCE* statement, *RDORM*, 9-376
 - SHOW CONSTRAINT* statement,
RDORM, 9-377
 - SHOW DATABASES* statement,
RDORM, 9-379
 - SHOW DICTIONARY* statement,
RDORM, 9-383
 - SHOW FIELDS* statement,
RDORM, 9-384
 - SHOW INDEXES* statement,
RDORM, 9-387
 - SHOW PRIVILEGES* statement,
RDORM, 9-392
 - SHOW PROTECTION* statement,
RDORM, 9-395

Database (Cont.)

information (Cont.)

- SHOW RELATIONS statement, *RDORM*, 9-398
- SHOW statements, *RDORM*, 9-372
- SHOW STORAGE AREAS statement, *RDORM*, 9-400
- SHOW STORAGE MAPS statement, *RDORM*, 9-404
- SHOW STREAMS statement, *RDORM*, 9-406
- SHOW TRANSACTION statement, *RDORM*, 9-407
- SHOW TRANSFER statement, *RDORM*, 9-409
- SHOW USERS statement, *RDORM*, F-17
- SHOW VERSIONS statement, *RDORM*, 9-416
- SQL SHOW statement, *SQLRM*, 6-521
- inserting row in, *GUSQL*, 5-3; *SQLRM*, 6-414
- integrity, *MAINT*, 6-4, 16-80
- integrity error, *GUSQL*, 11-27
- internal page structures, *MAINT*, 11-27
- interpreting statistics, *MAINT*, 15-10
- interrelated database performance parameters, *MAINT*, 16-9t
- investigating structure of, *GUSQL*, 6-2
- invoking, *GURRR*, 2-1
- journaling, *MAINT*, 9-1
- key
 - See Database key
- listing AIJ file, *RDORM*, 6-59
- listing backup file, *RDORM*, 6-62
- listing RUJ file, *RDORM*, 6-68
- loading data, *MAINT*, 16-129; *GUSQL*, 5-1; *RDORM*, 6-70

Database (Cont.)

loading data (Cont.)

- changing database definitions, *MAINT*, 2-45
- defining hashed indexes before, *DESIGN*, 4-13
- from RMS files, *MAINT*, 2-21
- using DATATRIEVE, *MAINT*, 2-46
- with RMU/LOAD command, *MAINT*, 2-37
- logical design, *DESIGN*, 2-1 to 2-15
- maintenance statements summary, *RDORM*, 2-7
- metadata updates, *DESIGN*, 5-3; *MAINT*, 10-2
- migration, *MAINT*, 10-70
- modifying data, *RDORM*, 9-338; *GURRR*, 6-4; *SQLRM*, 6-560
- modifying database definitions, *RDORM*, 9-9; *SQLRM*, 6-15
- modifying dictionary definitions, *DESIGN*, 7-13
- modifying using EXPORT and IMPORT statements, *MAINT*, 10-52
- monitoring, *MAINT*, 18-32
 - in VAXcluster, *MAINT*, 18-30
 - process, *MAINT*, 3-2
 - starting, *RDORM*, 6-80
 - stopping, *RDORM*, 6-79, 6-82
 - using, *MAINT*, 3-1
- moving
 - dictionary information, *MAINT*, 8-58
 - files, *MAINT*, 8-57
 - root file, *RDORM*, 6-85
 - storage areas, *MAINT*, 10-31, 10-35; *RDORM*, 6-85
 - using EXPORT/IMPORT statements, *DESIGN*, 3-11
 - using RMU, *DESIGN*, 3-10
- moving files, *MAINT*, 7-16
- multifile, *DESIGN*, 3-8

Database (Cont.)

multifile (Cont.)

- as opposed to single-file,
DESIGN, 3-8; *TUNING*, 2-3
- normalizing
 - to improve performance, *MAINT*, 16-11
- NUMBER OF BUFFERS parameter,
MAINT, 16-91
- opening, *MAINT*, 5-2; *RDORM*, 6-89
- OPEN IS option, *MAINT*, 10-10
- PAGE SIZE parameter, *MAINT*, 16-109
- parameters, *MAINT*, 16-86
 - See also* Parameter
 - default values, *RDORM*, D-1
 - maximum values, *RDORM*, D-1
 - minimum values, *RDORM*, D-1
- performance
 - understanding your data,
MAINT, 16-4
 - understand youring data,
MAINT, 16-5t
- performance evaluation, *MAINT*, 13-1
 - database statistics, *MAINT*, 13-6
 - hardware resources, *MAINT*, 13-5
 - locking, *MAINT*, 13-6
 - operating system resources,
MAINT, 13-5
 - operating system utilities,
MAINT, 13-7
 - sample procedure, *MAINT*, 13-11
 - space usage, *MAINT*, 13-5
 - VAXcluster environment,
MAINT, 13-9
 - VAX Performance Advisor (VPA),
MAINT, 13-7
- performance issues, *MAINT*, 16-1
- performance-related changes,
MAINT, 13-11

Database (Cont.)

physical design

- changes, *MAINT*, 10-1
- default values
 - database-wide, *MAINT*, 16-2t; *RDORM*, D-1t
 - storage area parameter,
MAINT, 16-3t; *RDORM*, E-1t
 - storage map parameter,
MAINT, 16-4t
- implementing with minimal effort,
MAINT, 16-2t
- problems with redundancy, *MAINT*, 16-16
- protection, *SQLRM*, 6-359
 - CHANGE PROTECTION
 - statement, *RDORM*, 9-43
 - DEFINE PROTECTION
 - statement, *RDORM*, 9-132
 - DELETE PROTECTION
 - statement, *RDORM*, 9-241
- query optimizer, *MAINT*, 17-1;
GURRR, 2-28
- Rdb/ELN
 - converting to Rdb/VMS database,
MAINT, 2-54
- read-only storage areas
 - backup, *MAINT*, 8-25e
 - considerations, *MAINT*, 8-25
 - restore, *MAINT*, 8-45e
- reconstructing, *RDORM*, 6-91
- record fragmentation, *MAINT*, 16-93
- records
 - adding
 - STORE statement, *RDORM*, 9-457
 - deleting
 - ERASE statement, *RDORM*, 9-263
 - storage structures, *MAINT*, 11-27
- recovering, *RDORM*, 6-91
- recovery, *MAINT*, 9-1, 9-18, 12-2

Database (Cont.)

recovery (Cont.)

- in VAXclusters, *MAINT*, 18-29
- re-creating lost AIJ file, *MAINT*, 9-17
- redundant data, *MAINT*, 16-16
- remote access, *GURRR*, 2-3
- reorganizing, *MAINT*, 10-53
 - single to multifile, *MAINT*, 8-59
- reporting bugcheck dumps, *MAINT*, 12-8
- resolving an unresolved transaction, *RDORM*, 6-103
- resolving distributed transaction, *DIST_TRANS*, 1-7, 2-17, 6-1 to 6-7
- resource locking, *MAINT*, 16-17
- restoring, *MAINT*, 8-38, 8-44e; *RDORM*, 6-107; *DIST_TRANS*, 6-6
 - by area, *RDORM*, 6-112
 - incremental, *MAINT*, 8-42
 - modifying after-image journaling, *MAINT*, 8-54
 - modifying database characteristics, *MAINT*, 8-52
 - modifying page size, *MAINT*, 8-56
 - modifying thresholds, *MAINT*, 8-56
 - using multiple tape drives, *MAINT*, 8-50
 - using single tape drive, *MAINT*, 8-49
- restructuring, *DESIGN*, 5-1 to 5-36
- root file (RDB), *DESIGN*, 4-2
- single-file, *DESIGN*, 3-8
 - as opposed to multifile, *DESIGN*, 3-8
- snapshot (SNP) file, *DESIGN*, 3-11, 4-2
- snapshot pages, *MAINT*, 11-27
- snapshot storage area, *DESIGN*, 3-54

Database (Cont.) (Cont.)

- space area management (SPAM)
 - pages, *MAINT*, 11-55
- space usage, *MAINT*, 14-1
- statistics
 - ANALYZE* statement, *RDORM*, 9-2
 - display, *RDORM*, 6-148
 - interpreting, *MAINT*, 15-10
- RMU/ANALYZE* command, *RDORM*, 6-5
- RMU/SHOW STATISTICS* command, *RDORM*, 6-148
- storage area (RDA) file, *DESIGN*, 4-2
 - types of pages, *MAINT*, 11-10
- storage area structures, *MAINT*, 11-27
- storage map parameters, *MAINT*, 16-116
- strategy to detect problems, *MAINT*, 6-7
- system relations, *RDORM*, 8-1
- tuning
 - defined, *TUNING*, 1-1
 - methodology, *TUNING*, 1-5
 - transaction type, *MAINT*, 16-92
 - what to tune, *TUNING*, 1-10
 - when to tune, *TUNING*, 1-1
 - types of data storage, *MAINT*, 11-2
 - types of statistics, *MAINT*, 15-10
- unloading, *DESIGN*, 5-26; *RDORM*, 6-158
 - record definition format, *MAINT*, 2-39
 - with *RMU/UNLOAD* command, *MAINT*, 2-37
- updating, *SQLRM*, 6-414
- using space management, *MAINT*, 11-65
- using VAXcluster nodes, *MAINT*, 18-2, 18-19
- verification
 - troubleshooting, *MAINT*, 6-18

Database (Cont.)

verification

troubleshooting (Cont.)

checksum check, *MAINT*,
6-19

data integrity, *MAINT*, 6-21
summary, *MAINT*, 6-25

verifying, *MAINT*, 6-1

integrity, *RDORM*, 6-161

problems checked by verify,
MAINT, 6-4

process of, *MAINT*, 6-4

protection for, *DESIGN*, 6-34

reasons for, *MAINT*, 6-2

what verify detects, *MAINT*, 6-6

what to do if AIJ file is inaccessible,
MAINT, 9-17

what verify checks, *MAINT*, 6-9

Database administrator (DBA)

See Maintenance

Database conversion

from V2.3 to V3.0 or V3.1 to V4.0,
MAINT, 9-24

from V2.3 to V4.0, *MAINT*, 9-22

from V3.0 or V3.1 to V4.0, *MAINT*,
9-26

strategy and procedure, *MAINT*,
9-22

Database corruption, *MAINT*, 6-26e

causes, *MAINT*, 6-3

clearing a corruption flag, *MAINT*,
7-3

clearing an inconsistent flag, *MAINT*,
7-22

data integrity, *MAINT*, 6-36e

page header, *MAINT*, 6-27e

patching, *MAINT*, 7-1

sorted index, *MAINT*, 6-32e

Database design

See also Logical design; Physical
design

and concealed logical names,
DESIGN, 3-10

Database design (Cont.)

choosing between single-file and
multifile, *DESIGN*, 3-8

CREATE STORAGE MAP statement,
DESIGN, 3-10

defining protection, *DESIGN*, 3-8,
6-1

entity-relationship map, *DESIGN*,
2-3

for distributed transaction, *DIST_*
TRANS, 3-1 to 3-9

logical, *DESIGN*, 1-4

methods of, *DESIGN*, 1-3

normalization, *DESIGN*, 2-8

physical, *DESIGN*, 1-4, 1-6

prototype, *DESIGN*, 4-1

prototype transactions, *DESIGN*,
2-12

requirements analysis, *DESIGN*, 2-1
storage, *DESIGN*, 4-1

transaction analysis, *DESIGN*, 2-9

using interactive SQL, *DESIGN*, 3-3

using primary and foreign keys,
DESIGN, 2-3

using RdbExpert, *DESIGN*, 1-6

using the data dictionary, *DESIGN*,
3-4, 7-1 to 7-28

volume tables, *DESIGN*, 2-14

Database file

updating

using dictionary definitions,
DESIGN, 7-22

Database files

moving, *MAINT*, 7-16

Database handle, *RDORM*, 9-74, 9-99

scope of, *RDORM*, 9-329

using to distinguish between
databases, *GURRR*, 9-38

Database handle clause, *RDMLRM*, 6-20

Database key, *DESIGN*, 3-13;

RDMLRM, 2-26; *SQLRM*, 3-71

compressed dbkey, *MAINT*, 11-40

controlling scope of, *GURRR*, 9-32

Database key (Cont.)

- hashed index, *MAINT*, 11-41, 14-39, 14-60, 16-35, 16-73
 - overflow, *MAINT*, 11-49
- PLACE statement, *MAINT*, 16-129
- indexes, *MAINT*, 11-49, 14-1, 16-34, 16-40, 17-5
- index node segments, *MAINT*, 11-38
- scope, *DESIGN*, 3-13
- segmented string, *MAINT*, 11-37
- sorted indexes, *MAINT*, 16-34
 - duplicate index node, *MAINT*, 11-41
- sorted index node links, *MAINT*, 16-40
- specifying as value expression, *GUSQL*, 3-5
- uncompressed dbkey, *MAINT*, 11-41, 16-51
- using to retrieve records, *GURRR*, 9-33

Database keys (dbkeys), *RDORM*, 3-32

Database maintenance overview, *MAINT*, 1-9

Database metadata duplicating, *DESIGN*, 5-23

Database names CDD/Plus restrictions, *RDORM*, 9-97

Database OPEN IS option changing, *MAINT*, 10-10

Database page, *MAINT*, 11-27e area bit maps, *MAINT*, 11-62 - uniform page format, *MAINT*, 11-63e

area inventory pages uniform page format, *MAINT*, 11-64e

checksum, *MAINT*, 11-29

common format used, *MAINT*, 11-11e

displaying contents, *MAINT*, 7-11

Database page (Cont.)

- fragmented record, *MAINT*, 11-51, 11-52e
- free space, *MAINT*, 11-29, 11-31
- fullness threshold, *MAINT*, 11-55
- hashed index, *MAINT*, 11-42e
 - structure with page pointer, *MAINT*, 11-49f
- header, *MAINT*, 11-28, 11-29e
- index node record, *MAINT*, 11-38, 11-39e
- line index, *MAINT*, 11-30, 11-30e
- locked and unlocked free space, *MAINT*, 11-29, 11-32e, 11-33f
- mixed page format
 - SPAM intervals, *MAINT*, 11-68f
 - SPAM page, *MAINT*, 11-65e
- number, *MAINT*, 11-29
- page tail
 - mixed storage area, *MAINT*, 11-50e
 - uniform storage area, *MAINT*, 11-51e
- segmented string record, *MAINT*, 11-36e
- snapshots, *MAINT*, 11-27
- space area management (SPAM)
 - pages, *MAINT*, 11-57e
- SPAM entry information, *MAINT*, 11-73t
- SPAM interval formula, *MAINT*, 11-69
- storage area number, *MAINT*, 11-29
- storage segment structure, *MAINT*, 11-33
- structure with system record, *MAINT*, 11-48f
- time and date stamp, *MAINT*, 11-29
- transaction sequence number index, *MAINT*, 11-31e
- user-stored record, *MAINT*, 11-34e

Database parameters

- default values, *RDORM*, D-1
- maximum values, *RDORM*, D-1

Database parameters (Cont.)
 minimum values, *RDORM*, D-1

Database recovery (DBR), *MAINT*, 9-2
 bugcheck, *MAINT*, 12-2

Database recovery (DBR) process
 in distributed transaction, *DIST_*
TRANS, 2-9, 2-15

DATABASE statement, *RDMLRM*, 6-11

Database statistics, *MAINT*, 15-1,
 15-10
 RMU/ANALYZE command, *RDORM*,
 6-5

Database storage area, *DESIGN*, 4-2
See also Storage area
 adjusting parameters, *MAINT*,
 16-106, 16-107
 parameters, *MAINT*, 16-106

Database storage map
 adjusting parameters for performance,
MAINT, 16-116

Database table
 definition for storing security audit
 journal records, *MAINT*, 4-25
 for storing security audit journal
 records, *MAINT*, 4-25; *RDORM*,
 6-71

Database verification, *MAINT*, 6-4
 after changes in RdbALTER, *MAINT*,
 7-23
 after using RdbALTER, *MAINT*,
 7-16
 devising a strategy, *MAINT*, 6-7
 examples of running verify operations,
MAINT, 6-14
 improving verify performance,
MAINT, 6-13
 problems checked by verify, *MAINT*,
 6-4
 process of, *MAINT*, 6-4
 reasons to perform, *MAINT*, 6-2
 troubleshooting, *MAINT*, 6-18
 checksum check, *MAINT*, 6-19
 data integrity, *MAINT*, 6-21
 summary, *MAINT*, 6-25

Database verification (Cont.)
 what verify checks, *MAINT*, 6-9
 what verify detects, *MAINT*, 6-6

Data compression
 considerations, *MAINT*, 16-128
 disabling, *MAINT*, 16-118
 enabling, *MAINT*, 16-118, 16-121e

Data definition
 performing in RDML program,
RDMLRM, 1-3
 statements
 Callable RDO, *GURRR*, 19-29
 summary, *RDORM*, 2-2;
SQLRM, 2-2

Data dictionary
 allowing access to database, *GUSQL*,
 1-22
 anchor, *DESIGN*, 7-2
 copying definitions
 using FROM clause, *GUSQL*,
 10-10
 using INCLUDE statement,
GUSQL, 10-8
 using SQL module language,
SQLRM, 4-10
 creating data definitions with,
DESIGN, 7-6e
 creating shareable definitions,
DESIGN, 7-6
 criteria for using, *DESIGN*, 7-4
 defined, *DESIGN*, 7-1
 defining protection for shareable
 Rdb/VMS definitions, *DESIGN*,
 6-37
 definitions
 deleting, *DESIGN*, 7-26;
SQLRM, 6-308
 field, *DESIGN*, 7-1
 including in RDBPRE programs,
GURRR, 12-4
 including in RDML programs,
GURRR, 16-3
 including in SQL module language
 programs, *SQLRM*, 4-10

Data dictionary

definitions (Cont.)

modifying, *DESIGN*, 7-13

updating, *SQLRM*, 6-424

ENTER command, *DESIGN*, 7-17e

integrating with database, *DESIGN*, 7-15

naming conventions of, *DESIGN*, 7-2

path, *DESIGN*, 7-2

path names, *SQLRM*, 3-9

specifying access for database attachment, *GUSQL*, 2-6

specifying in database definition, *DESIGN*, 3-4

tracking entities, *DESIGN*, 7-1

tradeoffs of using, *DESIGN*, 3-4, 7-27

using to create domains, *DESIGN*, 3-14

using to create tables, *DESIGN*, 3-23, 3-24

using VAXcluster nodes, *MAINT*, 18-18

using with Rdb/VMS, *DESIGN*, 7-1 to 7-28

Data Distributor

See also VAX Data Distributor

Data Distributor statements (SQL)

CREATE SCHEDULE, *SQLRM*, 6-88

CREATE TRANSFER, *SQLRM*, 6-168

DROP SCHEDULE, *SQLRM*, 6-309

DROP TRANSFER, *SQLRM*, 6-320

REINITIALIZE TRANSFER, *SQLRM*, 6-450

SHOW TRANSFER, *SQLRM*, 6-549

START TRANSFER, *SQLRM*, 6-554

STOP TRANSFER, *SQLRM*, 6-558

Data manipulation statements

Callable RDO programs, *GURRR*, 19-6

summary, *RDORM*, 2-5; *SQLRM*, 2-5

Data page

contents of, *MAINT*, 11-27e

Data retrieval methods, *DESIGN*, 1-10

Data selection

See Retrieving data; SELECT statement

Data storage, *MAINT*, 11-2

Data structure, *SQLSRV*, 10-1

ASSOCIATE_STR, *SQLSRV*, 2-12, 10-3

documentation format, *SQLSRV*, 10-1

SQLCA, *SQLSRV*, 2-12, 10-6

SQLDA, *SQLSRV*, 2-12, 10-11

SQLSRV_ENV_STR, *SQLSRV*, 2-13, 10-16

SQLVAR, *SQLSRV*, 10-13

summary of, *SQLSRV*, 2-12 to 2-13

used in SQL/Services, *SQLSRV*, 2-12 to 2-13

DATATRIEVE

formatting clauses, *DESIGN*, 3-22; *SQLRM*, 3-49

loading Rdb/VMS databases, *MAINT*, 2-46

support clause, *DESIGN*, 3-16, 3-22; *RDORM*, 5-16

DEFAULT_VALUE clause, *RDORM*, 5-17

EDIT_STRING clause, *RDORM*, 5-17

QUERY_HEADER clause, *RDORM*, 5-17

QUERY_NAME clause, *RDORM*, 5-17

Data type, *RDORM*, 5-5; *GURRR*, A-1; *SQLSRV*, 11-1; *SQLRM*, 3-32

BASIC equivalents, *GURRR*, 8-7t

BYTE VARYING, *SQLRM*, 3-37

C equivalents, *GURRR*, 8-8, 8-9

CHAR, *DESIGN*, 3-18; *SQLRM*, 3-34

character, *SQLRM*, 3-34

Data type (Cont.)

conversion, *GURRR*, 8-4; *SQLRM*, 3-39
DATE, *DESIGN*, 3-18; *RDORM*, 5-6; *SQLRM*, 3-36
DECIMAL, *SQLRM*, 3-35
DOUBLE PRECISION, *SQLRM*, 3-36
fixed-point numeric, *SQLRM*, 3-35
FLOAT, *SQLRM*, 3-36
floating-point numeric, *SQLRM*, 3-36
FORTRAN equivalents, *GURRR*, 8-10t
generated by RDML, *RDMLRM*, A-1
for VAX C, *RDMLRM*, A-1
for VAXELN Pascal, *RDMLRM*, A-3
for VAX Pascal, *RDMLRM*, A-2
host language equivalent, *GURRR*, 8-7
INTEGER, *DESIGN*, 3-19; *SQLRM*, 3-35
list, *INTRO*, 1-5
LIST OF BYTE VARYING, *DESIGN*, 3-19; *SQLRM*, 3-37
LIST OF VARBYTE
 See LIST OF BYTE VARYING
LONG VARCHAR, *SQLRM*, 3-34
NUMERIC, *SQLRM*, 3-35
Pascal equivalents, *GURRR*, 8-11t
QUADWORD, *SQLRM*, 3-35
Rdb/VMS compared to VMS, *GURRR*, 8-2t
REAL, *RDORM*, 5-6; *SQLRM*, 3-36
segmented string, *GURRR*, 8-3
SEGMENTED STRING, *RDORM*, 5-8
SMALLINT, *DESIGN*, 3-19; *SQLRM*, 3-35
specifying, *DESIGN*, 3-18, 3-27
\$SQL_VARCHAR, *GUSQL*, 8-4, 10-33
summary for Rdb/VMS, *RDORM*, 5-7

Data type (Cont.)

supported by Rdb/VMS, *GURRR*, 8-1 to 8-13
TINYINT, *SQLRM*, 3-35
used in SQL module language, *SQLRM*, 4-22
VARBYTE, *SQLRM*, 3-37
VARCHAR, *DESIGN*, 3-19; *SQLRM*, 3-34
Data validation, *MAINT*, 16-80; *RDORM*, 5-11
DATE data type, *DESIGN*, 3-18; *RDORM*, 5-6; *SQLRM*, 3-36
converting in programs, *SQLRM*, 3-40
format of, *RDORM*, 3-10
Date format, *GUSQL*, 3-24
displaying language for, *RDORM*, 9-391
SET DATE_FORMAT statement, *RDORM*, 9-363
SHOW DATE_FORMAT statement, *RDORM*, 9-381
Date literal
 conversion to international format, *RDMLRM*, 1-5
DATE literal, *SQLRM*, 3-46
DBA
 See Maintenance
Dbkey
 See Database key
DCL
 accessing, *SQLRM*, 6-231
 SHOW AUDIT/JOURNAL command, *MAINT*, 4-27
DCL invoke statement, *SQLRM*, 6-231
 accessing DCL, *RDORM*, 9-81
Deadlock
 avoiding, *SQLRM*, 6-507
Deadlock error, *GUSQL*, 2-32, 11-25
 avoiding with distributed transaction, *DIST_TRANS*, 4-4
DEADLOCKWAIT parameter
 values, *MAINT*, 16-138

Debugger

See VMS Debugger

Debugging program, *GUSQL*, 9-6;
GURRR, 11-24

using interactive SQL, *GUSQL*, 6-5

DECdtm services, *DIST_TRANS*, 1-4
coordinating distributed transaction
with, *DIST_TRANS*, 1-4, 2-2

DECIMAL data type, *SQLRM*, 3-35

DECLARE CURSOR statement,
GUSQL, 4-1; *SQLRM*, 6-233
in dynamic SQL

See also Dynamic DECLARE
CURSOR statement

See also Extended Dynamic
DECLARE CURSOR
statement

in SQL module, *GUSQL*, 7-17;
SQLRM, 4-18, 4-20

DECLARE SCHEMA statement,
GUSQL, 2-5; *SQLRM*, 6-255
in SQL module, *GUSQL*, 7-15

DECLARE statement
in SQL module, *GUSQL*, 7-12

DECLARE STATEMENT statement,
SQLRM, 6-266

DECLARE TABLE statement, *SQLRM*,
6-268

DECLARE TRANSACTION statement,
GUSQL, 1-22, 2-18, 2-21, 2-22e,
2-43; *SQLRM*, 6-273

WAIT clause, *GUSQL*, 2-30;
DIST_TRANS, 4-4

DECLARE_STREAM statement,
RDMLRM, 6-26; *GURRR*, 6-8
record stream, *RDORM*, 9-82

DECLARE_VARIABLE clause,
RDMLRM, 6-32

DECLARE_VARIABLE clause (RDML),
GURRR, 17-7, 18-6

Declaring

cursors, *GUSQL*, 4-1; *SQLRM*,
6-233

Declaring (Cont.)

databases, *GURRR*, 9-3
function and type, *RDMLRM*, 6-4
host language parameter, *GUSQL*,
10-2

multiple databases, *GURRR*, 9-4
schemas, *DESIGN*, 6-7; *GUSQL*,
2-3; *SQLRM*, 6-255

statements, *SQLRM*, 6-266

tables, *SQLRM*, 6-268

transactions, *SQLRM*, 6-273

DEC Multinational Character Set (MCS)
See Multinational Character Set
(MCS)

DECnet software

allowing client/server communication,
SQLSRV, 1-4

use with SQL/Services, *SQLSRV*, 1-2

DECTrace

collecting event data on Rdb/VMS
applications, *MAINT*, 15-50

collecting workload information for
RdbExpert, *MAINT*, 15-60

collection class defined, *MAINT*,
15-50

creating a customized report,
MAINT, 15-72

creating a facility selection, *MAINT*,
15-57

creating a report
formatting and merging data files,
MAINT, 15-61

creating a report from collected data,
MAINT, 15-61

creating a summary report
specifying the statistics, *MAINT*,
15-71

events defined, *MAINT*, 15-49

facility defined, *MAINT*, 15-57

gathering statistics on Rdb/VMS
applications, *MAINT*, 15-49

generating a report, *MAINT*, 15-62

items defined, *MAINT*, 15-50

DECTrace (Cont.)

- Rdb/VMS ALL collection class of events and items, *MAINT*, 15-55t
- Rdb/VMS database relations in ALL collection classes, *MAINT*, 15-78
- in WORKLOAD collection classes, *MAINT*, 15-88
- Rdb/VMS data items, *MAINT*, 15-52t
- Rdb/VMS events, *MAINT*, 15-51t
- Rdb/VMS PERFORMANCE collection class of events and items, *MAINT*, 15-56t
- Rdb/VMS resource utilization items, *MAINT*, 15-51t
- Rdb/VMS WORKLOAD collection class of events and items, *MAINT*, 15-56t
- scheduling data collection, *MAINT*, 15-58
- specifying the processes to collect data, *MAINT*, 15-59
- using registration IDs, *MAINT*, 15-59
- Default protection, *DESIGN*, 6-3
- Default value, *GUSQL*, 5-5
 - See also* Missing value; Null for column, *DESIGN*, 3-30
 - for domain, *DESIGN*, 3-20
- DEFAULT_VALUE FOR DTR clause, *RDORM*, 5-17
- DEFINE COLLATING_SEQUENCE statement, *RDORM*, 9-85
- DEFINE CONSTRAINT statement restricting values, *RDORM*, 9-88
- DEFINE DATABASE statement creating a database, *RDORM*, 9-93
- DEFINE FIELD statement creating field definitions, *RDORM*, 9-112

- DEFINE INDEX statement creating index definitions, *RDORM*, 9-119
- DEFINE PROTECTION statement accessing the database, *RDORM*, 9-132
- DEFINE RELATION statement creating relation definitions, *RDORM*, 9-149
- DEFINE SCHEDULE statement defining schedules, *RDORM*, 9-164
- DEFINE STORAGE MAP statement defining storage maps, *RDORM*, 9-171
- DEFINE TRANSFER statement, *RDORM*, 9-180
- DEFINE TRIGGER statement creating trigger definitions, *RDORM*, 9-204
- DEFINE VIEW statement, *GURRR*, 5-2
 - creating view definitions, *RDORM*, 9-219
- DEFINE_TYPE clause, *RDMLRM*, 6-35
- Defining
 - data definition statements, *RDORM*, 2-2; *SQLRM*, 2-2
 - dynamic SQL statements, *SQLRM*, 2-9
 - indexes, *RDORM*, 9-119
 - privileges, *SQLRM*, 6-359, 6-379
 - protection, *RDORM*, 9-132
 - for shareable Rdb/VMS definitions in CDD/Plus, *DESIGN*, 6-37
 - through a command procedure, *DESIGN*, 6-19
 - protection for views, *DESIGN*, 6-22
 - storage maps
 - DEFINE STORAGE MAP statement, *RDORM*, 9-171
 - for tables in multifile databases, *DESIGN*, 4-4
 - user privileges, *RDORM*, 9-136

Defining (Cont.)

- views, *GURRR*, 5-2, 5-4
- DELETE COLLATING_SEQUENCE**
 - statement
 - deleting collating sequences, *RDORM*, 9-225
- DELETE CONSTRAINT** statement
 - deleting constraints, *RDORM*, 9-228
- DELETE DATABASE** statement
 - deleting database definitions, *RDORM*, 9-232
- DELETE FIELD** statement
 - deleting field definitions, *RDORM*, 9-234
- DELETE INDEX** statement
 - deleting indexes, *RDORM*, 9-237
- DELETE PATHNAME** statement
 - deleting dictionary definitions, *RDORM*, 9-240
- DELETE PROTECTION** statement
 - deleting protection, *RDORM*, 9-241
- DELETE RELATION** statement
 - deleting relations, *RDORM*, 9-244
- DELETE SCHEDULE** statement
 - deleting schedules, *RDORM*, 9-247
- DELETE** statement, *GUSQL*, 5-18;
SQLRM, 6-283
 - in SQL module, *GUSQL*, 7-19
- DELETE STORAGE MAP** statement
 - deleting storage maps, *RDORM*, 9-249
- DELETE TRANSFER** statement
 - deleting transfers, *RDORM*, 9-251
- DELETE TRIGGER** statement
 - deleting triggers, *RDORM*, 9-253
- DELETE VIEW** statement
 - deleting views, *RDORM*, 9-255

Deleting

- cascading, *DESIGN*, 3-38
- collating sequence, *SQLRM*, 6-295
- collating sequences, *RDORM*, 9-225
- columns, *DESIGN*, 5-7; *SQLRM*, 6-316

Deleting (Cont.)

- constraints, *DESIGN*, 5-7, 5-17;
RDORM, 9-228; *SQLRM*, 6-298
 - data, *GURRR*, 6-7; *SQLRM*, 6-283
 - database definitions, *RDORM*, 9-232
 - database files, *DESIGN*, 5-23
 - data using triggers, *GURRR*, 6-8
 - dictionary definitions, *DESIGN*, 7-27; *RDORM*, 9-240; *SQLRM*, 6-308
 - using SQL and CDO, *DESIGN*, 7-26
 - domains, *DESIGN*, 5-6; *SQLRM*, 6-301
 - field definitions, *RDORM*, 9-234
 - indexes, *DESIGN*, 5-22; *RDORM*, 9-237; *SQLRM*, 6-305
 - protection, *RDORM*, 9-241
 - relations, *RDORM*, 9-244
 - rows, *GUSQL*, 5-18; *SQLRM*, 6-283
 - schedules, *RDORM*, 9-247
 - schema definitions, *SQLRM*, 6-311
 - schemas, *DESIGN*, 3-8, 5-23
 - storage areas, *DESIGN*, 3-7
 - storage map definitions, *SQLRM*, 6-313
 - storage maps, *RDORM*, 9-249
 - tables, *DESIGN*, 5-7; *SQLRM*, 6-316
 - transfers, *RDORM*, 9-251
 - triggers, *DESIGN*, 5-20; *RDORM*, 9-253; *SQLRM*, 6-322
 - views, *DESIGN*, 5-21; *RDORM*, 9-255; *SQLRM*, 6-325
- ## Demonstration
- running online, *INTRO*, 1-2
- ## DEPOSIT (RdbALTER)
- moving database files, *MAINT*, 7-17
 - to move database files, *MAINT*, 7-17
- ## DEPOSIT command (RdbALTER),
- MAINT*, 7-13; *RDORM*, 7-8
 - restrictions, *MAINT*, 7-13
- ## DEPOSIT FILE command (RdbALTER),
- RDORM*, 7-14

- DEPOSIT ROOT command
 - (RdbALTER), *RDORM*, 7-16
- Descending index, *RDORM*, 9-123
- Descending value
 - ordering row in, *GUSQL*, 3-19
- DESC keyword
 - in ORDER BY clause, *GUSQL*, 3-19
- DESCRIBE statement, *GUSQL*, 12-5; *SQLRM*, 6-287
- Design concepts
 - creation parameters, *MAINT*, 11-65
- Designing
 - databases
 - creation parameters, *MAINT*, 11-65
 - for distributed transaction, *DIST_TRANS*, 3-1 to 3-9
 - snapshot storage areas, *DESIGN*, 3-54
- Design model, *DESIGN*, 1-3
- DETACH
 - in RdbALTER, *MAINT*, 7-25
- DETACH command (RdbALTER), *MAINT*, 7-3; *RDORM*, 7-17
- Detached process, *GURRR*, 9-2
- Detaching
 - from a database, *GUSQL*, 2-16; *GURRR*, 9-4
 - in RdbALTER, *MAINT*, 7-3
 - from declared schemas, *SQLRM*, 6-357
- Developing an Rdb/VMS program
 - BASIC, *GURRR*, 7-3
 - C, *GURRR*, 7-3
 - Callable RDO, *GURRR*, 7-3
 - COBOL, *GURRR*, 7-3
 - FORTRAN, *GURRR*, 7-3
 - Pascal, *GURRR*, 7-3
- Developing SQL program
 - guidelines, *GUSQL*, 6-2
- Dictionary definition
 - copying with INTEGRATE DATABASE, *RDORM*, 9-325
 - creating, *DESIGN*, 7-15, 7-20
- Dictionary definition (Cont.)
 - creating for schema, *DESIGN*, 7-11
 - deleting, *DESIGN*, 7-26, 7-27; *RDORM*, 9-240
 - integrating with database, *DESIGN*, 7-15
 - re-creating with INTEGRATE DATABASE, *RDORM*, 9-325
- Dictionary Management Utility (DMU)
 - See DMU
- DICTIONARY OPERATOR
 - DCL command, *DESIGN*, 7-2
- Dictionary path names
 - displaying
 - SHOW DICTIONARY statement, *RDORM*, 9-383
- Differing terminology, *INTRO*, 1-11t
- DIOLM parameter
 - values, *MAINT*, 16-140
- DIRECTORY command
 - in CDO, *DESIGN*, 7-20e
- Disabling audit event information, *MAINT*, 4-16
- Disabling DACCESS audit events, *MAINT*, 4-5, 4-13; *RDORM*, 6-128
- Disabling security alarms, *RDORM*, 6-128
- Disabling two-phase commit protocol, *DIST_TRANS*, 4-3
- Disk
 - dual-ported MSCP-served, *MAINT*, 18-7
- Disk drives
 - assigning snapshot and data storage areas to different, *DESIGN*, 4-3
 - table distribution among, *DESIGN*, 4-8
 - using more than one for a table, *DESIGN*, 4-9
- Disk files
 - reducing disk I/O contention, *MAINT*, 16-114
 - sharing in VAXclusters, *MAINT*, 18-6

- Disk space
 - specifying default allocation for, *DESIGN*, 3-12
- Display
 - interactive, *GUSQL*, 1-14
- DISPLAY command (RdbALTER), *MAINT*, 7-11; *RDORM*, 7-18
- DISPLAY FILE command (RdbALTER), *RDORM*, 7-25
- Displaying
 - access rights, *RDORM*, 9-392
 - all database information, *RDORM*, 9-375
 - collating sequence, *RDORM*, 9-376
 - constraints, *RDORM*, 9-377
 - current dictionary default directory, *RDORM*, 9-383
 - database information, *RDORM*, 6-54, 6-142, 9-379
 - using RMU, *RDORM*, 6-1
 - database page contents, *MAINT*, 7-11
 - date format, *RDORM*, 9-381
 - field names, *RDORM*, 9-384
 - index names, *RDORM*, 9-387
 - information interactively, *GUSQL*, 1-5
 - language for date format, *RDORM*, 9-391
 - names of streams, *RDORM*, 9-406
 - privileges, *RDORM*, 9-392, 9-395
 - radix point character, *RDORM*, 9-397
 - relation definitions, *RDORM*, 9-398
 - schema and database information, *SQLRM*, 6-521 to 6-547
 - storage area names, *RDORM*, 9-400
 - storage map names, *RDORM*, 9-404
 - time format, *RDORM*, 9-381
 - transaction information, *RDORM*, 9-407
 - transfer information, *RDORM*, 9-409
 - triggers, *RDORM*, 9-413
 - version number, *RDORM*, 9-416
- DISPLAY ROOT command (RdbALTER), *RDORM*, 7-27
- DISTINCT keyword, *GUSQL*, 3-13, 3-54
- DISTRIBTRAN privilege, *DIST_TRANS*, 3-6
- Distributed lock manager, *MAINT*, 18-9
- Distributed transaction, *RDMLRM*, 6-130; *DIST_TRANS*, 1-2
 - completed unresolved transaction in corrupt database, *DIST_TRANS*, 6-5
 - completing, *DIST_TRANS*, 1-7, 2-17, 6-1 to 6-7
 - defined, *DIST_TRANS*, 1-1
 - designing database for, *DIST_TRANS*, 3-1 to 3-9
 - error handling, *DIST_TRANS*, 4-7, 5-12
 - privilege needed, *DIST_TRANS*, 3-6
 - starting, *DIST_TRANS*, 2-2
 - in VAXcluster, *DIST_TRANS*, 3-6
- START_TRANSACTION statement, *RDORM*, 9-431
 - unexpected termination, *DIST_TRANS*, 2-15
 - using embedded SQL statements, *SQLRM*, 5-2
 - using module language procedures, *SQLRM*, 4-39
 - using with Callable RDO, *DIST_TRANS*, 5-1
 - using with precompiled SQL, *DIST_TRANS*, 4-3
 - using with RDBPRE, *DIST_TRANS*, 5-1 to 5-18
 - existing application, *DIST_TRANS*, 5-4
 - new application, *DIST_TRANS*, 5-7 to 5-18
 - using with RDML, *DIST_TRANS*, 5-2

- Distributed transaction (Cont.)
 - using with SQL, *DIST_TRANS*, 4-1 to 4-24
 - using with SQL module language, *DIST_TRANS*, 4-2, 4-9 to 4-15
 - using with SQL precompiler, *DIST_TRANS*, 4-15 to 4-22
 - using with VAX DBMS, *DIST_TRANS*, 4-2
- Distributed transaction identifier (TID), *RDMLRM*, 6-131; *DIST_TRANS*, 2-3
 - passing to participants, *DIST_TRANS*, 2-4
- DISTRIBUTED_TRANSACTION clause of START_TRANSACTION statement, *DIST_TRANS*, 5-8, 5-14
- DISTRIBUTED_TRANSACTION keyword of START_TRANSACTION statement, *DIST_TRANS*, 5-8
- /DISTRIBUTED_TRANSACTION qualifier to RDBPRE command line, *DIST_TRANS*, 5-4
- Distributing databases
 - DEFINE SCHEDULE statement, *RDORM*, 9-164
 - DEFINE TRANSFER statement, *RDORM*, 9-180
 - DELETE SCHEDULE statement, *RDORM*, 9-247
 - DELETE TRANSFER statement, *RDORM*, 9-251
 - REINITIALIZE TRANSFER statement, *RDORM*, 9-357
 - SHOW TRANSFER statement, *RDORM*, 9-409
 - START TRANSFER statement, *RDORM*, 9-453
 - STOP TRANSFER statement, *RDORM*, 9-456
 - using RDO statements, *RDORM*, 2-9
- DMU, *DESIGN*, 7-6
- Documentation format for SQL/Services data structures, *SQLSRV*, 10-1
- Dollar sign command (\$) invoking DCL, *RDORM*, 9-81
- Domain, *SQLRM*, 3-18
 - based on data dictionary, *DESIGN*, 3-14
 - changing, *DESIGN*, 5-6; *SQLRM*, 6-2 to 6-10
 - creating, *DESIGN*, 3-14; *SQLRM*, 6-68 to 6-75
 - default value, *DESIGN*, 3-20
 - definition of, *INTRO*, 1-4
 - deleting, *DESIGN*, 5-6; *SQLRM*, 6-301
 - modifying, *DESIGN*, 5-6; *SQLRM*, 6-2 to 6-10
 - naming, *DESIGN*, 3-18; *SQLRM*, 3-18
 - user-defined
 - displaying, *GUSQL*, 1-9
- DOUBLE PRECISION data type, *SQLRM*, 3-36
- DROP CONSTRAINT statement, *SQLRM*, 6-298
- DROP DOMAIN statement, *DESIGN*, 5-6; *SQLRM*, 6-301
- DROP INDEX statement, *SQLRM*, 6-305
- DROP PATHNAME statement, *DESIGN*, 7-27e; *SQLRM*, 6-308
- Dropping
 - See also* Deleting
 - columns, *DESIGN*, 5-7
 - tables, *DESIGN*, 5-7
 - triggers, *DESIGN*, 5-20
- DROP SCHEDULE statement, *SQLRM*, 6-309
- DROP SCHEMA statement, *SQLRM*, 6-311
- DROP STORAGE AREA clause, *DESIGN*, 3-7

DROP STORAGE MAP statement,
SQLRM, 6-313

DROP TABLE statement, *DESIGN*,
 5-7; *SQLRM*, 6-316

DROP TRANSFER statement, *SQLRM*,
 6-320

DROP TRIGGER statement, *SQLRM*,
 6-322

DROP VIEW statement, *SQLRM*,
 6-325

Duplicate node record
 hashed index structure, *MAINT*,
 11-44

Duplicate row
 eliminating, *GUSQL*, 3-16

Duplicate value, *DESIGN*, 3-42
 error, *GUSQL*, 11-28

Dynamically executed SQL statement,
GUSQL, 12-2

Dynamic cursor, *GUSQL*, 4-5, 4-18,
 12-5, 12-16, 12-19

Dynamic DECLARE CURSOR
 statement, *GUSQL*, 4-5, 4-18,
 12-5, 12-15, 12-16, 12-19;
SQLRM, 6-245
See also DECLARE CURSOR
 statement

Dynamic SQL, *GUSQL*, 12-1 to 12-50;
SQLSRV, 2-1 to 2-8; *SQLRM*,
 3-29, D-1
 advantages, *INTRO*, 1-26

DECLARE STATEMENT, *SQLRM*,
 6-266

DESCRIBE statement, *SQLRM*,
 6-287

EXECUTE IMMEDIATE statement,
SQLRM, 6-343

EXECUTE statement, *SQLRM*,
 6-338

INCLUDE statement, *SQLRM*,
 6-407
 parameter markers, *SQLSRV*, 2-6

PREPARE statement, *SQLRM*,
 6-439

Dynamic SQL (Cont.)

purpose of SQLCA, *SQLSRV*, 2-8
See also SQLCA

purpose of SQLDA, *SQLSRV*, 2-7
See also SQLDA

RELEASE statement, *SQLRM*,
 6-452

statement name, *GUSQL*, 12-5

statements, *SQLSRV*, 2-2

statement summary, *SQLRM*, 2-9

summary, *INTRO*, 1-26

supplying parameter at run time,
GUSQL, 12-42

unknown statements, *SQLSRV*, 2-7

use in SQL/Services, *SQLSRV*, 1-2

using, *SQLSRV*, 2-3 to 2-8

using parameter, *GUSQL*, 12-42,
 12-46

using SELECT statements, *SQLSRV*,
 2-6

with distributed transaction,
DIST_TRANS, 4-3

E

Editor

defining for EDIT statement,
GUSQL, 1-24

EDIT statement, *GUSQL*, 1-16;
SQLRM, 6-328
 editing command lines, *RDORM*,
 9-257

EDIT STRING clause, *SQLRM*, 3-52

EDIT_STRING FOR DTR clause,
RDORM, 5-17

EDT editor
 using with EDIT statement, *GUSQL*,
 1-24

Embedded SQL
 Ada-specific requirement, *GUSQL*,
 8-4
 and distributed transaction,
DIST_TRANS, 4-15 to 4-22

Embedded SQL (Cont.)

- COBOL-specific requirement,
GUSQL, 8-5
- C-specific requirement, *GUSQL*, 8-4
- distributed transactions, *SQLRM*,
5-2
- FORTRAN-specific requirement,
GUSQL, 8-6
- Pascal-specific requirement, *GUSQL*,
8-8
- PL/I-specific requirement, *GUSQL*,
8-9
- testing statement interactively,
GUSQL, 6-5
- Embedding DML statements, *GURRR*,
9-2
- Embedding SQL statements in
programs, *SQLRM*, 5-2
- Enabling audit event information,
MAINT, 4-16
- Enabling DACCESS audit events,
MAINT, 4-5, 4-13; *RDORM*, 6-128
- Enabling security alarms, *RDORM*,
6-128
- Enabling security auditing, *RDORM*,
6-127, 6-128
- END DECLARE statement, *SQLRM*,
6-332
- Ending SQL statement
 - in Ada program, *GUSQL*, 8-4
 - in COBOL program, *GUSQL*, 8-5
 - in C program, *GUSQL*, 8-4
 - in FORTRAN program, *GUSQL*, 8-6
 - in interactive SQL, *GUSQL*, 1-13
 - in Pascal program, *GUSQL*, 8-8
 - in PL/I program, *GUSQL*, 8-9
- Ending transactions
 - COMMIT statement, *GUSQL*, 2-46;
RDORM, 9-69; *SQLRM*, 6-60
 - distributed, *DIST_TRANS*, 2-6, 4-10
 - ROLLBACK statement, *GUSQL*,
2-46; *RDORM*, 9-359; *SQLRM*,
6-472

- END_SEGMENTED_STRING statement
 - closing a segmented string, *RDORM*,
9-260
- END_STREAM statement
 - closing an open stream, *RDORM*,
9-261
 - declared, *RDMLRM*, 6-36
 - undeclared, *RDMLRM*, 6-40
- ENQLM parameter
 - values, *MAINT*, 16-138
- ENTER command
 - in CDO, *DESIGN*, 7-17e
- Entering data, *GURRR*, 6-1
- Entering RdbALTER, *MAINT*, 7-3
- Entity-relationship map, *DESIGN*, 2-3
- Environment variable
 - in SQL/Services, *SQLSRV*, B-1
- ERASE statement, *RDMLRM*, 6-42;
GURRR, 6-7
 - deleting records from a database,
RDORM, 9-263
- Erasing data, *GURRR*, 6-7, 9-29
- Error
 - database attachment, *GUSQL*, 11-38
 - database integrity, *GUSQL*, 11-27
 - deadlock, *GUSQL*, 11-25
 - debugging program, *GUSQL*, 9-6
 - in module processing, *GUSQL*, 7-12
 - logging, *GUSQL*, 7-27
 - SQL module processor error,
GUSQL, 7-27
 - SQL precompiler error, *GUSQL*,
8-17
 - message display, *GUSQL*, 11-19
- Error code
 - RDB\$LU_STATUS, *GUSQL*, 11-28
 - SQLCODE, *GUSQL*, 11-28
- Error handling, *GUSQL*, 11-1 to
11-41; *GURRR*, 10-1 to 10-14
- AT END clause, *RDORM*, 9-276
- in BASIC programs, *GURRR*, 13-44
- in COBOL programs, *GURRR*, 14-44
- in FORTRAN programs, *GURRR*,
15-47

- Error handling (Cont.)
 - in SQL, *SQLRM*, A-1, C-1
 - ON ERROR clause, *RDMLRM*, 6-90; *RDORM*, 9-343; *GURRR*, 10-4
 - Rdb/VMS, *RDORM*, B-1
 - run-time, *GURRR*, 10-1
- Error messages
 - in Rdb/VMS
 - explanation files, *RDORM*, B-1
- Errors
 - returning, *SQLSRV*, 9-50
 - returning text of, *SQLSRV*, 9-52
- Evaluating databases
 - performance, *MAINT*, 13-1
- EVE editor
 - editing in interactive SQL, *SQLRM*, 6-328
- Event tracking
 - using DECTrace, *MAINT*, 15-50
- Exact match retrieval performance
 - hashed indexes, *DESIGN*, 4-11
 - optimizing, *DESIGN*, 4-11
- Exception
 - See Bugcheck dump
- Exception condition, *GUSQL*, 11-6
 - handling, *GURRR*, 10-12
 - handling fatal errors, *GURRR*, 10-13
- EXEC SQL flag, *GUSQL*, 8-2; *SQLRM*, 5-2
- Executable image
 - creating, *GUSQL*, 9-1
- Executable SQL statement
 - in SQL module procedure, *GUSQL*, 7-23
- EXECUTE (@) statement, *GUSQL*, 1-22
- Execute (@) statement
 - running RDO command files, *RDORM*, 9-267
- EXECUTE (@) statement, *SQLRM*, 6-335
 - running command files, *SQLRM*, 6-335
- EXECUTE DYNAMIC statement, *SQLRM*, 6-338
- EXECUTE IMMEDIATE statement, *GUSQL*, 12-5; *SQLRM*, 6-343
- EXECUTE statement, *GUSQL*, 12-5
 - in dynamic SQL
 - using parameter, *GUSQL*, 12-42, 12-44
- Execution server process
 - selecting class of, *SQLSRV*, 9-48
- EXISTS operator, *GUSQL*, 3-37, 3-42; *SQLRM*, 3-85
- EXIT command
 - in RdbALTER, *MAINT*, 7-25
- EXIT command (RdbALTER), *RDORM*, 7-28
- Exiting
 - from RDO, *RDORM*, 9-270
 - interactive SQL, *SQLRM*, 6-348
- EXIT statement, *SQLRM*, 6-348
 - exiting from RDO, *RDORM*, 9-270
 - in interactive SQL, *GUSQL*, 1-20
- Exporting databases, *DESIGN*, 5-26; *SQLRM*, 6-349
 - alternatives to, *DESIGN*, 5-26
 - archiving, *MAINT*, 10-69
 - containing no data, *MAINT*, 10-66
 - migration, *MAINT*, 10-70
- EXPORT statement, *DESIGN*, 5-25; *MAINT*, 8-59; *SQLRM*, 6-349
 - backing up a database, *RDORM*, 9-271
 - duplicating database metadata, *DESIGN*, 5-23
 - NO DATA option, *DESIGN*, 5-23
 - using, *DESIGN*, 5-36e
- Expression
 - arithmetic, *RDORM*, 3-21; *SQLRM*, 3-75
 - column select, *GUSQL*, 3-4
 - concatenated, *RDORM*, 3-25
 - conditional, *RDMLRM*, 3-1 to 3-42; *RDORM*, 3-36
 - See Predicate

Expression (Cont.)

- filter, *SQLSRV*, A-1
- segmented string, *RDORM*, 3-34
- select, *GUSQL*, 3-4; *SQLRM*, 3-98
- statistical, *RDORM*, 3-11; *SQLRM*, 3-68
- value, *GUSQL*, 3-5; *RDORM*, 3-2
 - comparing, *GUSQL*, 3-22
- Extended dynamic cursor, *GUSQL*, 4-5, 4-19, 12-5, 12-16, 12-19
- Extended Dynamic DECLARE CURSOR statement, *GUSQL*, 4-5, 12-5, 12-16, 12-19, 12-20; *SQLRM*, 6-249
 - See also DECLARE CURSOR statement
 - using parameter, *GUSQL*, 12-42, 12-47

F

Features of Rdb/VMS

- overview, *INTRO*, 1-6 to 1-10

Fetching

- a page in RdbALTER, *MAINT*, 7-5
- a storage area in RdbALTER, *MAINT*, 7-4

- FETCH statement, *GUSQL*, 4-1, 4-6; *RDMLRM*, 6-49; *GURRR*, 6-8; *SQLRM*, 6-352

- advancing in a stream, *RDORM*, 9-276

- in dynamic SQL, *GUSQL*, 12-6
- in SQL module, *GUSQL*, 7-18
- using parameter, *GUSQL*, 12-46

Field

- extracting data type and size, *RDMLRM*, 6-4

Field attribute, *RDORM*, 5-1

- global, *RDORM*, 5-1
- local, *RDORM*, 5-2

Field definition

- CHANGE FIELD statement, *RDORM*, 9-27
- CHANGE RELATION statement, *RDORM*, 9-48
- DEFINE FIELD statement, *RDORM*, 9-112
- DELETE FIELD statement, *RDORM*, 9-234
- displaying, *RDORM*, 9-384

Field name

- displaying, *RDORM*, 9-384

- File activity statistics, *MAINT*, 15-27, 15-29

File I/O statistics

- by file, *MAINT*, 15-36

File name

- truncation of during RMU/BACKUP command, *RDORM*, 6-33

File qualifiers

- See Parameter qualifiers

Files

- naming, *SQLRM*, 3-7
- naming conventions in VAXcluster configurations, *MAINT*, 18-8
- reducing disk I/O contention, *MAINT*, 16-114
- specifications, *SQLRM*, 3-7

FILLM parameter

- values, *MAINT*, 16-139

Filter expression functions

- in SQL/Services, *SQLSRV*, A-1

Filtering

- of result tables in SQL/Services, *SQLSRV*, 5-2

- FINISH statement, *RDMLRM*, 6-54; *SQLRM*, 6-357

- closing a database, *RDORM*, 9-280
- detaching from database, *GUSQL*, 2-16

- FIRST clause, *RDMLRM*, 4-23

- of record selection expression, *RDORM*, 4-3

FIRST FROM expression, *RDMLRM*, 2-13; *RDORM*, 3-31

FIRST *n* clause, *GURRR*, 3-7

Fixed-point numeric data type, *SQLRM*, 3-35

Flagging extension, *GUSQL*, 7-27, 8-12

Flags

- the O flag, *MAINT*, 17-39
- the R flag, *MAINT*, 17-51
- the SE flags, *MAINT*, 17-27
- the S flag, *MAINT*, 17-22
- the S\ flags, *MAINT*, 17-38
- using to examine access strategies, *MAINT*, 17-20

FLOAT data type, *SQLRM*, 3-36

Floating-point numeric data type, *SQLRM*, 3-36

Foreign key, *MAINT*, 16-16

- and referential integrity, *DESIGN*, 3-38

Formatted binary file

- statistics, *MAINT*, 15-45

Forming record streams, *GURRR*, 3-1

FOR statement, *RDMLRM*, 6-58

- loops, *RDORM*, 9-282
- nested, *GURRR*, 4-9
- segmented strings, *RDMLRM*, 6-66; *RDORM*, 9-287
- transaction handle restriction, *RDORM*, 9-283

FORTRAN

See also Embedded SQL; Program data types, *GURRR*, 8-10t

- declaring host language variables, *GURRR*, 9-2
- declaring parameter in, *GUSQL*, 10-52
- designing program in RDO, *GURRR*, 7-3
- developing an Rdb/VMS program using, *GURRR*, 15-1 to 15-58
- DICTIONARY statement, *GURRR*, 12-10
- precompiled program, *GUSQL*, 8-6

FORTRAN (Cont.)

- supported variable declarations, *SQLRM*, 5-27
- using parameter in, *GUSQL*, 10-52

Fragmentation

- erasing, *MAINT*, 16-94
- modifying, *MAINT*, 16-94
- record, *MAINT*, 16-110
- storage records, *MAINT*, 11-51
- storing, *MAINT*, 16-93

Freeing data and indicator variables

- sqlsrv_free_sqlda_data* routine, *SQLSRV*, 9-30

Free space, *MAINT*, 11-29

- database page, *MAINT*, 11-55
- locked, *MAINT*, 11-29, 11-31, 11-32e, 11-33f
- SPAM pages, *MAINT*, 11-65
- unlocked, *MAINT*, 11-29, 11-31, 11-32e, 11-33f

FROM clause, *GUSQL*, 3-4, 3-9

FROM dictionary clause

- of SQL module language, *GUSQL*, 10-3, 10-10

Function, *GUSQL*, 3-53; *SQLRM*, 3-68

- AVERAGE*, *RDMLRM*, 5-4; *RDORM*, 3-3, 3-11
- AVG*, *GUSQL*, 3-54; *SQLRM*, 3-70
- COUNT*, *GUSQL*, 3-54; *RDMLRM*, 5-8; *RDORM*, 3-3, 3-11; *SQLRM*, 3-69
- global aggregate, *RDORM*, 3-16
- MAX*, *GUSQL*, 3-54; *RDMLRM*, 5-12; *RDORM*, 3-3, 3-11; *SQLRM*, 3-70
- MIN*, *GUSQL*, 3-54; *RDMLRM*, 5-17; *RDORM*, 3-3, 3-11; *SQLRM*, 3-70
- SUM*, *GUSQL*, 3-54; *SQLRM*, 3-69
- TOTAL*, *RDMLRM*, 5-23; *RDORM*, 3-3, 3-11

G

- GBLPAGES parameter
 - values, *MAINT*, 16-136
- GBL program section attribute,
GUSQL, 9-3
- GBLSECTIONS parameter
 - values, *MAINT*, 16-137
- GENERAL language identifier
 - in SQL module, *GUSQL*, 7-11
- Generating sample databases,
DESIGN, 1-10
- GET statement, *RDMLRM*, 6-71
 - retrieving records from a stream,
RDORM, 9-291
- Global aggregate function, *RDORM*,
3-16
- Global attributes, *RDORM*, 5-1
- GRANT statement, *DESIGN*, 6-11
 - ACL style, *SQLRM*, 6-359
 - ANSI/ISO style, *SQLRM*, 6-379
- GROUP BY clause, *GUSQL*, 3-21, 3-55

H

Handle

- database, *RDORM*, 3-4, 9-74, 9-99
 - scope of, *RDORM*, 9-329
- request, *RDORM*, 3-5, 9-82, 9-282,
9-425, 9-458
 - setting scope, *RDORM*, 9-332
- segmented string, *RDORM*, 9-74
- transaction, *RDORM*, 3-5, 9-83,
9-283, 9-426, 9-458
- using in structured programs,
GURRR, 9-37

Handling an error

- ON ERROR, *RDMLRM*, 6-90
- Rdb/VMS, *RDORM*, B-1

Hash bucket record

- hashed index structure, *MAINT*,
11-44

Hashed index, *DESIGN*, 3-41;

- RDORM*, 9-122, 9-127; *SQLRM*,
6-82

Hashed index (Cont.)

See also Index; Sorted index

- ALLOCATION clause, *DESIGN*,
4-12
- and PAGE SIZE clause, *DESIGN*,
4-12
- calculating size
 - potential problems, *MAINT*,
16-73
- changing, *MAINT*, 10-41
- data clustering strategies, *MAINT*,
2-15
- data placement strategies, *MAINT*,
2-15
- define before loading table, *DESIGN*,
4-13
- defining and using
 - important parameters, *MAINT*,
16-56
- designing, *DESIGN*, 4-11
- duplicate node record, *MAINT*, 11-44
- estimating
 - fixed and variable page overhead,
MAINT, 16-59
 - number of SPAM pages, *MAINT*,
16-70e
 - page size, *MAINT*, 16-58, 16-67e
 - data rows, *MAINT*, 16-64
 - hashed index structures,
MAINT, 16-60
 - size of duplicate node record,
MAINT, 16-60
 - size of hash bucket, *MAINT*,
16-60
 - size of system record, *MAINT*,
16-60
 - storage area allocation, *MAINT*,
16-57
 - storage area size, *MAINT*,
16-68e
- estimating size of hashed index
structures, *MAINT*, 16-62e
- hash bucket record, *MAINT*, 11-44
- node records, *MAINT*, 11-41

Hashed index

node records (Cont.)

contents of, *MAINT*, 11-44

page format required for storing,
DESIGN, 4-7

shadow pages, *MAINT*, 16-74

statistics, *MAINT*, 15-20

storing, *DESIGN*, 4-14

structure, *MAINT*, 16-54, 16-55f

SYSTEM record, *MAINT*, 11-44

tuning considerations, *MAINT*,
16-56

HAVING clause, *GUSQL*, 3-55

Help command (RdbALTER), *MAINT*,
7-25

HELP command (RdbALTER), *RDORM*,
7-29

HELP statement, *SQLRM*, 6-389
assistance on RDO topics, *RDORM*,
9-297

Hexadecimal character string literal,
SQLRM, 3-45

High-level language

See Language

Host language program

developing, *GUSQL*, 6-1 to 6-9
from interactive statement,
GUSQL, 6-7

Host language variable, *RDMLRM*,
2-20; *GURRR*, 9-2

See also Parameter

compared with parameter marker,
GUSQL, 12-3, 12-11

declaring in RDML, *GURRR*, 16-2

Host parameter

See Parameter

Host structure

See Parameter

Host variable, *RDORM*, 3-4

I

I/O operations

RDMS\$DEBUG_FLAGS, *MAINT*,
17-51

I/O resource problems

improving performance, *TUNING*,
2-14

resolving, *TUNING*, 2-2

I/O statistics, *MAINT*, 15-13, 15-32

Ikey

high, *MAINT*, 17-51

indexes, *MAINT*, 17-51

low, *MAINT*, 17-51

Importing databases, *SQLRM*, 6-391

See also Restoring

changing device specifications,
MAINT, 10-68

containing no data, *MAINT*, 10-66

from Rdb/ELN databases, *MAINT*,
2-54

using TRACE, *MAINT*, 10-66

IMPORT statement, *DESIGN*, 5-25;

MAINT, 8-59; *SQLRM*, 6-391

duplicating database metadata,
DESIGN, 5-23

NO DATA option, *DESIGN*, 5-23

restoring a database, *RDORM*, 9-299

using, *DESIGN*, 5-36e

INCLUDE SQLCA statement, *GUSQL*,
11-7

See also SQLCA

INCLUDE statement, *GUSQL*, 10-8;
SQLRM, 6-407

Including data dictionary definitions

in RDBPRE programs, *GURRR*, 12-4

in RDML programs, *GURRR*, 16-3

Incomplete distributed transaction,

DIST_TRANS, 1-7, 2-17, 6-1 to
6-7

in corrupt database

resolving, *DIST_TRANS*, 6-5

resolving, *DIST_TRANS*, 6-4

Index, *GURRR*, 2-23

See also Hashed index; Sorted index and performance, *DESIGN*, 3-43
ascending, *RDORM*, 9-122
B-tree, *DESIGN*, 3-40
changes affect storage maps, *MAINT*, 10-43
changing, *MAINT*, 10-38; *RDORM*, 9-36; *SQLRM*, 6-11 to 6-14
changing hashed, *MAINT*, 10-41
changing sorted, *MAINT*, 10-39
chronological key, *MAINT*, 16-42
compressing, *SQLRM*, 6-76
compression, *DESIGN*, 3-46; *MAINT*, 16-35; *RDORM*, 9-119
creating, *DESIGN*, 3-40; *SQLRM*, 6-76 to 6-87
database keys, *MAINT*, 16-40
defining, *RDORM*, 9-119
deleting, *DESIGN*, 5-22; *MAINT*, 10-38, 10-44; *RDORM*, 9-237; *SQLRM*, 6-305
descending, *RDORM*, 9-123
displaying, *GUSQL*, 1-9; *RDORM*, 9-387
distribution among storage areas, *DESIGN*, 4-8
duplicate nodes, *MAINT*, 11-41, 16-38
ensuring unique values in, *DESIGN*, 3-42
for validation, *MAINT*, 16-84
hashed, *DESIGN*, 3-41; *MAINT*, 16-54; *RDORM*, 9-122, 9-127; *SQLRM*, 6-82
ALLOCATION clause, *DESIGN*, 4-12
and page size, *DESIGN*, 4-12
designing, *DESIGN*, 4-11
page format required for storage, *DESIGN*, 4-7
retrieval for exact matches, *DESIGN*, 1-10
high Ikey, *MAINT*, 17-51

Index (Cont.)

loading the database, *MAINT*, 16-129
locking, *MAINT*, 16-38; *GUSQL*, 2-39
low Ikey, *MAINT*, 17-51
modifying, *RDORM*, 9-36; *SQLRM*, 6-11 to 6-14
multisegmented, *MAINT*, 16-54
multi-user access, *MAINT*, 16-49
naming, *SQLRM*, 3-29
node record, *MAINT*, 11-38
See also Index node record
partitioned, *SQLRM*, 6-83
reorganizing databases
definition sequence, *MAINT*, 10-58
retrieval, *MAINT*, 16-34
setting structural characteristics, *MAINT*, 16-50
sorted, *DESIGN*, 3-40; *RDORM*, 9-122; *SQLRM*, 6-79
order, *MAINT*, 17-47
retrieval with data values, *DESIGN*, 1-10
retrieval with dbkey, *DESIGN*, 1-10
specifying storage area for, *DESIGN*, 4-3
statistics
insertion, *MAINT*, 15-17
removal, *MAINT*, 15-19
retrieval, *MAINT*, 15-15
storing hashed, *DESIGN*, 4-14
tradeoffs of using, *DESIGN*, 3-43
updating, *MAINT*, 16-40
Index node record, *MAINT*, 11-38, 11-39e
data region length, *MAINT*, 11-40
fragmentation, *MAINT*, 11-38
level type, *MAINT*, 11-40
storage record type ID, *MAINT*, 11-39

Index node record (Cont.)

- uncompressed owner dbkey, *MAINT*, 11-40
- Indicator array, *SQLRM*, 3-25
 - specifying in SQL module parameter declarations, *SQLRM*, 4-10
- Indicator parameter, *GUSQL*, 10-16 to 10-23; *SQLRM*, 3-21
- IN operator, *GUSQL*, 3-31; *SQLRM*, 3-86
- INSERT statement, *GUSQL*, 5-1 to 5-21; *SQLRM*, 6-414
 - in SQL module, *GUSQL*, 7-19
- INTEGER data type, *DESIGN*, 3-19; *SQLRM*, 3-35
- INTEGRATE DATABASE statement
 - re-creating dictionary definitions, *RDORM*, 9-325
- INTEGRATE statement, *DESIGN*, 7-15, 7-20, 7-22; *SQLRM*, 6-424
- Integrating dictionary and database definitions, *DESIGN*, 7-15
- Integrity
 - database, *MAINT*, 6-4, 16-80
- Interactive control statements
 - summary, *RDORM*, 2-8
- Interactive SQL
 - command procedure, *GUSQL*, 1-21
 - ending statement, *GUSQL*, 1-13
 - exiting, *GUSQL*, 1-4, 1-20
 - invoking with DCL symbol, *GUSQL*, 1-4
 - setting up environment, *GUSQL*, 1-22
 - starting transaction, *GUSQL*, 1-20
 - statements
 - continuing, *GUSQL*, 1-15
 - ending, *GUSQL*, 1-15
 - summary, *INTRO*, 1-20
 - summary of control statements, *SQLRM*, 2-8
 - use in database design, *DESIGN*, 3-3
- Interfaces to Rdb/VMS, *INTRO*, 1-1

Interfaces to Rdb/VMS (Cont.)

- summary, *INTRO*, 1-20 to 1-30
- Internal database structures, *MAINT*, 11-27
- Internationalization, *RDMLRM*, 1-5, 1-6, 3-17, 3-23, 3-34
 - Rdb/VMS support for, *GUSQL*, 1-27; *GURRR*, 1-14
 - relational operators, *GURRR*, 3-9
- Interpreting database statistics, *MAINT*, 15-10
- INVOKE DATABASE statement, *GURRR*, 2-1
 - connecting to a database, *RDORM*, 9-329
 - remote access, *GURRR*, 2-3
 - transaction handle restriction, *RDORM*, 9-332, 9-333
- Invoking
 - databases, *GURRR*, 2-1
 - RdbALTER utility, *RDORM*, 6-4
 - remote databases, *GURRR*, 2-3
- IRPCOUNT parameter
 - values, *MAINT*, 16-134
- IRPCOUNTV parameter
 - values, *MAINT*, 16-134
- IS NULL operator, *GUSQL*, 3-40

J

- Joining relations, *GUSQL*, 3-61; *GURRR*, 4-1 to 4-11
 - See also* Relational join
 - clustering rows by index values, *DESIGN*, 4-16
 - improving performance with index, *DESIGN*, 3-44
- Journal
 - reviewing security audit records, *MAINT*, 4-28
 - security audit, *MAINT*, 4-3, 4-25; *RDORM*, 6-127
- Journaling, *MAINT*, 9-1, 9-35e, 16-79

Journaling (Cont.)

- See also* After-image journal (AIJ);
Recovery-unit journal
- after-image journals, *MAINT*, 9-3, 16-79
- ALTER SCHEMA statement, *MAINT*, 9-4
- backing up after-images to tape, *MAINT*, 9-8
- CHANGE DATABASE statement, *MAINT*, 9-4
- enabling, *MAINT*, 9-3
- file placement, *MAINT*, 9-3, 9-6
- file size, *MAINT*, 9-40
- information not written to AIJ file, *MAINT*, 9-4
- information written to AIJ file, *MAINT*, 9-4
- performance, *MAINT*, 9-4, 9-30
- recovery-unit journals
 - displaying, *RDORM*, 6-68
- RUJ file placement, *MAINT*, 9-30
- setting allocation size, *MAINT*, 9-5
- setting extent size, *MAINT*, 9-5
- steps to creating, *MAINT*, 9-4
- tradeoffs
 - multiple AIJ files versus incremental backup, *MAINT*, 9-8
- truncating AIJ files, *MAINT*, 9-40

K

Key

- foreign, *MAINT*, 16-16
- primary, *MAINT*, 16-16
- using, *DESIGN*, 2-3

Keywords

- for Rdb/VMS statements, *RDORM*, A-1
- required, *SQLRM*, 3-2

L

Labels

- checking on tapes, *RDORM*, 6-34, 6-66, 6-120

Language

- and syntax elements, *SQLRM*, 3-1 to 3-109
- for date format
 - SHOW LANGUAGE statement, *RDORM*, 9-391
- identifier in SQL module processing, *GUSQL*, 7-11
- supported by RDBPRE, *GURRR*, 7-1
- supported by RDML preprocessor, *GURRR*, 7-1
- supported by SQL precompiler, *INTRO*, 1-25

LANGUAGE keyword

- in SQL module, *GUSQL*, 7-9

Language-Sensitive Editor (LSE)

See LSE

- LCL program section attribute, *GUSQL*, 9-3

- LIB\$CONVERT_DATE_STRING, *RDMLRM*, 1-5

- LIB\$DT_FORMAT, *RDMLRM*, 1-6

- LIKE operator, *GUSQL*, 3-23, 3-29; *SQLRM*, 3-89

- LIMIT TO clause, *GUSQL*, 3-15, 4-10

- Line index, *MAINT*, 11-30

- Line terminators, *SQLRM*, 3-2

LINK command (DCL)

- to create
 - executable image, *GUSQL*, 9-2e
 - shareable image, *GUSQL*, 9-4e

Linking

- programs, *GURRR*, 11-12
 - RDML, *GURRR*, 11-12
- RDML, RDBPRE, and SQL modules, *GURRR*, 11-13
- SQL programs, *GUSQL*, 9-1 to 9-6

List, *DESIGN*, 3-19; *GUSQL*, 4-5;
SQLRM, 3-37
 accessing with cursor, *SQLRM*,
 6-234
 changing, *GUSQL*, 5-12
 data type, *INTRO*, 1-5
 inserting, *GUSQL*, 5-11
 performance issues, *DESIGN*, 4-19
 placing in own area, *DESIGN*, 4-8
 segment length, *SQLRM*, 3-37
 storage area, *SQLRM*, 6-134
 storing, *DESIGN*, 4-19

List cursor, *GUSQL*, 4-5, 4-11, 5-11
See also Cursor
 declaring, *SQLSRV*, 9-16

LIST OF BYTE VARYING data type,
SQLRM, 3-37

LIST OF VARBYTE data type
See LIST OF BYTE VARYING data
 type

Literal, *GUSQL*, 3-23; *RDORM*, 3-6;
SQLRM, 3-42
 character string, *SQLRM*, 3-43
 compile-time translation of,
RDORM, 3-8; *SQLRM*, 3-44

DATE literal, *SQLRM*, 3-46
 hexadecimal character string,
SQLRM, 3-45
 in SQL module, *GUSQL*, 7-24
 keywords, *SQLRM*, 3-47
 numeric, *SQLRM*, 3-42
 quoted character string, *SQLRM*,
 3-44
 string
 TODAY, *RDORM*, 3-8
 TOMORROW, *RDORM*, 3-8
 YESTERDAY, *RDORM*, 3-8

Loading

data, *MAINT*, 2-1
 changing database definitions,
MAINT, 2-45
 troubleshooting, *MAINT*, 2-19
 using DATATRIEVE, *MAINT*,
 2-46

Loading

data (Cont.)

using RMU/LOAD command,
MAINT, 2-37

databases, *MAINT*, 16-129;
GUSQL, 5-1
 record order, *MAINT*, 16-129
 to improve performance, *MAINT*,
 16-129
 troubleshooting, *MAINT*, 2-19
 using exclusive write, *MAINT*,
 16-130
 using the PLACE statement,
MAINT, 16-130

from RMS files, *MAINT*, 2-21

relations, *RDORM*, 6-70
 from security audit journal,
MAINT, 4-25; *RDORM*, 6-70

tables, *RDORM*, 6-70
 define hashed index first,
DESIGN, 4-13
 from security audit journal,
MAINT, 4-2, 4-25; *RDORM*,
 6-70

Local attributes, *RDORM*, 5-2

Lock

See Locking

Lock conflict

error, *GUSQL*, 11-25

LOCKIDTBL parameter

values, *MAINT*, 16-135

LOCKIDTBL_MAX parameter

values, *MAINT*, 16-135

Locking, *MAINT*, 16-17 to 16-29;

GUSQL, 2-29

adjustable lock, *MAINT*, 16-35

adjustable lock granularity, *MAINT*,
 16-29

adjustable lock levels, *MAINT*,
 16-29f

auto-locking, *GURRR*, 2-13

compatibility, *MAINT*, 16-24, 16-25t

Locking (Cont.)

- conflict
 - concurrent transactions, *MAINT*, 16-27
 - error
 - with read-only transaction mode, *GUSQL*, 2-31
 - conflict resolution, *GURRR*, 2-17
 - consistency, *MAINT*, 16-17
 - conversion
 - with SNAPSHOT file disabled, *MAINT*, 16-99
 - database areas, *MAINT*, 16-17
 - definition of, *GUSQL*, 2-18
 - demotion, *MAINT*, 16-29
 - duplicate nodes, *MAINT*, 11-41, 16-38
 - handling conflicts in preprocessed programs, *GURRR*, 10-11
 - indexes, *MAINT*, 16-38; *GUSQL*, 2-39
 - information, *MAINT*, 13-6
 - levels, *MAINT*, 16-25
 - lock types, *GURRR*, 2-12
 - multi-user access, *MAINT*, 16-25
 - promotion, *MAINT*, 16-29, 16-47
 - records, *GURRR*, 2-5, 6-4
 - resources, *MAINT*, 16-47
 - rows, *GUSQL*, 2-33
 - sequential retrieval, *MAINT*, 16-47
 - statistics
 - one lock type, *MAINT*, 15-39
 - one statistics field, *MAINT*, 15-44
 - summary, *MAINT*, 15-23
 - tables, *GUSQL*, 2-33
 - with distributed transaction, *DIST_TRANS*, 4-4
- LOG command (RdbALTER), *MAINT*, 7-24; *RDORM*, 7-30
- Log file, *GUSQL*, 1-22
 - for compile-time error, *GUSQL*, 7-27
 - for precompiler error, *GUSQL*, 8-17

Logging

- association, *SQLSRV*, 6-2
- message protocol, *SQLSRV*, 6-4
- program execution, *SQLSRV*, 6-1
- routine, *SQLSRV*, 6-2
- Logical database design, *DESIGN*, 1-4
 - requirements, *DESIGN*, 1-4
 - techniques, *DESIGN*, 2-1
- Logical name, *SQLRM*, G-1
 - CDD\$COMPATIBILITY, *MAINT*, 18-19
 - defining, *MAINT*, 8-3
 - defining RDM\$BUGCHECK_DIR, *MAINT*, 12-3
 - defining RDMS\$BIND_SORT_WORKFILES, *MAINT*, 12-7
 - defining systemwide concealed, *MAINT*, 10-69
 - RDM\$BIND_BUFFERS, *MAINT*, 16-145
 - RDM\$BIND_LOCK_TIMEOUT_INTERVAL, *GUSQL*, 2-30; *DIST_TRANS*, 4-4
 - RDM\$BIND_RUJ_EXTEND_BLKCNT, *MAINT*, 16-146
 - RDM\$BUGCHECK_DIR, *MAINT*, 16-146
 - RDMS\$BIND_LOCK_TIMEOUT_INTERVAL, *MAINT*, 16-149
 - RDMS\$BIND_SEGMENTED_STRING_BUFFER, *MAINT*, 16-148
 - RDMS\$BIND_SORT_WORKFILES, *MAINT*, 16-149
 - RDMS\$BIND_WORK_FILE, *MAINT*, 16-76, 16-147
 - RDMS\$BIND_WORK_VM, *MAINT*, 16-76, 16-147
 - RDMS\$DEBUG_FLAGS, *MAINT*, 16-145
 - RDMS\$DEBUG_FLAGS_OUTPUT, *MAINT*, 16-145
 - RDMS\$DIAG_FLAGS, *MAINT*, 16-143

Logical name (Cont.)

- RDMS\$KEEP_PREP_FILES,
MAINT, 16-144
- RDMS\$RUJ, *MAINT*, 16-144
- recovery-unit journal, *MAINT*, 9-30
- SQL\$DATABASE, *GUSQL*, 1-4
- SQL\$DISABLE_CONTEXT,
DIST_TRANS, 4-3
- SQL\$KEEP_PREP_FILES, *MAINT*,
16-144
- SQLINI, *GUSQL*, 1-22
- SY\$COMMON, *MAINT*, 18-19
- using concealed, *MAINT*, 16-115
- Logical names (Rdb/VMS), *MAINT*,
16-141, 16-141t
- Logical operators, *RDORM*, 3-47
- LONG VARCHAR data type, *SQLRM*,
3-34
- Loops
 - FOR statement, *RDORM*, 9-282
- LRPCOUNT parameter
 - values, *MAINT*, 16-134
- LRPCOUNTV parameter
 - values, *MAINT*, 16-134
- LSE, *GUSQL*, 1-25
 - editing in interactive SQL, *SQLRM*,
6-328
 - using templates, *GUSQL*, 1-25

M

- Main parameter, *SQLRM*, 3-21
- Maintenance
 - activities requiring reload, *MAINT*,
1-8
 - automatic housekeeping activities,
MAINT, 1-6
 - offline DBA activities, *MAINT*, 1-7
 - online DBA activities, *MAINT*, 1-6
- MAKE_CONSISTENT command
(RdbALTER), *MAINT*, 7-22;
RDORM, 7-31
- Mapping relationships, *DESIGN*, 2-3
- MATCHING expression, *RDMLRM*,
3-21

MATCHING expression (Cont.)

- with multinational characters,
RDMLRM, 3-23
- MAXBUF parameter
 - values, *MAINT*, 16-137
- MAX function, *GUSQL*, 3-54;
RDMLRM, 5-12; *RDORM*, 3-3;
SQLRM, 3-70
- MAX statistical function, *RDORM*,
3-15
- Memory
 - specifying default allocation for,
DESIGN, 3-12
- Memory resource problems
 - resolving, *TUNING*, 2-46
- Message file
 - user-defined, *GUSQL*, 11-25
- Message vector, *GURRR*, 10-5
 - in SQLCA, *SQLRM*, C-1
- Metadata, *RDORM*, 8-1
 - creating using SQL, *DESIGN*, 7-20
 - defined, *DESIGN*, 7-1
 - deleting, *DESIGN*, 7-26
 - duplicating, *DESIGN*, 5-23
 - integrating with database, *DESIGN*,
7-15
 - modifying, *DESIGN*, 7-13
 - modifying using CDO, *DESIGN*,
7-14
 - modifying using SQL, *DESIGN*, 7-17
- MF_PERSONNEL sample database,
DESIGN, 1-10
 - creating, *INTRO*, 1-17
 - definitions for, *DESIGN*, A-14
 - physical and logical structure,
MAINT, 14-3f
 - using VAXcluster nodes, *MAINT*,
18-20
- Migration
 - to multifile database, *MAINT*, 8-59
- MIN function, *GUSQL*, 3-54;
RDMLRM, 5-17; *RDORM*, 3-3;
SQLRM, 3-70

Missing value, *DESIGN*, 3-20;
RDMLRM, 2-29; *RDORM*, 3-26,
5-13; *GURRR*, 6-12
See also Null; Default value
MISSING value, *RDMLRM*, 3-27
MISSING_VALUE clause, *RDORM*,
5-13
Mixed page format, *DESIGN*, 4-5, 4-6
Model of design method, *DESIGN*, 1-3
Modifying
access control list, *DESIGN*, 6-11
column definitions, *DESIGN*, 5-7
columns, *SQLRM*, 6-35
constraints, *DESIGN*, 5-17;
RDORM, 9-48; *SQLRM*, 6-35
data, *RDORM*, 9-338; *GURRR*, 6-4;
SQLRM, 6-560
database characteristics, *MAINT*,
10-6 to 10-28
database files, *DESIGN*, 5-25
databases, *DESIGN*, 5-27t;
RDORM, 9-9; *SQLRM*, 6-15
using EXPORT and IMPORT
statements, *MAINT*, 10-52
definitions, *RDORM*, 9-9
with data dictionary, *DESIGN*,
7-13
dictionary definitions using CDO,
DESIGN, 7-14
domains, *DESIGN*, 5-6; *SQLRM*,
6-2 to 6-10
fields, *RDORM*, 9-27, 9-48
indexes, *RDORM*, 9-36; *SQLRM*,
6-11 to 6-14
lists, *GUSQL*, 5-12
memory usage for data access,
DESIGN, 5-25
relations, *RDORM*, 9-48
schemas, *SQLRM*, 6-15 to 6-28
segmented strings, *GURRR*, 9-28
SQL parameters, *SQLRM*, 6-482
storage maps, *RDORM*, 9-61;
SQLRM, 6-29 to 6-34
storage parameters, *DESIGN*, 5-25

Modifying (Cont.)
tables, *DESIGN*, 5-7; *SQLRM*, 6-35
to 6-51
triggers, *DESIGN*, 5-20
MODIFY statement, *RDMLRM*, 6-77;
GURRR, 6-4
changing record values, *RDORM*,
9-338
MODULE keyword
in SQL module, *GUSQL*, 7-9
Module language, *SQLRM*, 2-11, 4-1
to 4-41
See also SQL module language
naming modules, *SQLRM*, 3-30
parameter names, *SQLRM*, 3-30
procedure names, *SQLRM*, 3-30
processor command line, *SQLRM*,
4-35
summary, *INTRO*, 1-25
Module processor
See SQL module processor
Monitor
See also Monitor process
closing log file, *RDORM*, 6-79
multiple processes
in VAXcluster, *MAINT*, 18-15
process
VAXclusters, *MAINT*, 18-9
starting, *RDORM*, 6-80
stopping, *RDORM*, 6-82
Monitor bugs
monitor log file, *MAINT*, 12-3
Monitor log
SYS\$SYSTEM:RDMMON.LOG file,
MAINT, 12-2
Monitor log file
RDMS_MONITOR process
information, *MAINT*, 12-3
reopening, *MAINT*, 3-6
Monitor process, *MAINT*, 3-2
changing base priority, *MAINT*, 3-4
changing priority, *MAINT*, 3-5
changing the log file, *MAINT*, 3-4
in VAXclusters, *MAINT*, 18-9

Monitor process (Cont.)

log file

creating new version, *MAINT*,
3-4

multiple processes in VAXclusters,
MAINT, 18-15

reading monitor log file, *MAINT*, 3-7

starting, *MAINT*, 3-3

stopping, *MAINT*, 3-3

stopping with active users, *MAINT*,
3-4

Monitor utility

MONITOR LOCK, *MAINT*, 16-135

MOVE command (RdbALTER), *MAINT*,
7-19; *RDORM*, 7-33

to move database files, *MAINT*, 7-19

Moving databases, *MAINT*, 7-16

using EXPORT and IMPORT
statements, *DESIGN*, 3-11

using RMU, *DESIGN*, 3-10

Moving storage areas, *RDORM*, 6-85

using RMU/MOVE_AREA, *DESIGN*,
5-26

Moving the root file, *RDORM*, 6-85

Multifile database, *GURRR*, 1-3

as opposed to single-file, *DESIGN*,
3-8

file types, *DESIGN*, 4-2

root file, *DESIGN*, 4-2

sample, *DESIGN*, A-14

specifying storage areas for,
DESIGN, 3-10

storage areas, *DESIGN*, 4-8

storage design, *DESIGN*, 4-1

Multifile databases

as opposed to single-file, *TUNING*,
2-3

Multifile sample database

definitions for, *DESIGN*, A-14

Multiline literal

in program, *GUSQL*, 8-2

in SQL module, *GUSQL*, 7-24

Multiline statement

in precompiled program, *GUSQL*,
8-2

Multinational characters

and the CONTAINING expression,
RDMLRM, 3-17

and the MATCHING expression,
RDMLRM, 3-23

and the STARTING WITH expression,
RDMLRM, 3-34

Multinational Character Set (MCS)

in database object names, *RDMLRM*,
1-5

Multiple databases

using, *GURRR*, 9-4

Multiple storage areas

for segmented strings, *RDORM*,
9-173

Multi-user access

locking, *MAINT*, 16-17

Multi-user conflict, *GUSQL*, 11-25

N

Naming

aliases, *SQLRM*, 3-15

authorization identifiers, *SQLRM*,
3-10

changing table names, *DESIGN*,
5-14; *SQLRM*, 3-12

columns, *SQLRM*, 3-13

constraints, *SQLRM*, 3-30

cursors, *SQLRM*, 3-29

data dictionary path names, *SQLRM*,
3-9

domains, *DESIGN*, 3-18; *SQLRM*,
3-18

files, *SQLRM*, 3-7

indexes, *SQLRM*, 3-29

modules in SQL, *SQLRM*, 3-30

schemas, *SQLRM*, 3-7

storage areas, *SQLRM*, 3-31

storage maps, *SQLRM*, 3-31

tables, *SQLRM*, 3-12

Naming (Cont.)

- triggers, *SQLRM*, 3-19
- views, *SQLRM*, 3-12
- Nested FOR statement, *GURRR*, 4-9
- Networks
 - accessing remote Rdb/VMS database,
MAINT, 16-131
- NOLOG command (RdbALTER),
MAINT, 7-24; *RDORM*, 7-35
- Non-supported statements
 - See Obsolete statements
- Normalization, *DESIGN*, 2-8;
GURRR, 1-4, 4-1
 - effects on performance, *MAINT*,
16-11
- NOSHR program section attribute,
GUSQL, 9-3
- NOT BETWEEN operator, *GUSQL*,
3-41
- NOT CONTAINING operator, *GUSQL*,
3-24
- NOT EQUAL operator, *GUSQL*, 3-30
- Not found condition
 - detecting, *GUSQL*, 11-16
- NOT IN operator, *GUSQL*, 3-34
- NOT LIKE operator, *GUSQL*, 3-23,
3-30, 3-41
- NOT operator, *GUSQL*, 3-28, 3-45
- NPAGEDYN parameter
 - values, *MAINT*, 16-135
- NPAGEVIR parameter
 - values, *MAINT*, 16-136
- Null, *GUSQL*, 3-40
- NULL keyword, *SQLRM*, 3-47
- NULL operator, *SQLRM*, 3-88
- NUMERIC data type, *SQLRM*, 3-35
 - fixed-point, *SQLRM*, 3-35
 - floating-point, *SQLRM*, 3-36
- Numeric literal, *SQLRM*, 3-42

O

- Objects
 - auditing, *MAINT*, 4-13; *RDORM*,
6-129, 6-130
 - auditing for specific privileges,
MAINT, 4-13; *RDORM*, 6-130
- Object types
 - auditing, *MAINT*, 4-13
 - auditing for specific privileges,
MAINT, 4-13
- Obsolete SQL syntax, *SQLRM*, H-1
 - diagnostic messages, *SQLRM*, 5-8
- Obsolete statements, *RDORM*, F-1
 - BACKUP statement, *RDORM*, F-1
 - CLOSE statement, *RDORM*, F-3
 - CONVERT statement, *RDORM*, F-5
 - OPEN statement, *RDORM*, F-6
 - RECOVER statement, *RDORM*, F-7
 - REFRESH MONITOR LOG
statement, *RDORM*, F-9
 - RESTORE statement, *RDORM*, F-10
 - SHOW MONITOR statement,
RDORM, F-16
 - SHOW USERS statement, *RDORM*,
F-17
 - SPOOL statement, *RDORM*, F-18
 - STOP MONITOR statement,
RDORM, F-20
- On-disk structure, *DESIGN*, 1-7
- ON ERROR clause, *RDMLRM*, 6-90;
GURRR, 10-4
 - handling an error, *RDORM*, 9-343
- Online demonstration, *INTRO*, 1-2
- Online help, *SQLRM*, 6-389
 - accessing, *INTRO*, 1-15
- Online program examples for RDBPRE,
RDML, and RDO, *INTRO*, 1-30
- Online program examples for SQL,
INTRO, 1-26
- Opening a database, *MAINT*, 5-2;
RDORM, 6-89; *GURRR*, 2-1
- READY statement, *RDORM*, 9-353

Opening cursors, *GUSQL*, 4-1;
SQLRM, 6-435
OPEN statement, *GUSQL*, 4-1;
SQLRM, 6-435
See also RMU/OPEN command
in dynamic SQL, *GUSQL*, 12-6
in SQL module, *GUSQL*, 7-18
obsolete, *RDORM*, F-6
using parameter, *GUSQL*, 12-46
Operating system parameters, *MAINT*,
13-10
Operator
logical, *GURRR*, 3-10
Optimization access strategy with
 \ debug
 determining, *MAINT*, 17-38
Optimization access strategy with E
 debug
 determining, *MAINT*, 17-27
Optimizer
 See Query optimizer
Optional keywords, *SQLRM*, 3-2
ORDER BY clause, *GUSQL*, 3-19
Ordering row, *GUSQL*, 3-55
OR operator, *GUSQL*, 3-28, 3-49
Outer references, *SQLRM*, 3-17

P

PAGE command (RdbALTER), *RDORM*,
7-36
Page format
 mixed, *DESIGN*, 4-5; *MAINT*,
 16-111
 options, *DESIGN*, 4-5
 uniform, *DESIGN*, 4-5; *MAINT*,
 16-111
Page header, *MAINT*, 11-28
Page size
 estimating, *MAINT*, 16-67e
 considerations, *MAINT*, 16-58
PAGE SIZE clause
 importance for hashed index,
 DESIGN, 4-12

Page tail, *MAINT*, 11-50
Parameter, *SQLRM*, 3-20
 See also Variable
 ALLOCATION, *MAINT*, 16-110
 ASTLM, *MAINT*, 16-140
 AUTOGEN, *MAINT*, 16-132
 BIOLM, *MAINT*, 16-140
 BUFFER SIZE, *MAINT*, 16-89
 BYTLM, *MAINT*, 16-139
 CHANNELCNT, *MAINT*, 16-133
 CLISYMTBL, *MAINT*, 16-133
 database, *MAINT*, 16-86
 database storage area, *MAINT*,
 16-106
 database storage map, *MAINT*,
 16-116
 DEADLOCKWAIT, *MAINT*, 16-138
 declaring
 using FROM clause, *GUSQL*,
 10-10
 declaring in host language, *GUSQL*,
 10-2
 default values
 for database, *RDORM*, D-1t
 for storage areas, *RDORM*, E-1t
 DIOLM, *MAINT*, 16-140
 ENQLM, *MAINT*, 16-138
 FILLM, *MAINT*, 16-139
 for RMU commands, *RDORM*, 6-2
 GBLPAGES, *MAINT*, 16-136
 GBLSECTIONS, *MAINT*, 16-137
 host structure, *SQLRM*, 3-25
 in call to procedure in SQL module,
 GUSQL, 7-12
 indicator, *SQLRM*, 3-21
 in dynamic SQL, *GUSQL*, 12-42,
 12-46
 in module procedure, *SQLRM*, 4-18,
 4-20
 in precompiled program, *GUSQL*,
 10-11
 in SQL module procedure, *GUSQL*,
 7-14, 7-15
 IRPCOUNT, *MAINT*, 16-134

Parameter (Cont.)

IRPCOUNTV, *MAINT*, 16-134
LOCKIDTBL, *MAINT*, 16-135
LOCKIDTBL_MAX, *MAINT*, 16-135
LRPCOUNT, *MAINT*, 16-134
LRPCOUNTV, *MAINT*, 16-134
main, *GUSQL*, 10-11; *SQLRM*,
3-21
 declaring, *GUSQL*, 10-12
 using, *GUSQL*, 10-13
marker, *GUSQL*, 12-3
 in dynamic SQL, *SQLSRV*, 2-6
MAXBUF, *MAINT*, 16-137
maximum values
 for database, *RDORM*, D-1t
 for storage areas, *RDORM*, E-1t
minimum values
 for database, *RDORM*, D-1t
 for storage areas, *RDORM*, E-1t
naming, *SQLRM*, 3-30
NPAGEDYN, *MAINT*, 16-135
NPAGEVIR, *MAINT*, 16-136
NUMBER OF BUFFERS, *MAINT*,
16-91
PAGE SIZE, *MAINT*, 16-109
passing mechanism for, *GUSQL*,
7-12
PGFLQUOTA, *MAINT*, 16-77,
16-141
PRCLM, *MAINT*, 16-141
PROCSECTCNT, *MAINT*, 16-137
qualifiers
 defined, *RDORM*, 6-3
 global use of, *RDORM*, 6-3
 local use of, *RDORM*, 6-3
 positional semantics of, *RDORM*,
6-3
retrieving rows, *GUSQL*, 10-11
SQL precompiler treatment of,
GUSQL, 8-3
SRPCOUNT, *MAINT*, 16-134
SRPCOUNTV, *MAINT*, 16-134
SYSMWCNT, *MAINT*, 16-136

Parameter (Cont.)

tuning buffer size for transaction
type, *MAINT*, 16-92
user account, *MAINT*, 16-138
VIRTUALPAGECNT, *MAINT*,
16-137
WSDEFAULT, *MAINT*, 16-139
WSEXTENT, *MAINT*, 16-139
WSMAX, *MAINT*, 16-137
WSQUOTA, *MAINT*, 16-139
Partitioned index, *SQLRM*, 6-83
Partitioning
 data, *RDORM*, 9-171
 horizontal, *DESIGN*, 4-17
Pascal
 See also Embedded SQL; Program
 data types, *GURRR*, 8-11t
 data types generated by RDML,
RDMLRM, A-2
 declaring
 function, *RDMLRM*, 6-4
 designing program in RDO, *GURRR*,
7-3
 developing an Rdb/VMS program in,
GURRR, 18-1 to 18-52
 parameter in, *GUSQL*, 10-62
 precompiled program, *GUSQL*, 8-8
 supported variable declarations in
 precompiled SQL, *SQLRM*, 5-30
 using parameter in, *GUSQL*, 10-62
Passing a database value
 data type conversion, *GURRR*, 8-4
Patching database corruption, *MAINT*,
7-1
Patching databases
 See RMU/ALTER command
PATHNAME clause
 use in database design, *DESIGN*,
3-4
Path names, *SQLRM*, 3-9
dictionary
 displaying default directory,
RDORM, 9-383

Performance

- adjusting storage area parameters, *MAINT*, 16-106
- adjusting storage map parameters
 PLACEMENT VIA INDEX option, *MAINT*, 16-117
- after-image journaling strategy, *MAINT*, 16-79
- AIJ file extents, *MAINT*, 16-97
- allocation for AIJ file, *MAINT*, 16-95
- allocation for snapshot file, *MAINT*, 16-96
- analyzing the database, *MAINT*, 14-1
- changing operating system parameters, *MAINT*, 13-10
- CPU resource problems, *TUNING*, 2-48
- data compression
 - considerations, *MAINT*, 16-128
 - enabling and disabling, *MAINT*, 16-118
- degradation
 - caused by index, *DESIGN*, 3-43
 - caused by overusing resources, *TUNING*, 1-10
- degree of normalization, *MAINT*, 16-11
- developing change database procedure, *MAINT*, 13-10
- disabling snapshot file, *MAINT*, 16-98
- enhancing application, *SQLSRV*, 5-1
- evaluating, *MAINT*, 13-4
 - utilities and tools, *MAINT*, 13-5
 - VAX Software Performance Monitor (SPM), *MAINT*, 13-7
- evaluating databases, *MAINT*, 13-1
 - operating system utilities, *MAINT*, 13-7
 - sample procedure, *MAINT*, 13-11
 - VAXcluster environment, *MAINT*, 13-9

Performance (Cont.)

- evaluating problem areas, *MAINT*, 13-2t
- evaluating space usage, *MAINT*, 13-5
- evaluating test database, *MAINT*, 13-9
- export/import procedure, *MAINT*, 13-10
- fragmentation, *MAINT*, 16-110
- I/O resource problems, *TUNING*, 2-2
 - balancing I/O load, *TUNING*, 2-5
 - detecting I/O resource bottlenecks, *TUNING*, 2-2
 - reducing I/O, *TUNING*, 2-14
- improving, *GUSQL*, 1-23, 2-24; *TUNING*, 1-10
 - access to data
 - data distribution, *MAINT*, 16-77
 - by application tuning, *TUNING*, 1-12
 - by creating index after loading database, *GUSQL*, 5-2
 - by creating redundant table, *GUSQL*, 5-9
 - by database tuning, *TUNING*, 1-11
 - by evaluating constraint at commit time, *GUSQL*, 5-1
 - by system tuning, *TUNING*, 1-10
 - by understanding your data, *MAINT*, 16-4
 - by using temporary table, *GUSQL*, 5-10
- more buffers reduce RUJ file I/O operations, *MAINT*, 16-92
- SQL module processor, *GUSQL*, 7-30
- SQL precompiler, *GUSQL*, 8-18
 - with hashed index, *DESIGN*, 3-41
 - with index, *DESIGN*, 3-44

Performance (Cont.)

- interrelated database parameters, *MAINT*, 16-9t
- issues, *MAINT*, 16-1
- journaling, *MAINT*, 9-30
- locking information, *MAINT*, 13-6
- making changes in order of difficulty, *MAINT*, 13-13
- memory resource problems, *TUNING*, 2-46
- modifying the test database, *MAINT*, 13-9
- monitoring tools
 - Monitor utility, *MAINT*, 16-132; *TUNING*, 1-6
 - RMU/SHOW/STATISTICS
 - command, *MAINT*, 16-86
 - Software Performance Monitor (SPM), *MAINT*, 16-132; *TUNING*, 1-6
 - VAX Performance Advisor (VPA), *MAINT*, 16-132; *TUNING*, 1-6
- normalization, *MAINT*, 16-11
- number of recovery buffers, *MAINT*, 16-94
- optimization
 - in retrieving data, *DESIGN*, 4-11
 - using exact match retrieval, *DESIGN*, 4-11
- overview, *MAINT*, 1-14
- page format, *MAINT*, 16-111
- performance-related database changes, *MAINT*, 13-11
- problems
 - memory management, *MAINT*, 13-2
 - process parameters, *MAINT*, 13-2
 - system resources, *MAINT*, 13-2
- range retrieval, *DESIGN*, 4-14
- reducing disk I/O contention, *MAINT*, 16-114

Performance (Cont.)

- reducing disk I/O operations, *MAINT*, 16-76
- selecting threshold values for SPAMs, *MAINT*, 16-111
- sizing SPAM intervals, *MAINT*, 16-113
- snapshot file extents, *MAINT*, 16-97
- statistics, *MAINT*, 13-6, 15-1
- transaction type
 - recommended buffer settings, *MAINT*, 16-92t
- tuning working set parameters, *MAINT*, 13-3
- using deferred snapshots, *MAINT*, 16-103
- using lists, *DESIGN*, 4-19
- using RDM\$BIND_BUFFERS
 - Rdb/VMS logical name, *MAINT*, 16-145
- using RDM\$BIND_RUJ_EXTEND_BLKCNT Rdb/VMS logical name, *MAINT*, 16-146
- using RDM\$BUGCHECK_DIR
 - Rdb/VMS logical name, *MAINT*, 16-146
- using RDMS\$BIND_LOCK_TIMEOUT_INTERVAL Rdb/VMS logical name, *MAINT*, 16-149
- using RDMS\$BIND_SEGMENTED_STRING_BUFFER Rdb/VMS logical name, *MAINT*, 16-148
- using RDMS\$BIND_SORT_WORKFILES Rdb/VMS logical name, *MAINT*, 16-149
- using RDMS\$BIND_WORK_FILE Rdb/VMS logical name, *MAINT*, 16-147
- using RDMS\$BIND_WORK_VM Rdb/VMS logical name, *MAINT*, 16-147
- using RDMS\$DEBUG_FLAGS
 - Rdb/VMS logical name, *MAINT*, 16-145

Performance (Cont.)

- using RDMS\$DEBUG_FLAGS_ OUTPUT Rdb/VMS logical name, *MAINT*, 16-145
- using RDMS\$DIAG_FLAGS Rdb/VMS logical name, *MAINT*, 16-143
- using RDMS\$KEEP_PREP_FILES Rdb/VMS logical name, *MAINT*, 16-144
- using RDMS\$RUJ Rdb/VMS logical name, *MAINT*, 16-144
- using SORTWORKn Rdb/VMS logical name, *MAINT*, 16-149
- using SQL\$KEEP_PREP_FILES Rdb/VMS logical name, *MAINT*, 16-144
- using the query optimizer, *MAINT*, 17-1; *GURRR*, 2-28
- VAX Performance Advisor (VPA), *MAINT*, 13-7
- VAX Software Performance Monitor (SPM), *MAINT*, 13-7
- with distributed transaction, *DIST_TRANS*, 3-2, 3-7
- PERSONNEL sample database, *DESIGN*, 1-10
 - creating, *INTRO*, 1-15
 - definitions for, *DESIGN*, A-1
- PGFLQUOTA parameter values, *MAINT*, 16-141
- Physical database design, *DESIGN*, 1-4, 1-6
 - creating, *DESIGN*, 4-1
 - implementing with minimal effort, *MAINT*, 16-2t
 - making a prototype, *DESIGN*, 4-1
 - using RdbExpert, *DESIGN*, 1-6
- Pieces tracking
 - defined, *DESIGN*, 7-1
- PL/I
 - See also* Embedded SQL; Program declaring parameter in, *GUSQL*, 10-71
 - precompiled program, *GUSQL*, 8-9

PL/I (Cont.)

- supported variable declarations in precompiled SQL, *SQLRM*, 5-36
- using parameter in, *GUSQL*, 10-71
- PLACE statement
 - defined, *RDORM*, 9-346
- Positional qualifiers
 - See* Parameter qualifiers
- PRCLM parameter
 - values, *MAINT*, 16-141
- Precompiled SQL
 - and distributed transaction, *DIST_TRANS*, 4-3, 4-15 to 4-22
 - embedding statements, *SQLRM*, 5-2
 - summary, *INTRO*, 1-25
- Precompiler, *SQLRM*, 5-1 to 5-38
 - See also* SQL precompiler command line, *SQLRM*, 5-4
 - statements, *SQLRM*, 2-10
 - supported Ada variable declarations, *SQLRM*, 5-11
 - supported COBOL variable declarations, *SQLRM*, 5-24
 - supported C variable declarations, *SQLRM*, 5-18
 - supported FORTRAN variable declarations, *SQLRM*, 5-27
 - supported host language variable declarations, *SQLRM*, 5-10
 - supported Pascal variable declarations, *SQLRM*, 5-30
 - supported PL/I variable declarations, *SQLRM*, 5-36
- Predicate, *GUSQL*, 3-22; *SQLRM*, 3-77
 - See also* Boolean operator; Conditional Expression; Conditional operator; Relational operator
- BASIC, *SQLRM*, 3-81
- BETWEEN, *SQLRM*, 3-81
- complex, *SQLRM*, 3-82
- CONTAINING, *SQLRM*, 3-84
- EXISTS, *SQLRM*, 3-85
- IN, *SQLRM*, 3-86

Predicate (Cont.)

- LIKE, *SQLRM*, 3-89
- NULL, *SQLRM*, 3-88
- quantified, *SQLRM*, 3-94
- STARTING WITH, *SQLRM*, 3-96
- UNIQUE, *SQLRM*, 3-97

Prepare phase

- of distributed transaction, *DIST_TRANS*, 1-6, 2-7

PREPARE statement, *GUSQL*, 12-5;

- RDMLRM*, 6-98; *SQLRM*, 6-439
- using parameter, *GUSQL*, 12-42, 12-44, 12-46

Preprocessing

- error handling, *GURRR*, 10-3
- linking, *GURRR*, 11-12
- RDML*, *GURRR*, 11-12
- preparing programs, *GURRR*, 11-1
- RDML*, *GURRR*, 11-6 to 11-11

Preprocessor, *RDMLRM*, 1-7

- C, *GURRR*, 17-2
- RDBPRE, *GURRR*, 12-1
- RDBPRE BASIC, *GURRR*, 13-2
- RDBPRE COBOL, *GURRR*, 14-2
- RDBPRE FORTRAN, *GURRR*, 15-2
- RDML*, *GURRR*, 16-1, 18-2

Primary key, *MAINT*, 16-16

- and referential integrity, *DESIGN*, 3-38

PRINT statement, *SQLRM*, 6-447

- retrieving records from a stream, *RDORM*, 9-349

Privilege

- ACL-style, *DESIGN*, 6-2; *SQLRM*, 6-359, 6-455
- adding, *SQLRM*, 6-359
- advantages of ACL-style protection, *DESIGN*, 6-6
- advantages of ANSI/ISO-style protection, *DESIGN*, 6-7
- ANSI/ISO style, *DESIGN*, 6-3; *SQLRM*, 6-379, 6-465
- building ACLs, *DESIGN*, 6-12

Privilege (Cont.)

- choosing between ANSI/ISO and ACL style, *DESIGN*, 6-5
 - creating ACLs, *DESIGN*, 6-9
 - creating identification codes, *DESIGN*, 6-10
 - declaring schema to specify, *DESIGN*, 6-7
 - default access to schema, *DESIGN*, 6-3
 - defining, *SQLRM*, 6-359, 6-379
 - denying, *DESIGN*, 6-12
 - differences between ANSI/ISO and ACL style, *DESIGN*, 6-5
 - displaying, *RDORM*, 9-392, 9-395
 - granting, *SQLRM*, 6-359, 6-379
 - needed to create ACLs, *DESIGN*, 6-8
 - overriding, *DESIGN*, 6-4, 6-35; *SQLRM*, 6-370
 - required for database operations, *DESIGN*, 6-26
 - required for distributed transaction, *DIST_TRANS*, 3-6
 - revoking, *SQLRM*, 6-455
 - role-oriented, *SQLRM*, 6-370
 - specifying, *DESIGN*, 6-25
 - specifying in ACEs, *DESIGN*, 6-25
 - timing of changes taking effect, *DESIGN*, 6-12
- ## Procedure
- in SQL module, *GUSQL*, 7-13, 7-14, 7-21, 7-23
 - naming, *SQLRM*, 3-30
- ## PROCEDURE keyword, *GUSQL*, 7-13
- ## Processing
- Callable RDO programs, *GURRR*, 11-12
 - SQL module, *GUSQL*, 7-25
- ## Process pooling
- components of, *SQLSRV*, 7-3f
 - component summary, *SQLSRV*, 7-4t
 - SQL/Services, *SQLSRV*, 7-1 to 7-17
- ## PROSECTCNT parameter
- values, *MAINT*, 16-137

Product kits, *INTRO*, 1-33

Program

- building SQL/Services applications,
SQLSRV, 2-13 to 2-15
 - on Macintosh, *SQLSRV*, 2-15
 - on MS-DOS, *SQLSRV*, 2-13
 - on OS/2, *SQLSRV*, 2-14
 - on ULTRIX, *SQLSRV*, 2-14
 - on ULTRIX for RISC, *SQLSRV*,
2-14
 - on VMS, *SQLSRV*, 2-13
- creating executable image, *GUSQL*,
9-1 to 9-7
- debugging, *GUSQL*, 6-5, 9-6;
GURRR, 11-24
- designing
 - BASIC, *GURRR*, 7-3
 - C, *GURRR*, 7-3
 - Callable RDO, *GURRR*, 7-3
 - COBOL, *GURRR*, 7-3
 - FORTRAN, *GURRR*, 7-3
 - Pascal, *GURRR*, 7-3
- developing, *GUSQL*, 6-1 to 6-9
 - from interactive statement,
GUSQL, 6-7
 - with RDBPRE, *GURRR*, 12-1 to
12-11
- embedding DML statements,
GURRR, 9-2
- guideline for developing, *GUSQL*,
6-2
- interface
 - Callable RDO, *GURRR*, 19-2
 - C preprocessor, *GURRR*, 17-2
 - RDBPRE BASIC preprocessor,
GURRR, 13-2
 - RDBPRE COBOL preprocessor,
GURRR, 14-2
 - RDBPRE FORTRAN preprocessor,
GURRR, 15-2
 - RDBPRE preprocessor, *GURRR*,
12-1
 - RDML preprocessor, *GURRR*,
16-1, 18-2

Program (Cont.)

- languages supported by SQL
precompiler, *INTRO*, 1-25
- linking, *GURRR*, 11-12
- main parameter in, *GUSQL*, 10-11
- modifying records, *GURRR*, 9-27
- parameter declaration
 - for Ada, *GUSQL*, 10-26
 - for C, *GUSQL*, 10-33
 - for COBOL, *GUSQL*, 10-43
 - for FORTRAN, *GUSQL*, 10-52
 - for Pascal, *GUSQL*, 10-62
 - for PL/I, *GUSQL*, 10-71
 - SQL module, *GUSQL*, 10-79
- precompiled, *GUSQL*, 8-2
- processing, *GUSQL*, 8-10
- processing SQL module used by,
GUSQL, 7-25
- prototyping queries with interactive
interface, *GURRR*, 7-3
- running, *GUSQL*, 8-10, 9-6
- running SQL/Services sample
application, *SQLSRV*, 4-4
- run-time error, *GUSQL*, 11-2
- SQL/Services sample application,
SQLSRV, 4-1
 - on Macintosh, *SQLSRV*, 4-3
 - on MS-DOS, *SQLSRV*, 4-2
 - on OS/2, *SQLSRV*, 4-3
 - on ULTRIX, *SQLSRV*, 4-3
 - on ULTRIX for RISC, *SQLSRV*,
4-3
 - on VMS, *SQLSRV*, 4-2
- SQLSRV\$DYNAMIC*
source code listings, *SQLSRV*,
C-1
- structure, *GURRR*, 9-1
- structured programming in
preprocessed programs, *GURRR*,
9-35
- testing with command procedure,
GUSQL, 6-5
- using Callable RDO, *GURRR*, 9-46
- using cursor, *GUSQL*, 4-13

Program (Cont.)

- using Rdb/VMS, *GURRR*, 1–11
- using transactions, *GURRR*, 9–33
- using VMS Debugger, *GUSQL*, 9–6

Protection

See also Access rights; Privilege changing using CHANGE

- PROTECTION statement, *RDORM*, 9–43
 - column level, *DESIGN*, 6–13, 6–24
 - default, *DESIGN*, 6–3; *SQLRM*, 6–359
 - defining
 - for columns, *DESIGN*, 6–24
 - for tables, *DESIGN*, 6–22
 - for views, *DESIGN*, 6–22
 - using CREATE SCHEMA statement, *DESIGN*, 3–8
 - using DEFINE PROTECTION statement, *RDORM*, 9–132
 - defining using CHANGE PROTECTION statement, *RDORM*, 9–43
 - deleting
 - using DELETE PROTECTION statement, *RDORM*, 9–241
 - for shareable Rdb/VMS definitions in CDD/Plus, *DESIGN*, 6–5
 - overriding, *DESIGN*, 6–4, 6–35
 - schema level, *DESIGN*, 6–13
 - table level, *DESIGN*, 6–13, 6–22
 - view level, *DESIGN*, 6–22
- ## PROTECTION audit events
- enabling or disabling, *MAINT*, 4–9
- ## Protection definition
- CHANGE PROTECTION statement, *RDORM*, 9–43
- ## Prototype
- physical database design, *DESIGN*, 4–1
- ## Prototype transaction, *DESIGN*, 2–12
- ## Prototyping queries, *GURRR*, 7–3

Proxy-like access

- SQL/Services, *SQLSRV*, 8–3
- PSECT names, *RDMLRM*, 1–5

Q

- ## QUADWORD data type, *SQLRM*, 3–35
- ## Qualifiers for RMU commands, *RDORM*, 6–2
- ## Quantified predicate, *SQLRM*, 3–94
- ## Query
- multisegmented key
 - avoid OR condition, *MAINT*, 17–3
- ## Query header, *SQLRM*, 3–51
- ## Query optimizer, *MAINT*, 17–1, 17–1 to 17–5; *GURRR*, 2–28
- access strategies, *MAINT*, 17–2
 - analyzing the RDMS\$DEBUG_FLAGS display, *MAINT*, 17–46
- ## assisting it, *MAINT*, 17–4
- ## determining
- access strategy with SE flags, *MAINT*, 17–27
 - access strategy with S\ flags, *MAINT*, 17–38
 - optimization cost with O flag, *MAINT*, 17–39
 - strategy with S flag, *MAINT*, 17–22
- ## dynamic leaf-level optimization, *MAINT*, 17–5, 17–9
- background-only leaf, *MAINT*, 17–12
 - examples, *MAINT*, 17–10
 - fast-first leaf, *MAINT*, 17–14
 - four-leaf types, *MAINT*, 17–11
 - index-only leaf, *MAINT*, 17–19
 - sorted-leaf, *MAINT*, 17–16
- ## dynamic OR optimization, *MAINT*, 17–5, 17–6
- new notation, *MAINT*, 17–7
- ## dynamic versus traditional OR optimization, *MAINT*, 17–5

Query optimizer (Cont.)
 join predicate, *MAINT*, 17-4
 optimization cost
 determining, *MAINT*, 17-39
 strategy, *MAINT*, 16-32
 tasks, *MAINT*, 17-4
 traditional OR optimization, *MAINT*,
 17-6
 using both S and O flags, *MAINT*,
 17-42
 using *RDMS\$DEBUG_FLAGS*,
MAINT, 17-20
 using *RDMS\$DEBUG_FLAGS_*
OUTPUT, *MAINT*, 17-20
QUERY_HEADER FOR DTR clause,
RDORM, 5-17
QUERY_NAME FOR DTR clause,
RDORM, 5-17
QUIT statement, *SQLRM*, 6-449
 Quitting interactive SQL, *SQLRM*,
 6-449
 Quoted character string literal,
SQLRM, 3-44

R

RADIX command (*RdbALTER*),
MAINT, 7-13; *RDORM*, 7-37
RADIX in *RdbALTER*, *MAINT*, 7-23
 Range retrieval, *GUSQL*, 3-35
 optimizing performance, *DESIGN*,
 4-14
RBR file, *DESIGN*, 5-23
RDA (storage area) file, *DESIGN*, 4-2
RDB\$CONSTRAINTS system relation,
RDORM, 8-4
RDB\$CONSTRAINT_RELATIONS
 system relation, *RDORM*, 8-5
RDB\$DATABASE system relation,
RDORM, 8-6
RDB\$DBHANDLE authorization
 identifier, *GUSQL*, 7-12
RDB\$DB_KEY expression, *RDMLRM*,
 2-26; *RDORM*, 3-32

RDB\$FIELDS system relation,
RDORM, 8-11
RDB\$FIELD_VERSIONS system
 relation, *RDORM*, 8-9
RDB\$INDEX_SEGMENTS system
 relation, *RDORM*, 8-15
RDB\$INDICES system relations,
RDORM, 8-16
RDB\$INTERPRET, *GURRR*, 7-2
RDB\$LU_STATUS field, *GUSQL*, 11-11
RDB\$MESSAGE_VECTOR, *GUSQL*,
 11-10; *GURRR*, 10-5
RDB\$MISSING expression, *RDMLRM*,
 2-29
RDB\$RELATIONS system relation,
RDORM, 8-20
RDB\$RELATION_FIELDS system
 relation, *RDORM*, 8-17
RDB\$SYSTEM
 read-only, *MAINT*, 16-31
 storage area, *DESIGN*, 4-3
RDB\$VIEW_RELATIONS system
 relation, *RDORM*, 8-23
RDB (database root) file, *DESIGN*, 4-2
RdbALTER, *MAINT*, 7-1
 altering data fields, *RDORM*, 7-8
AREA command, *MAINT*, 7-4
ATTACH command, *MAINT*, 7-3
 attaching to a database, *MAINT*,
 7-3; *RDORM*, 7-5
 changing area and snapshot file
 specifications, *RDORM*, 7-14
 changing page contents, *MAINT*,
 7-13
COMMIT command, *MAINT*, 7-16,
 7-25
 committing changes, *RDORM*, 7-7
 completing transactions, *MAINT*,
 7-24
DEPOSIT command, *MAINT*, 7-13,
 7-17
DETACH command, *MAINT*, 7-3,
 7-25

RdbALTER (Cont.)

- detaching from a database, *MAINT*, 7-3; *RDORM*, 7-17
- DISPLAY command, *MAINT*, 7-11
 - *, *MAINT*, 7-11
 - AREA_NUMBER, *MAINT*, 7-11
 - CHECKSUM, *MAINT*, 7-11
 - COUNT, *MAINT*, 7-11
 - DATA, *MAINT*, 7-12
 - FREE_SPACE, *MAINT*, 7-11
 - HEADER, *MAINT*, 7-11
 - INDEX, *MAINT*, 7-12
 - LINE, *MAINT*, 7-12
 - LOCKED_FREE_SPACE, *MAINT*, 7-11
 - PAGE_NUMBER, *MAINT*, 7-11
 - SPACE, *MAINT*, 7-12
 - TIME_STAMP, *MAINT*, 7-11
- displaying
 - area or snapshot file
 - specifications, *RDORM*, 7-25
 - data fields, *RDORM*, 7-18
 - page contents, *MAINT*, 7-11
 - root file specifications, *RDORM*, 7-27
- ending the session, *RDORM*, 7-28
- entering, *MAINT*, 7-3
- EXIT command, *MAINT*, 7-25
- exiting, *MAINT*, 7-25
- fetching a page, *MAINT*, 7-5; *RDORM*, 7-36
- fetching a storage area, *MAINT*, 7-4
- getting information on, *RDORM*, 7-29
- HELP command, *MAINT*, 7-25
- information needed before using, *MAINT*, 7-2
- invoking, *RDORM*, 6-4
- keeping an audit trail, *MAINT*, 7-24; *RDORM*, 7-30
- LOG command, *MAINT*, 7-2, 7-24
- logging a session, *MAINT*, 7-2, 7-24
- MAKE_CONSISTENT command, *MAINT*, 7-22

RdbALTER (Cont.)

- MOVE command, *MAINT*, 7-19
- moving data, *MAINT*, 7-19; *RDORM*, 7-33
- moving database files, *MAINT*, 7-16, 7-17
- NOLOG command, *MAINT*, 7-24
- PAGE command, *MAINT*, 7-5
- RADIX command, *MAINT*, 7-13, 7-23
- resetting an inconsistent flag, *RDORM*, 7-31
- resetting the corruption flag, *RDORM*, 7-39
- ROLLBACK command, *MAINT*, 7-25
- setting default radix, *RDORM*, 7-37
- specifying an area, *MAINT*, 7-4; *RDORM*, 7-3
- specifying a page, *MAINT*, 7-4; *RDORM*, 7-3
- stopping a session log, *MAINT*, 7-24; *RDORM*, 7-35
- UNCORRUPT command, *MAINT*, 7-3
- undoing changes, *MAINT*, 7-2; *RDORM*, 7-38
- using with RMU/VERIFY, *MAINT*, 7-2
- VERIFY command, *MAINT*, 7-23
- verifying alterations, *MAINT*, 7-23
- verifying a page, *RDORM*, 7-41
- VERIFY versus RMU/VERIFY, *MAINT*, 7-24
- when to use, *MAINT*, 7-1

RdbExpert

- using for database design, *DESIGN*, 1-6

RDBPRE command line qualifier

- /DISTRIBUTED_TRANSACTION, *DIST_TRANS*, 5-4

RDBPRE preprocessor

- creating a shareable image, *GURRR*, 11-14
- creating output files, *GURRR*, 11-4

RDBPRE preprocessor (Cont.)

- developing applications, *GURRR*, 12-1 to 12-11
- DML differences from RDO, *GURRR*, 12-2
- error message output file, *GURRR*, 11-6
- invoking, *GURRR*, 11-2
- supported languages, *GURRR*, 7-1
- using, *GURRR*, 11-1
- using with distributed transaction, *DIST_TRANS*, 5-1 to 5-18
 - existing application, *DIST_TRANS*, 5-4
 - new application, *DIST_TRANS*, 5-7 to 5-18
- using with host language compile qualifiers, *GURRR*, 11-3

RDBVMS\$COLLATIONS system relation, *RDORM*, 8-24

RDBVMS\$INTERRELATIONS system relation, *RDORM*, 8-25

RDBVMS\$PRIVILEGES system relation, *RDORM*, 8-26

RDBVMS\$RELATION_CONSTRAINTS system relation, *RDORM*, 8-27

RDBVMS\$RELATION_CONSTRAINT_FLDS system relation, *RDORM*, 8-31

RDBVMS\$STORAGE_MAPS system relation, *RDORM*, 8-31

RDBVMS\$STORAGE_MAP_AREAS system relation, *RDORM*, 8-32

RDBVMS\$TRIGGERS system relation, *RDORM*, 8-33

Rdb/VMS error messages

- explanation files, *RDORM*, B-1

Rdb/VMS Management Utility (RMU)
See *RMU*, *RDORM*, 6-1

RDM\$BIND_BUFFERS Rdb/VMS logical name, *MAINT*, 16-145

RDM\$BIND_LOCK_TIMEOUT_INTERVAL logical name, *GUSQL*, 2-30; *DIST_TRANS*, 4-4

RDM\$BIND_RUJ_EXTEND_BLKCNT Rdb/VMS logical name, *MAINT*, 16-146

RDM\$BUGCHECK_DIR logical name, *MAINT*, 16-146

RDM\$BUGCHECK_DIR Rdb/VMS logical name

- defining, *MAINT*, 12-3

RDML

- and Rdb/ELN, *RDMLRM*, 1-2
- and Rdb/VMS, *RDMLRM*, 1-2
- conditional expression, *RDMLRM*, 1-2
- element
 - clause and statement, *RDMLRM*, 1-2
- keyword list, *RDMLRM*, 1-3
- language, *RDMLRM*, 1-1
- language element, *RDMLRM*, 1-1
- linking programs, *GURRR*, 11-12
- naming convention, *RDMLRM*, 1-3
- program development in, *GURRR*, 16-1
- record selection expression, *RDMLRM*, 1-2
- statements
 - in C programs, *GURRR*, 17-1
 - in Pascal programs, *GURRR*, 18-1
- statistical function, *RDMLRM*, 1-2
- syntax differences from RDO, *GURRR*, 16-2
- using the preprocessor, *GURRR*, 11-6 to 11-11
- using with distributed transaction, *DIST_TRANS*, 5-2
- value expression, *RDMLRM*, 1-1

RDML-generated data types

- for VAX C, *RDMLRM*, A-1
- for VAXELN Pascal, *RDMLRM*, A-3
- for VAX Pascal, *RDMLRM*, A-2

RDML preprocessor

- supported languages, *GURRR*, 7-1

RDMS\$BIND_LOCK_TIMEOUT_
 INTERVAL Rdb/VMS logical name,
MAINT, 16-149

RDMS\$BIND_SEGMENTED_STRING_
 BUFFER Rdb/VMS logical name,
MAINT, 16-148

RDMS\$BIND_SORT_WORKFILES
 Rdb/VMS logical name, *MAINT*,
 16-149

RDMS\$BIND_WORK_FILE Rdb/VMS
 logical name, *MAINT*, 16-147

RDMS\$BIND_WORK_VM Rdb/VMS
 logical name, *MAINT*, 16-147

RDMS\$DEBUG_FLAGS, *MAINT*,
 17-20 to 17-54
 analyzing display, *MAINT*, 17-46
 conjunct, *MAINT*, 17-47
 displaying access strategies, *MAINT*,
 17-20
 index segments, *MAINT*, 17-51
 sort statistics, *MAINT*, 17-51
 the O flag, *MAINT*, 17-39
 the R flag, *MAINT*, 17-51
 the SE flags, *MAINT*, 17-27
 the S flag, *MAINT*, 17-22
 the S\ flags, *MAINT*, 17-38

RDMS\$DEBUG_FLAGS logical name,
MAINT, 16-145

RDMS\$DEBUG_FLAGS_OUTPUT
 saving output access strategies,
MAINT, 17-20

RDMS\$DEBUG_FLAGS_OUTPUT
 Rdb/VMS logical name, *MAINT*,
 16-145

RDMS\$DIAG_FLAGS Rdb/VMS logical
 name, *MAINT*, 16-143

RDMS\$KEEP_PREP_FILES Rdb/VMS
 logical name, *MAINT*, 16-144

RDMS\$RUJ Rdb/VMS logical name,
MAINT, 16-144

RDMS error messages
 explanation files, *RDORM*, B-1

RDO

CHANGE DATABASE
 ADJUSTABLE LOCK
 GRANULARITY IS, *MAINT*,
 10-15

ADJUSTABLE LOCK
 GRANULARITY IS
 DISABLED, *MAINT*, 10-15

DICTIONARY IS REQUIRED,
MAINT, 10-26

EXTENT IS extent-pages PAGES,
MAINT, 10-11

JOURNAL FILE IS, *MAINT*,
 10-18

NOJOURNAL, *MAINT*, 10-18

NUMBER OF BUFFERS IS,
MAINT, 2-8, 10-16

NUMBER OF RECOVERY
 BUFFERS IS, *MAINT*, 10-17

NUMBER OF USERS IS,
MAINT, 10-13

NUMBER OF VAXCLUSTER
 NODES IS, *MAINT*, 10-14

OPEN IS, *MAINT*, 10-11

SNAPSHOT IS ENABLED
 DEFERRED, *MAINT*, 10-24

SNAPSHOT IS ENABLED
 IMMEDIATE, *MAINT*, 10-24

/SNAPSHOTS, *MAINT*, 10-18

CHANGE DATABASE statement
 changing database characteristics,
MAINT, 10-7

CHANGE INDEX statement,
MAINT, 16-50

database maintenance statements,
RDORM, 2-7

data definition statements, *RDORM*,
 2-2

data manipulation statements,
RDORM, 2-5

DEFINE DATABASE statement
 using, *MAINT*, 11-65

RDO (Cont.)

- DEFINE STORAGE MAP statement
 - PLACEMENT VIA INDEX clause,
MAINT, 10-62
- distributing database statements,
RDORM, 2-9
- DML differences from RDBPRE,
GURRR, 12-2
- error messages
 - explanation files, RDORM, B-1
- EXPORT statement
 - when to use, MAINT, 8-59
- HELP statement, RDORM, 9-297
- IMPORT statement
 - when to use, MAINT, 8-59
- interactive
 - using with distributed
transaction, DIST_TRANS,
5-1
- interactive control statements,
RDORM, 2-8
- invoking, RDORM, 1-1; GURRR,
1-5
- language elements, RDORM, 1-3
- obsolete statements
 - See Obsolete statements
- PLACE statement
 - use with defined hashed indexes,
MAINT, 16-130
- prompts, RDORM, 1-3
- reference descriptions, RDORM, 1-1
- START_TRANSACTION statement
 - reserving options, MAINT, 16-18
 - using, GURRR, 1-4
 - using Help in, RDORM, 1-2
- Read-only storage areas, MAINT, 16-30
- Read-only transaction, GUSQL, 2-22,
2-24; GURRR, 2-5, 2-7
- with SNAPSHOT file deferred,
MAINT, 16-103
- with SNAPSHOT file disabled,
MAINT, 16-99

- Read/write transaction, GUSQL, 2-23,
2-26; GURRR, 2-5, 2-7, 2-8
- READY statement, RDMLRM, 6-102
- opening a database, RDORM, 9-353
- REAL data type, RDORM, 5-6;
SQLRM, 3-36
- Record
 - See also Record value
 - active and inactive, MAINT, 16-78
 - data
 - availability of during trigger
updates, RDORM, 9-211
 - deleting, GURRR, 6-7
 - fragmentation, MAINT, 11-51,
16-93, 16-110
 - index node, MAINT, 11-38
 - locking, GURRR, 2-5
 - order for loading the database,
MAINT, 16-129
 - retrieving from a stream, RDORM,
9-291
 - segmented string, MAINT, 11-36
 - storage
 - performance and space usage,
MAINT, 13-5
 - storing, GURRR, 6-1
 - user-stored, MAINT, 11-35
- Record selection expression, GURRR,
1-7, 3-2
 - conditional expressions, GURRR, 3-7
 - CROSS clause, RDORM, 4-12
 - defined, RDORM, 4-1
 - FIRST clause, RDORM, 4-3
 - REDUCED TO clause, RDORM, 4-11
 - relation clause, RDORM, 4-6
 - restriction for START_STREAM
statement, RDORM, 9-422
 - SORTED BY clause, RDORM, 4-9;
GURRR, 3-4
 - syntax, RDORM, 4-2
 - WITH clause, RDORM, 4-8
- Record selection expression (RSE),
RDMLRM, 1-2
- Record stream, RDORM, 4-1

Record stream (Cont.)

- advancing
 - FETCH statement, *RDORM*, 9-276
- closing
 - END_STREAM statement, *RDORM*, 9-261
- DECLARE_STREAM statement, *RDORM*, 9-82
- displaying name of current stream, *RDORM*, 9-406
- forming, *GURRR*, 3-1, 9-5
- FOR statement, *RDORM*, 9-282
- retrieving records, *RDORM*, 9-291
- START_STREAM statement, *RDORM*, 9-422, 9-425
- updating data, *GURRR*, 6-8

Record value

- modifying
 - MODIFY statement, *RDORM*, 9-338
- retrieving
 - GET statement, *RDORM*, 9-291
 - PRINT statement, *RDORM*, 9-349

Recovering a database, *MAINT*, 9-1; *RDORM*, 6-91

- access after failure in VAXclusters, *MAINT*, 18-19
- in VAXclusters, *MAINT*, 18-29
- placement of RUJ files, *MAINT*, 9-30
- proper order to apply, *MAINT*, 9-19
- proper timing for changing AIJ file, *MAINT*, 9-20
- steps for, *MAINT*, 9-18
- surviving nodes, *MAINT*, 18-30
- using after-image journals, *MAINT*, 9-18, 16-79
- using recovery-unit journals, *MAINT*, 9-29

RECOVER statement

- See RMU/RECOVER command
- obsolete, *RDORM*, F-7

Recovery-unit journal, *MAINT*, 9-29

Recovery-unit journal (Cont.)

- directories, *MAINT*, 9-30
- displaying contents, *MAINT*, 9-32
- displaying output, *RDORM*, 6-68
- interpreting file headings, *MAINT*, 9-32
- logical names, *MAINT*, 9-30
- Re-creating dictionary definitions with INTEGRATE DATABASE, *RDORM*, 9-325
- REDUCED TO clause, *RDMLRM*, 4-30
 - of record selection expression, *RDORM*, 4-11
 - using with the SORTED BY clause, *RDMLRM*, 4-30

Redundancy

- eliminating, *DESIGN*, 2-8
- multi-user access, *MAINT*, 16-14
- problems, *MAINT*, 16-15
- Referential integrity, *DESIGN*, 3-37;
GURRR, 6-16
 - changing, *DESIGN*, 5-20

Reflexive join, *RDMLRM*, 4-16; *GURRR*, 4-7

REFRESH MONITOR LOG statement See RMU/MONITOR REOPEN_LOG command

- obsolete, *RDORM*, F-9
- REINITIALIZE TRANSFER statement, *SQLRM*, 6-450
- distributing databases, *RDORM*, 9-357

Relation

- See also Table definitions
- DEFINE RELATION statement, *RDORM*, 9-149
- DELETE RELATION statement, *RDORM*, 9-244
- SHOW RELATIONS statement, *RDORM*, 9-398
- for storing security audit journal records, *MAINT*, 4-25; *RDORM*, 6-71

Relation (Cont.)

- inserting records into, *RDORM*, 9-457
- loading, *RDORM*, 6-70
- modifying, *RDORM*, 9-48
- system, *RDORM*, 8-1
- Relational database design
 - goals of, *DESIGN*, 1-1
 - theory of, *DESIGN*, 1-1
 - tradeoffs, *DESIGN*, 1-1
- Relational Database Operator (RDO)
 - See RDO
- Relational join, *GUSQL*, 3-61; *GURRR*, 4-1
 - See also Joining relations
 - more than two relations, *GURRR*, 4-6, 5-4
 - reflexive joins, *GURRR*, 4-7
 - two relations, *GURRR*, 4-2
- Relational model, *GURRR*, 1-2; *SQLRM*, 1-2
- Relational operator, *RDMLRM*, 3-31; *RDORM*, 3-39; *GURRR*, 3-8
 - See also Conditional operator
 - internationalization support, *GURRR*, 3-9
- Relation clause, *RDMLRM*, 4-36
- Relation definitions
 - specifying default protection for, *RDORM*, 9-144
- Relation-specific constraint
 - See Table-specific constraint, *INTRO*, 1-4
- RELEASE statement, *GUSQL*, 12-6; *SQLRM*, 6-452
- Remote access, *MAINT*, 16-131
 - and distributed transaction, *DIST_TRANS*, 3-6
 - INVOKE DATABASE statement, *GURRR*, 2-3
- Request handle, *RDMLRM*, 5-4; *RDORM*, 3-5, 9-82, 9-282, 9-425, 9-458; *GURRR*, 9-41

Request handle (Cont.)

- restriction for FOR statement, *RDORM*, 9-283
- restriction for INVOKE DATABASE statement, *RDORM*, 9-332, 9-333
- restriction for START_STREAM statement, *RDORM*, 9-422
- setting scope, *RDORM*, 9-332
- REQUEST_HANDLE clause, *RDMLRM*, 6-106
- Requirements analysis in logical design, *DESIGN*, 2-1
- Reserved words
 - for Rdb/VMS statements, *RDORM*, A-1
- RESERVING clause, *GUSQL*, 2-19
- Reserving option
 - access conflicts, *MAINT*, 16-18
 - transactions, *MAINT*, 16-18
- Reserving options, *GURRR*, 2-11
- Resolving
 - unresolved transactions, *RDORM*, 6-100, 6-103
- Resolving distributed transaction, *DIST_TRANS*, 6-1 to 6-7
- Resource manager, *DIST_TRANS*, 2-2
 - in distributed transaction, *DIST_TRANS*, 1-4, 2-3
- Resource problems
 - CPU, *TUNING*, 2-48
 - I/O, *TUNING*, 2-2, 2-14
 - memory, *TUNING*, 2-46
- Restore operation
 - by area
 - purpose of, *RDORM*, 6-110
- RESTORE statement
 - obsolete, *RDORM*, F-10
- Restoring
 - databases, *MAINT*, 8-1, 8-38, 8-44e
 - by area, *RDORM*, 6-112
 - by area, purpose of, *RDORM*, 6-110
 - from tape, *MAINT*, 8-48

Restoring

databases (Cont.)

- IMPORT statement, *DESIGN*, 5-25, 5-36e; *RDORM*, 9-299
 - involved in distributed transactions, *DIST_TRANS*, 6-6
 - modifying database characteristics, *MAINT*, 8-52
 - RMU/RESTORE command, *RDORM*, 6-107
 - to change schema and storage area characteristics, *DESIGN*, 5-26
 - modifying after-image journaling, *MAINT*, 8-54
 - modifying page size, *MAINT*, 8-56
 - moving files, *MAINT*, 8-57
 - read-only storage areas, *MAINT*, 8-45e
 - threshold values, *MAINT*, 8-56
- ### Restructuring
- databases, *DESIGN*, 5-1 to 5-36
 - SQL options, *DESIGN*, 5-1
 - tables, *GUSQL*, 5-9
- ### Result table, *GUSQL*, 1-19, 3-1
- cursor, *GUSQL*, 4-6
 - deleting, *GUSQL*, 1-19
 - filtering
 - in SQL/Services, *SQLSRV*, 5-2

Retrieval

- indexed, *MAINT*, 16-34
 - sequential
 - buffer size, *MAINT*, 16-92
- ### Retrieving data, *GUSQL*, 3-1 to 3-75; *SQLRM*, 6-474
- achieving optimal performance for range retrieval, *DESIGN*, 4-14
 - based on
 - alternative condition, *GUSQL*, 3-45, 3-49
 - combined condition, *GUSQL*, 3-45, 3-47

Retrieving data

based on (Cont.)

- existence of set of rows, *GUSQL*, 3-42
- negated condition, *GUSQL*, 3-45
- null, *GUSQL*, 3-40
- range of values, *GUSQL*, 3-35
- conditional expressions, *GURRR*, 3-7
- eliminating duplicates, *GURRR*, 3-7
 - using REDUCED TO clause, *GURRR*, 3-18
- FIRST *n* clause, *GURRR*, 3-7
- for exact matches, *DESIGN*, 1-10
- improving performance using hashed index, *DESIGN*, 3-41
- joining relations, *GURRR*, 4-1 to 4-11
- joining table and view, *GUSQL*, 3-61
- LIMIT TO clause, *GUSQL*, 3-15, 4-10
- methods of, *DESIGN*, 1-10
- missing value, *GURRR*, 6-12
- optimizing performance, *DESIGN*, 4-11
- ordering rows, *GUSQL*, 3-19
- range retrieval and indexes, *DESIGN*, 3-41
- records in a relation, *GURRR*, 3-2
- records in sorted order, *GURRR*, 3-4
- records that do not satisfy a condition, *GURRR*, 3-13
- records that satisfy a single condition, *GURRR*, 3-10
- records that satisfy multiple conditions, *GURRR*, 3-11
- segmented strings, *GURRR*, 9-16
- selecting
 - all rows in table, *GUSQL*, 3-13
 - unique row, *GUSQL*, 3-16
- selecting columns, *GUSQL*, 3-4
- selecting fields, *GURRR*, 3-7
- sequentially, *DESIGN*, 1-10

Retrieving data (Cont.)

using

- column select expressions,
GUSQL, 3-67
- cursor, *GUSQL*, 4-1
- dbkey, *DESIGN*, 3-13; *GURRR*,
9-33
- hashed indexes, *DESIGN*, 1-10
- host language parameter,
GUSQL, 10-11
- indexes, *DESIGN*, 1-10
- indicator parameter, *GUSQL*,
10-20
- record streams, *GURRR*, 9-5,
9-6
- views, *GUSQL*, 3-67

using cursors, *SQLRM*, 6-352

Retrieving missing value, *RDMLRM*,
2-29

REVOKE statement

ACL style, *SQLRM*, 6-455

ANSI/ISO style, *SQLRM*, 6-465

Revoking privileges, *SQLRM*, 6-455

RMS (Record Management Services)
files

loading data, *MAINT*, 2-21

using RMU/LOAD command,
MAINT, 2-37

unloading data

using RMU/UNLOAD command,
MAINT, 2-37

RMU, *RDORM*, 6-1

area qualifiers

See Parameter qualifiers

bugcheck files, *MAINT*, 12-2

command parameters, *RDORM*, 6-2

command qualifiers, *RDORM*, 6-2

commands

auditing the use of, *MAINT*,
4-10; *RDORM*, 6-132

file qualifiers

See Parameter qualifiers

parameter qualifiers

defined, *RDORM*, 6-3

RMU

parameter qualifiers (Cont.)

positional semantics of, *RDORM*,
6-3

positional qualifiers

See Parameter qualifiers

RMU/ANALYZE/CARDINALITY

command, *MAINT*, 16-32;
RDORM, 6-9

RMU/ANALYZE command, *DESIGN*,

4-9; *MAINT*, 14-1 to 14-65;
RDORM, 6-5

analyzing record storage,
MAINT, 13-5

/AREAS qualifier, *MAINT*, 14-8
to 14-25, 18-32

/OPTIONS=DEBUG qualifier,
MAINT, 14-20

/OPTIONS=FULL qualifier,
MAINT, 14-14

/OPTIONS=NORMAL

qualifier, *MAINT*, 14-8

creating a binary output file,
MAINT, 14-2, 14-63

/INDEXES qualifier, *MAINT*,
14-28 to 14-42, 18-32

hashed indexes, *MAINT*,
14-33, 14-36

/OPTIONS=DEBUG qualifier,
MAINT, 14-33

/OPTIONS=FULL qualifier,
MAINT, 14-31

/OPTIONS=NORMAL

qualifier, *MAINT*, 14-28

sorted indexes, *MAINT*,
14-34

interpreting output, *MAINT*,
14-5

/LAREAS qualifier, *MAINT*,
14-25 to 14-28, 18-32

/OPTIONS=DEBUG qualifier,
MAINT, 14-27

/OPTIONS=FULL qualifier,
MAINT, 14-27

RMU

RMU/ANALYZE command
 /LAREAS qualifier (Cont.)
 /OPTIONS=NORMAL
 qualifier, *MAINT*, 14-25
 locking information, *MAINT*,
 14-4
 /OPTIONS=DEBUG qualifier,
 MAINT, 14-12
 overview, *MAINT*, 14-4
 /PLACEMENT qualifier, *MAINT*,
 14-42 to 14-63, 18-32
 hashed indexes, *MAINT*,
 14-53
 /OPTIONS=DEBUG qualifier,
 MAINT, 14-53
 /OPTIONS=FULL qualifier,
 MAINT, 14-47
 /OPTIONS=NORMAL
 qualifier, *MAINT*, 14-42
 sorted indexes, *MAINT*,
 14-55, 14-57
 record fragmentation, *MAINT*,
 16-93
 using, *MAINT*, 14-4
RMU/ANALYZE/INDEXES command,
 RDORM, 6-13
RMU/ANALYZE/PLACEMENT
 command, *RDORM*, 6-16
RMU/BACKUP/AFTER_JOURNAL
 command, *RDORM*, 6-37
RMU/BACKUP command, *RDORM*,
 6-20
 /AFTER_JOURNAL, *MAINT*, 9-8
 /CONTINUOUS, *MAINT*,
 9-11
 /UNTIL, *MAINT*, 9-11
/BACKUP
 full, *MAINT*, 8-13
/CHECKSUM_VERIFICATION
 qualifier, *MAINT*, 8-22
/CRC, *MAINT*, 8-33
/CRC=AUTODIN_II, *MAINT*,
 8-33

RMU

RMU/BACKUP command (Cont.)
 /CRC=CHECKSUM, *MAINT*,
 8-33
 /INCREMENTAL, *MAINT*, 8-19
 /INCREMENTAL=BY_AREA,
 MAINT, 8-32
 /INCREMENTAL=COMPLETE,
 MAINT, 8-32
 /INTERVAL, *MAINT*, 9-12
 /NOCRC, *MAINT*, 8-33
 /ONLINE, *MAINT*, 8-10
 PROTECTION, *MAINT*, 8-9
 tasks permitted, *MAINT*, 8-7
 /THRESHOLD, *MAINT*, 9-12
 truncating file names, *RDORM*,
 6-33
RMU/CLOSE command, *MAINT*,
 10-11; *RDORM*, 6-42
 examples, *MAINT*, 5-14
RMU/CONVERT command, *RDORM*,
 6-46
RMU/COPY_DATABASE command,
 RDORM, 6-50
RMU/DUMP
 /SNAPSHOT qualifier, *MAINT*,
 11-22e
RMU/DUMP/AFTER_JOURNAL
 command, *RDORM*, 6-59;
 DIST_TRANS, 6-5
RMU/DUMP/AREA command,
 MAINT, 11-13e
RMU/DUMP/BACKUP_FILE
 command, *RDORM*, 6-62
RMU/DUMP command, *MAINT*,
 10-18, 11-1
 /AREA qualifier, *MAINT*, 11-11
 format, *RDORM*, 6-54
 /HEADER, *MAINT*, 10-8, 10-21
 /LAREA qualifier, *MAINT*, 11-11
 /RECOVERY_JOURNAL,
 MAINT, 9-32
 /SNAPSHOT qualifier, *MAINT*,
 11-11

RMU

RMU/DUMP command (Cont.)
 /USERS, *MAINT*, 10-21

RMU/DUMP/LAREA command,
 MAINT, 11-17

RMU/DUMP/RECOVERY_JOURNAL
 command, *RDORM*, 6-68

RMU/DUMP/USERS command,
 DIST_TRANS, 6-3

RMU/LOAD/AUDIT command,
 MAINT, 4-2, 4-27

RMU/LOAD command, *RDORM*,
 6-70
 loading databases, *MAINT*, 2-37

RMU/MONITOR command
 START, *MAINT*, 3-3
 START/OUTPUT, *MAINT*, 3-5
 STOP, *MAINT*, 3-3, 3-5

RMU/MONITOR REOPEN_LOG
 command, *RDORM*, 6-79

RMU/MONITOR START command,
 RDORM, 6-80

RMU/MONITOR STOP command,
 RDORM, 6-82

RMU/OPEN command, *MAINT*,
 5-2e, 10-11; *RDORM*, 6-89

RMU/RECOVER command, *MAINT*,
 9-18; *RDORM*, 6-91
 /AIJ_BUFFERS qualifier,
 MAINT, 9-27
 /AREAS qualifier, *MAINT*, 9-27
 /ROOT qualifier, *MAINT*, 9-27
 /TRACE and /NOTRACE
 qualifiers, *MAINT*, 9-27
 /UNTIL qualifier, *MAINT*, 9-27

RMU/RECOVER/RESOLVE
 command, *DIST_TRANS*, 6-6

RMU/RESOLVE command,
 DIST_TRANS, 6-4

RMU/RESTORE command, *RDORM*,
 6-107; *DIST_TRANS*, 6-6
 /BLOCKS_PER_PAGE, *MAINT*,
 8-56
 /DIRECTORY, *MAINT*, 8-57

RMU

RMU/RESTORE command (Cont.)
 /FILE, *MAINT*, 8-57
 /INCREMENTAL, *MAINT*, 8-42
 /NOCCD_INTEGRATE, *MAINT*,
 8-51
 /PATH, *MAINT*, 8-58
 /SNAPSHOTS, *MAINT*, 8-57
 tasks permitted, *MAINT*, 8-38
 /THRESHOLDS, *MAINT*, 8-56

RMU/SET AUDIT command,
 MAINT, 4-2, 4-12, 4-14, 4-16,
 4-17, 4-18, 4-19

RMU/SHOW AUDIT command,
 MAINT, 4-2, 4-12, 4-14, 4-16,
 4-17, 4-18, 4-19; *RDORM*,
 6-143

RMU/SHOW command, *RDORM*,
 6-142
 SYSTEM, *MAINT*, 3-9, 18-32
 USERS, *MAINT*, 3-9, 18-32
 VERSION, *MAINT*, 3-9

RMU/SHOW STATISTICS command,
 MAINT, 15-1, 16-86; *RDORM*,
 6-148
 display modes, *MAINT*, 15-2
 display options, *MAINT*, 15-8
 formatted binary file, *MAINT*,
 15-45
 getting online help, *MAINT*,
 15-10
 /INTERACTIVE, *MAINT*, 15-2
 monitor locking, *MAINT*, 13-6
 monitor locks, *MAINT*, 16-135
 /OUTPUT, *MAINT*, 15-2
 report format options, *MAINT*,
 15-8
 /TIME, *MAINT*, 15-45

RMU/SHOW SYSTEM command,
 RDORM, 6-154

RMU/SHOW USERS command,
 RDORM, 6-155

RMU/SHOW VERSION command,
 RDORM, 6-157

RMU (Cont.)

- RMU/UNLOAD command, *RDORM*, 6-158
 - unloading databases, *MAINT*, 2-37
- RMU/VERIFY command, *MAINT*, 6-4, 6-10; *RDORM*, 6-161
 - /ALL, *MAINT*, 6-4, 6-11
 - /AREAS, *MAINT*, 6-11, 6-31
 - /CHECKSUM_ONLY, *MAINT*, 6-11, 6-21
 - /CONSTRAINTS, *MAINT*, 6-12
 - /INCREMENTAL, *MAINT*, 6-10
 - /INDEXES, *MAINT*, 6-12
 - /INDEXES/[NO]DATA, *MAINT*, 6-12
 - /LAREA, *MAINT*, 6-12
 - /LOG, *MAINT*, 6-18
 - /[NO]ROOT, *MAINT*, 6-11
 - qualifier functions, *MAINT*, 6-9
 - /SNAPSHOTS, *MAINT*, 6-12
 - /TRANSACTION_TYPE, *MAINT*, 6-18
- summary of maintenance tools, *INTRO*, 1-31
- VMS security access privileges for, *DESIGN*, 6-36
- RMU/ALTER command
 - entering RdbALTER, *MAINT*, 7-3
 - invoking the RdbALTER utility, *RDORM*, 6-4
- RMU audit events
 - enabling or disabling, *MAINT*, 4-10
- RMU/CONVERT command
 - called by RMU/RESTORE, *RDORM*, 6-107
- RMU/DUMP command
 - to show storage area, page, and record numbers, *MAINT*, 7-5
- RMU/RESTORE command
 - by area
 - purpose of, *RDORM*, 6-110
 - calling RMU/CONVERT, *RDORM*, 6-107

- RMU/SET AUDIT command, *RDORM*, 6-127
- RMU/VERIFY versus RdbALTER VERIFY, *MAINT*, 7-24
- Role-oriented privilege, *SQLRM*, 6-370
- ROLLBACK
 - in RdbALTER, *MAINT*, 7-25
- ROLLBACK command (RdbALTER), *RDORM*, 7-38
- ROLLBACK statement, *GUSQL*, 2-46; *RDMLRM*, 6-111; *SQLRM*, 6-472
 - undoing changes to a database, *RDORM*, 9-359
- Rolling back a transaction
 - in RdbALTER, *MAINT*, 7-25
- Root file
 - moving, *RDORM*, 6-85
- Root file (RDB), *DESIGN*, 4-2
- Row
 - See also* Retrieving data
 - definition of, *INTRO*, 1-2
 - deleting, *GUSQL*, 5-18; *SQLRM*, 6-283
 - inserting, *GUSQL*, 5-3
 - locking, *GUSQL*, 2-33
 - ordering, *GUSQL*, 3-19
 - retrieving, *GUSQL*, 3-1 to 3-75
 - sorting using collating sequence, *DESIGN*, 3-21
- RSE
 - See* Record selection expression
- RUJ
 - See* Recovery-unit journal
- Run-time error
 - handling, *GURRR*, 10-1
- S**
 - Sample database, *DESIGN*, 1-10; *SQLRM*, F-1
 - creating, *MAINT*, 1-16; *INTRO*, 1-15
 - definitions for, *DESIGN*, A-1

Sample database (Cont.)

- disk space requirements for
PERSONNEL, *INTRO*, 1-16
- generating, *DESIGN*, 1-10
- use of, *DESIGN*, 1-10

Schedules

- defining, *RDORM*, 9-164
- deleting, *RDORM*, 9-247

Schema

- attaching to, *GUSQL*, 2-3; *SQLRM*, 6-255
- changing
 - characteristics, *MAINT*, 10-6
- context, *GUSQL*, 2-1
- copying without data, *SQLRM*, 6-350
- creating, *DESIGN*, 1-6, 3-4;
SQLRM, 6-97 to 6-120
- using CDO definitions, *DESIGN*, 7-11
- using dictionary, *DESIGN*, 3-4
- declaring, *DESIGN*, 3-5; *GUSQL*, 2-3; *SQLRM*, 6-255
- declaring before issuing privilege statements, *DESIGN*, 6-7
- default protection on, *SQLRM*, 6-359
- definition of, *INTRO*, 1-3
- definitions in, *DESIGN*, 3-2
- deleting, *DESIGN*, 3-8, 5-23;
SQLRM, 6-311
- displaying information about,
SQLRM, 6-521
- modifying, *SQLRM*, 6-15 to 6-28
- naming, *SQLRM*, 3-7

Scope

- of database handle, *RDMLRM*, 6-11
- of dbkey, *DESIGN*, 3-13; *RDMLRM*, 6-11
- of request handle, *RDMLRM*, 6-12
- of transaction, *MAINT*, 16-27;
GUSQL, 2-42

Security

- See also* Access control list; Access privilege set; Privilege; Protection access privilege set, *DESIGN*, 6-2

Security (Cont.)

- alarms, *MAINT*, 4-3, 4-20;
RDORM, 6-127
 - disabling, *RDORM*, 6-128
 - enabling, *RDORM*, 6-128
 - and VMS, *DESIGN*, 6-36
 - audit journal, *DESIGN*, 6-1;
MAINT, 4-3, 4-25; *RDORM*, 6-127
 - DACCESS level security auditing,
MAINT, 4-13
 - database objects, *DESIGN*, 6-1
 - database recovery, *MAINT*, 9-18
 - database subjects, *DESIGN*, 6-1
 - event level security auditing,
MAINT, 4-16
 - flowchart of access to SQL/Services,
SQLSRV, 8-1f
 - interpreting AUDIT alarms, *MAINT*, 4-22
 - interpreting DACCESS alarms,
MAINT, 4-22
 - interpreting PROTECTION alarms,
MAINT, 4-24
 - interpreting RMU alarms, *MAINT*, 4-24
 - overview of Rdb/VMS, *DESIGN*, 6-1 to 6-37
 - Rdb/VMS and VMS, *DESIGN*, 6-4
 - reference monitor concept, *DESIGN*, 6-1
 - reviewing audit journal, *MAINT*, 4-28
 - schema level, *DESIGN*, 6-3
 - securing access to SQL/Services,
SQLSRV, 8-1
 - top level security auditing, *MAINT*, 4-16
 - user level security auditing, *MAINT*, 4-12
- ## Security auditing
- alarms
 - enabling, *MAINT*, 4-11
 - AUDIT event type, *MAINT*, 4-4

Security auditing (Cont.)

- audit journal
 - enabling, *MAINT*, 4-11
- DACCESS event type, *MAINT*, 4-5
- DACCESS level security auditing, *MAINT*, 4-11
- DACCESS privileges for database objects, *MAINT*, 4-13
- default characteristics, *MAINT*, 4-2
- defining audit events, *MAINT*, 4-11
- disabling, *RDORM*, 6-128
- displaying characteristics, *MAINT*, 4-2; *RDORM*, 6-143
- enabling, *RDORM*, 6-127, 6-128
- enabling and disabling events, *MAINT*, 4-16
- event level security auditing, *MAINT*, 4-11
- event types, *MAINT*, 4-4
- every access, *MAINT*, 4-17
- first access only, *MAINT*, 4-17
- /FLUSH qualifier, *MAINT*, 4-18
- four levels of security auditing, *MAINT*, 4-11
- how to establish auditing, *MAINT*, 4-12
- interpreting alarms, *MAINT*, 4-20
- monitoring resources, *MAINT*, 4-3
- /NOFLUSH qualifier, *MAINT*, 4-18
- overview, *MAINT*, 4-2
- PROTECTION event type, *MAINT*, 4-9
- reviewing
 - AUDIT alarms, *MAINT*, 4-22
 - audit information, *MAINT*, 4-20
 - audit journal records, *MAINT*, 4-28
 - DACCESS alarms, *MAINT*, 4-22
 - PROTECTION alarms, *MAINT*, 4-24
 - RMU alarms, *MAINT*, 4-24
- RMU event type, *MAINT*, 4-10
- setting
 - alarms, *MAINT*, 4-16

Security auditing

setting (Cont.)

- DACCESS events, *MAINT*, 4-13
- record auditing, *MAINT*, 4-16
- user level events, *MAINT*, 4-12
- starting, *MAINT*, 4-11, 4-16; *RDORM*, 6-133
- stopping, *MAINT*, 4-11, 4-16; *RDORM*, 6-133
- strategy for defining security auditing, *MAINT*, 4-11
- top level security auditing, *MAINT*, 4-11
- use of the RMU/LOAD/AUDIT command, *MAINT*, 4-25
- user level security auditing, *MAINT*, 4-11
- Security audit journal
 - loading into database, *MAINT*, 4-25; *RDORM*, 6-70
- Security audit journal records
 - defining database table for storing, *MAINT*, 4-25; *RDORM*, 6-71
 - defining relation for storing, *MAINT*, 4-25; *RDORM*, 6-71
- Segmented string, *MAINT*, 11-36; *RDORM*, 3-34; *GURRR*, 3-21
- See also* List
- closing, *RDORM*, 9-260
- creating
 - CREATE_SEGMENTED_STRING statement, *RDORM*, 9-74
- data type, *GURRR*, 8-3
- END_SEGMENTED_STRING statement, *RDORM*, 9-260
- handle, *RDORM*, 9-74
- modifying, *GURRR*, 9-28
- retrieving, *GURRR*, 9-16
- START_SEGMENTED_STRING, *RDORM*, 9-417
- retrieving values
 - FOR statement, *RDORM*, 9-287

Segmented string (Cont.)

storing

CREATE_SEGMENTED_STRING

statement, *RDORM*, 9-74

STORE statement, *RDORM*,
9-464

storing in multiple storage areas,
RDORM, 9-173

using to store data, *GURRR*, 9-24

SEGMENTED STRING data type,
RDORM, 5-8

SELECT expression, *GUSQL*, 3-4;
SQLRM, 3-98, 3-99

Selecting data

See Retrieving data; SELECT
statement

Selecting fields to be displayed,
GURRR, 3-7

Select list item

in dynamic SQL, *GUSQL*, 12-4,
12-5, 12-15

SELECT statement, *GUSQL*, 3-1 to
3-75; *SQLRM*, 3-99t, 6-474

FROM clause, *GUSQL*, 3-9

in dynamic SQL, *SQLSRV*, 2-6

singleton select, *GUSQL*, 3-4

specifying data source, *GUSQL*, 3-9

structure, *GUSQL*, 3-3

WHERE clause, *GUSQL*, 3-22

Semicolon (;)

in interactive statement, *GUSQL*,
1-13

Sequential access

analyzing strategies, *MAINT*, 17-47
locks, *MAINT*, 16-48

Sequential data retrieval, *DESIGN*,
1-10

SET ALL CONSTRAINTS statement,
SQLRM, 6-500

SET NOOUTPUT statement, *GUSQL*,
1-22

SET OUTPUT statement, *GUSQL*,
1-21, 1-22

SET statement, *SQLRM*, 6-482

SET statement (Cont.)

changing RDO parameters, *RDORM*,
9-362

Setting index characteristics, *MAINT*,
16-50

Setting RDO parameters

SET statement, *RDORM*, 9-362

SET TRANSACTION statement,
GUSQL, 2-18, 2-22, 2-22e, 2-43;
SQLRM, 6-502

WAIT clause, *GUSQL*, 2-30;

DIST_TRANS, 4-4

SET VERIFY statement, *GUSQL*, 1-21,
6-6

Shadow pages

hashed index, *MAINT*, 16-74

using two storage areas, *MAINT*,
16-74

Shareable entity

creating, *DESIGN*, 7-6

deleting, *DESIGN*, 7-26

modifying, *DESIGN*, 7-13

tracking, *DESIGN*, 7-3

Shareable image

creating, *GUSQL*, 9-3

creating with *RDBPRE*, *GURRR*,
11-14

creating with *RDML*, *GURRR*, 11-14

Share mode, *GUSQL*, 2-33; *GURRR*,
2-12

exclusive, *GURRR*, 2-16

exclusive write, *MAINT*, 16-23

protected, *GURRR*, 2-15

protected read, *MAINT*, 16-22

protected write, *MAINT*, 16-23

reading data

multi-user access, *MAINT*, 16-19

shared, *GURRR*, 2-15

writing data, *MAINT*, 16-20

SHOW ALL statement

displaying all database information,
RDORM, 9-375

SHOW COLLATING_SEQUENCE
statement, *RDORM*, 9-376

SHOW CONSTRAINT statement,
 RDORM, 9-377
 SHOW DATABASES statement,
 RDORM, 9-379
 SHOW DATE_FORMAT statement,
 RDORM, 9-381
 displaying time format, *RDORM*,
 9-381
 SHOW DICTIONARY statement
 displaying current default, *RDORM*,
 9-383
 SHOW FIELDS statement, *RDORM*,
 9-384
 SHOW INDEXES statement, *RDORM*,
 9-387
 SHOW LANGUAGE statement
 displaying language for date format,
 RDORM, 9-391
 SHOW MONITOR statement
 See also *RMU/SHOW SYSTEM*
 command
 obsolete, *RDORM*, F-16
 SHOW PRIVILEGES statement
 displaying access rights, *RDORM*,
 9-392
 SHOW PROTECTION statement
 displaying all ACL entries, *RDORM*,
 9-395
 SHOW RADIX_POINT statement,
 RDORM, 9-397
 SHOW RELATIONS statement,
 RDORM, 9-398
 SHOW statement, *SQLRM*, 6-521
 displaying information about
 databases, *RDORM*, 9-372
 using to check data compression,
 MAINT, 16-127
 SHOW STORAGE AREAS statement,
 RDORM, 9-400
 SHOW STORAGE MAPS statement,
 RDORM, 9-404
 SHOW STREAMS statement
 displaying name of current stream,
 RDORM, 9-406
 SHOW TABLES statement, *GUSQL*,
 1-5, 1-12
 SHOW TRANSACTION statement,
 RDORM, 9-407
 SHOW TRANSFER statement,
 RDORM, 9-409; *SQLRM*, 6-549
 distributing databases, *RDORM*,
 9-409
 SHOW TRIGGERS statement,
 RDORM, 9-413
 SHOW USERS statement
 See *RMU/SHOW USERS*
 obsolete, *RDORM*, F-17
 SHOW VERSIONS statement,
 RDORM, 9-416
 SHOW VIEWS statement, *GUSQL*,
 1-12
 SHR program section attribute,
 GUSQL, 9-3
 Single-file database, *GURRR*, 1-3
 as opposed to multifile, *DESIGN*, 3-8
 sample, *DESIGN*, A-1
 Singleton select, *GUSQL*, 3-4
 SMALLINT data type, *DESIGN*, 3-19;
 SQLRM, 3-35
 Snapshot file, *DESIGN*, 3-7, 3-54, 4-2
 accessing, *MAINT*, 16-97
 data structures, *MAINT*, 11-27
 disabling, *MAINT*, 16-98
 enabling, *MAINT*, 16-98
 options, *DESIGN*, 3-11
 transaction identification number,
 MAINT, 16-98
 using deferred snapshots, *MAINT*,
 16-103, 16-104e
 Snapshot page, *MAINT*, 11-22
 Snapshot statistics, *MAINT*, 15-30
 Snapshot storage area
 creating, *DESIGN*, 3-54
 SNP file type
 See Snapshot file
 Software Performance Monitor
 See *VAX Software Performance*
 Monitor

Software Performance Report (SPR)
form, *MAINT*, 12-1
submitting, *MAINT*, 12-11
when to submit, *MAINT*, 12-9
SORTED BY clause, *RDMLRM*, 4-44
of record selection expression,
RDORM, 4-9
using with the REDUCED TO clause,
RDMLRM, 4-30
Sorted index, *DESIGN*, 3-40; *RDORM*,
9-122; *SQLRM*, 6-79
access strategy, *MAINT*, 17-47
calculating node size, *MAINT*, 16-50
changing, *MAINT*, 10-39
page format required for storing,
DESIGN, 4-14
range retrieval, *DESIGN*, 4-14
retrieval with dbkey, *DESIGN*, 1-10
setting index characteristics, *MAINT*,
16-50
Sorting records, *RDORM*, 4-9;
GURRR, 3-4
Sorting rows, *GUSQL*, 3-19
See ORDER BY clause
using collating sequences, *DESIGN*,
3-21
Sort key, *GURRR*, 3-4
ascending order, *GURRR*, 3-5
descending order, *GURRR*, 3-5
SORTWORKn Rdb/VMS logical name,
MAINT, 16-149
Space area management (SPAM) pages,
DESIGN, 4-8; *MAINT*, 11-65
controlling thresholds, *MAINT*,
11-69
defaults, *MAINT*, 11-65
defining, *MAINT*, 11-65
defining intervals, *MAINT*, 11-69
entries, *MAINT*, 11-55, 11-73t
format, *MAINT*, 11-61, 11-66
INTERVAL clause, *DESIGN*, 4-19
mixed page format, *MAINT*, 11-65,
11-65e

Space area management (SPAM) pages
(Cont.)
optimizing SPAM intervals, *MAINT*,
16-113
page interval, *MAINT*, 11-55
sample scenario, *MAINT*, 11-70e
selecting threshold values, *MAINT*,
16-111
SPAM interval
formula for uniform storage area,
MAINT, 11-57
structure, *MAINT*, 11-55
THRESHOLD clause, *DESIGN*, 4-19
threshold value, *MAINT*, 11-57
uniform page format, *MAINT*, 11-57,
11-57e
Space usage
database, *MAINT*, 13-5
SPAM page
See Space area management (SPAM)
page
Specifying database privileges,
DESIGN, 6-25
SPM
See VAX Software Performance
Monitor
SPOOL statement
See also RMU/BACKUP/AFTER_
JOURNAL statement
obsolete, *RDORM*, F-18
SPR
See Software Performance Report
SQL
ALTER DOMAIN statement,
DESIGN, 5-6
ALTER INDEX statement, *MAINT*,
16-50
ALTER SCHEMA
ADJUSTABLE LOCK
GRANULARITY IS, *MAINT*,
10-15
ADJUSTABLE LOCK
GRANULARITY IS
DISABLED, *MAINT*, 10-15

SQL

ALTER SCHEMA (Cont.)
 DICTIONARY IS REQUIRED,
 MAINT, 10-26
 EXTENT IS extent-pages PAGES,
 MAINT, 10-11
 JOURNAL FILE IS, MAINT,
 10-18
 NOJOURNAL, MAINT, 10-18
 NUMBER OF BUFFERS IS,
 MAINT, 2-8, 10-16
 NUMBER OF RECOVERY
 BUFFERS IS, MAINT, 10-17
 NUMBER OF USERS IS,
 MAINT, 10-13
 NUMBER OF VAXCLUSTER
 NODES IS, MAINT, 10-14
 OPEN IS, MAINT, 10-11
 SNAPSHOT IS ENABLED
 DEFERRED, MAINT, 10-24
 SNAPSHOT IS ENABLED
 IMMEDIATE, MAINT, 10-24
 /SNAPSHOTS, MAINT, 10-18
ALTER SCHEMA statement
 database characteristics that can
 be changed, MAINT, 10-7
ALTER TABLE statement, DESIGN,
 5-7
and VAX DATATRIEVE, SQLRM,
 3-49
authorization identifier, DESIGN,
 3-12; GUSQL, 2-9; SQLRM,
 3-10
column names, SQLRM, 3-13
CREATE SCHEMA statement
 using, DESIGN, 1-6, 3-4, 3-8;
 MAINT, 11-65
CREATE STORAGE MAP statement,
 DESIGN, 3-10, 4-3
 PLACEMENT VIA INDEX clause,
 MAINT, 10-62
creating databases, DESIGN, 3-1
data definition statements, SQLRM,
 2-2

SQL (Cont.)

 data dictionary path names, SQLRM,
 3-9
 data manipulation statements,
 SQLRM, 2-5
 data types, SQLSRV, 11-1t;
 SQLRM, 3-32
 differences from ANSI standard,
 SQLRM, B-1
 distributed transaction
 DISTRIBTRAN privilege,
 DIST_TRANS, 3-6
 using with, DIST_TRANS, 4-1
 to 4-24
 DROP TABLE statement, DESIGN,
 5-7
 dynamic DECLARE CURSOR
 statement, SQLRM, 6-245
 dynamic SQL, SQLRM, 3-29, 6-287,
 6-338, 6-343, 6-439, 6-452, D-1
 dynamic SQL statements, SQLRM,
 2-9
 error messages, SQLRM, A-1
 executable statements, SQLRM, 2-12
 EXPORT statement, DESIGN, 5-25
 when to use, MAINT, 8-59
 extended dynamic DECLARE
 CURSOR statement, SQLRM,
 6-249
 file specifications, SQLRM, 3-7
 formatting clause, SQLRM, 3-49
 functions, SQLRM, 2-1 to 2-16,
 3-68
 GRANT statement, SQLRM, 6-379
 handling errors, SQLRM, A-1, C-1
 help online, SQLRM, 6-389
 IMPORT statement, DESIGN, 5-25
 when to use, MAINT, 8-59
 interactive
 use in database design, DESIGN,
 3-3
 interactive statements, SQLRM, 2-8
 introduction to, SQLRM, 1-1

SQL (Cont.)

- language and syntax elements,
SQLRM, 3-1 to 3-109
- line terminators, *SQLRM*, 3-2
- logical names, *SQLRM*, G-1
- message vector, *SQLRM*, C-1
- module language, *SQLRM*, 2-11, 4-1
to 4-41
 - syntax, *SQLRM*, 4-3
- nonexecutable statements, *SQLRM*,
2-12
- optional keywords, *SQLRM*, 3-2
- precompiler statements, *SQLRM*,
2-10
- query header, *SQLRM*, 3-51
- required keywords, *SQLRM*, 3-2
- REVOKE ANSI-style statement,
SQLRM, 6-465
- sample database, *SQLRM*, F-1
- schema names, *SQLRM*, 3-7
- SET TRANSACTION statement
 - reserving options, *MAINT*, 16-18
- SQLCA, *SQLRM*, C-1
- SQLDA, *SQLRM*, D-1
- statements, *SQLRM*, 6-1 to 6-566
 - summary, *SQLRM*, 2-13
- summary of using to access Rdb/VMS
database, *INTRO*, 1-20
- table names, *SQLRM*, 3-12
- user-supplied names, *SQLRM*, 3-3
- using in programs, summary,
INTRO, 1-24
- using VAX Data Distributor
statements, *SQLRM*, 2-7
- using with other Digital database
products, *SQLRM*, 1-1
- view names, *SQLRM*, 3-12
- SQL\$CLOSE_CURSOR routine
 - with distributed transaction,
DIST_TRANS, 4-9
- SQL\$DATABASE logical name,
GUSQL, 1-4
- SQL\$DISABLE_CONTEXT logical
name, *DIST_TRANS*, 4-3
- SQL\$EDIT logical name, *GUSQL*, 1-23
- SQL\$GET_ERROR_TEXT call, *GUSQL*,
11-23
- SQL\$KEEP_PREP_FILES Rdb/VMS
logical name, *MAINT*, 16-144
- SQL\$SIGNAL call, *GUSQL*, 11-20
- SQLCA, *SQLSRV*, 2-12, 10-6;
SQLRM, 6-407, C-1
 - use of in dynamic SQL, *SQLSRV*,
2-8
- SQLCA (SQL Communications Area),
GUSQL, 11-7
- SQLCODE field, *GUSQL*, 11-6
- SQL Communications Area
 - See *SQLCA*
- SQLDA, *SQLSRV*, 2-7, 2-12, 10-11;
SQLRM, 6-407, D-1
 - DESCRIBE statement, *SQLRM*,
6-287
- SQLDA (SQL Descriptor Area),
GUSQL, 12-5; *SQLRM*, D-1
- SQL Descriptor Area
 - See *SQLDA*
- SQLINI
 - command procedure, *GUSQL*, 1-22
 - logical name, *GUSQL*, 1-22
- SQL module
 - declaring parameter
 - using FROM clause, *GUSQL*,
10-10
 - FROM dictionary clause, *GUSQL*,
10-3, 10-10
- SQL module language, *GUSQL*, 7-2 to
7-31; *SQLRM*, 4-1 to 4-41
 - See also Module language; SQL
advantages over interactive SQL,
INTRO, 1-25
 - and distributed transaction,
DIST_TRANS, 4-2, 4-9 to 4-15
 - converting interactive statement to,
GUSQL, 6-7
 - procedure in, *GUSQL*, 7-13
 - summary, *INTRO*, 1-25
- SQL module processor, *GUSQL*, 7-25

SQL module processor (Cont.)

- command line qualifier, *GUSQL*, 7-26
- invoking, *GUSQL*, 7-30
- SQL precompiler, *GUSQL*, 8-1 to 8-18; *SQLRM*, 5-1 to 5-38
- See also* Precompiler
- command line, *SQLRM*, 5-4
- command line qualifier, *GUSQL*, 8-12; *SQLRM*, 5-5
- data type conversion, *SQLRM*, 5-10
- error log file, *GUSQL*, 8-17
- input file, *GUSQL*, 8-11t
- invoking, *GUSQL*, 8-14
- output file, *GUSQL*, 8-11t
- parameter name processed by, *GUSQL*, 10-11
- processing source file, *GUSQL*, 8-10
- program section created by, *GUSQL*, 9-3
- statements, *SQLRM*, 2-10
- supported Ada variable declarations, *SQLRM*, 5-11
- supported COBOL variable declarations, *SQLRM*, 5-24
- supported C variable declarations, *SQLRM*, 5-18
- supported FORTRAN variable declarations, *SQLRM*, 5-27
- supported host language variable declarations, *SQLRM*, 5-10
- supported Pascal variable declarations, *SQLRM*, 5-30
- supported PL/I variable declarations, *SQLRM*, 5-36

SQL/Services

- API routines, *SQLSRV*, 9-1, 9-6 to 9-75
- as dynamic SQL statements, *SQLSRV*, 2-2
- names, *SQLSRV*, 9-2
- architecture, *SQLSRV*, 1-2f
- building application programs, *SQLSRV*, 2-13 to 2-15

SQL/Services

- building application programs (Cont.)
- on Macintosh, *SQLSRV*, 2-15
- on MS-DOS, *SQLSRV*, 2-13
- on OS/2, *SQLSRV*, 2-14
- on ULTRIX, *SQLSRV*, 2-14
- on ULTRIX for RISC, *SQLSRV*, 2-14
- on VMS, *SQLSRV*, 2-13
- client/server model, *SQLSRV*, 1-1
- components, *SQLSRV*, 1-2
- creating and releasing an association, *SQLSRV*, 4-6
- creating client/server association, *SQLSRV*, 2-8
- creating DECnet link, *SQLSRV*, 9-9
- data structures, *SQLSRV*, 10-1
- documentation format, *SQLSRV*, 10-1
- SQLVAR*, *SQLSRV*, 10-13
- data structures used, *SQLSRV*, 2-12 to 2-13
- data types, *SQLSRV*, 11-1
- disabling access to, *SQLSRV*, 8-2
- enabling access to, *SQLSRV*, 8-2
- enabling or disabling API functions, *SQLSRV*, 10-3
- environment variables, *SQLSRV*, B-1
- executing a prepared SQL statement, *SQLSRV*, 9-18
- filter expression functions, *SQLSRV*, A-1
- filtering result tables, *SQLSRV*, 5-2
- flowchart of access to, *SQLSRV*, 8-1f
- installation log files, *SQLSRV*, D-1
- introduction to, *SQLSRV*, 1-1
- IVP log files, *SQLSRV*, D-1
- logging used in, *SQLSRV*, 6-1
- parameters, *SQLSRV*, 9-3
- process pooling, *SQLSRV*, 7-1
- routines, *SQLSRV*, 2-8 to 2-12
- running the sample application, *SQLSRV*, 4-4

SQL/Services (Cont.)

- sample application, *SQLSRV*, 4-1
 - on VMS operating system, *SQLSRV*, 4-2
- sample application on Macintosh operating system, *SQLSRV*, 4-3
- sample application on MS-DOS operating system, *SQLSRV*, 4-2
- sample application on OS/2 operating system, *SQLSRV*, 4-3
- sample application on ULTRIX for RISC operating system, *SQLSRV*, 4-3
- sample application on ULTRIX operating system, *SQLSRV*, 4-3
- securing application access to, *SQLSRV*, 8-1
- security, *SQLSRV*, 8-1
- source code listings for *SQLSRV\$DYNAMIC* program, *SQLSRV*, C-1
- SQLCA*, *SQLSRV*, 2-12
- SQLCA* data structure, *SQLSRV*, 10-6
- SQLDA*, *SQLSRV*, 2-12
- SQLDA* data structure, *SQLSRV*, 10-11
- storing information when an error occurs, *SQLSRV*, 10-6
- summary, *INTRO*, 1-28
- using C language in, *SQLSRV*, 1-7
- using DECnet, *SQLSRV*, 1-2
- using dynamic SQL with, *SQLSRV*, 1-2
- SQLSRV_ENV_STR* data structure, *SQLSRV*, 10-16
- SQL statements, *SQLRM*, 6-1 to 6-566
 - See also SQL
 - compiling, *SQLSRV*, 9-36
 - data definition, *SQLRM*, 2-2
 - data manipulation, *SQLRM*, 2-5
 - dynamic, *SQLRM*, 2-9, 3-29
 - executable, *SQLRM*, 2-12

SQL statements (Cont.)

- interactive control, *SQLRM*, 2-8
- keywords, *SQLRM*, 3-1
- line terminators, *SQLRM*, 3-1, 3-2
- nonexecutable, *SQLRM*, 2-12
- not supported for V4.0, *SQLRM*, H-1
- optional keywords, *SQLRM*, 3-2
- prepared
 - executing, *SQLSRV*, 9-18
 - preparing, *SQLSRV*, 9-36
 - required keywords, *SQLRM*, 3-2
 - summary, *SQLRM*, 2-13
 - user-supplied names, *SQLRM*, 3-3
 - using VAX Data Distributor statements, *SQLRM*, 2-7
 - using with context structure, *DIST_TRANS*, 4-8
- SQLVAR* data structure, *SQLSRV*, 10-13
- \$SQL_VARCHAR* data type, *GUSQL*, 8-4, 10-33
- SRPCOUNT* parameter
 - values, *MAINT*, 16-134
- SRPCOUNTV* parameter
 - values, *MAINT*, 16-134
- Stall activity statistics, *MAINT*, 15-32
- Stall message display, *MAINT*, 15-34
- Starting
 - security auditing, *MAINT*, 4-11, 4-16
- Starting security auditing, *RDORM*, 6-133
- Starting transactions, *GURRR*, 2-4
 - distributed, *DIST_TRANS*, 2-2, 2-3
- STARTING WITH expression, *RDMLRM*, 3-33
 - with multinational characters, *RDMLRM*, 3-34
- STARTING WITH predicate, *SQLRM*, 3-96
- START TRANSFER statement, *SQLRM*, 6-554
 - distributing databases, *RDORM*, 9-453

START_SEGMENTED_STRING
 statement
 retrieving segmented strings,
RDORM, 9-417

START_STREAM statement, *GURRR*,
 6-8
 declared, *RDMLRM*, 6-115
 record stream, *RDORM*, 9-422,
 9-425
 restrictions for RSE and handles,
RDORM, 9-422
 transaction handle restriction,
RDORM, 9-426
 undeclared, *RDMLRM*, 6-120

START_TRANSACTION statement,
RDMLRM, 6-128; *GURRR*, 2-4,
 2-6, 2-30e, 6-1

DISTRIBUTED_TRANSACTION
 clause, *DIST_TRANS*, 5-8, 5-14
 with two-phase commit protocol,
RDMLRM, 6-128

Statement format
 in RDO, *RDORM*, 1-3

Statements
 database maintenance, *RDORM*, 2-7
 data definition, *RDORM*, 2-2;
SQLRM, 2-2
 data manipulation, *RDORM*, 2-5;
SQLRM, 2-5
 distributing databases, *RDORM*, 2-9
 interactive control, *RDORM*, 2-8
 no longer supported in RDO
 See *Obsolete statements*

VAX Data Distributor, *RDORM*, 2-9

Statistical expression, *RDORM*, 3-11;
SQLRM, 3-68

Statistical function
 See *Function*

Statistics
 See also *DECTrace*
 after-image journal, *MAINT*, 15-25
 database, *MAINT*, 13-6, 15-1
 database process, *RDORM*, 6-148
 display formats, *MAINT*, 15-4

Statistics (Cont.)
 displaying, *RDORM*, 6-148
 event-based data
 using *DECTrace*, *MAINT*, 15-49
 file I/O operations by file, *MAINT*,
 15-36
 formatted binary file, *MAINT*, 15-45
 gathering on Rdb/VMS applications
 using *DECTrace*, *MAINT*, 15-49
 getting online help, *MAINT*, 15-10
 graphic display format, *MAINT*, 15-4
 hashed index, *MAINT*, 15-20
 how to collect event data
 using *DECTrace*, *MAINT*, 15-50
 I/O display page, *MAINT*, 15-13
 I/O stall, *MAINT*, 15-32
 index (insertion), *MAINT*, 15-17
 index (removal), *MAINT*, 15-19
 index (retrieval), *MAINT*, 15-15
 lock, *MAINT*, 15-23
 locks for one statistics field, *MAINT*,
 15-44
 numbers display format, *MAINT*,
 15-6
 one lock type, *MAINT*, 15-39
 online mode, *MAINT*, 15-2
 output file description, *MAINT*,
 15-45
 PIO file activity, *MAINT*, 15-27
 record activity, *MAINT*, 15-29
 replay mode, *MAINT*, 15-2
 reset option, *MAINT*, 15-8
 set rate option, *MAINT*, 15-8
 snapshot, *MAINT*, 15-30
 stall messages, *MAINT*, 15-34
 time plot display, *MAINT*, 15-7
 transaction duration, *MAINT*, 15-33
 types of, *MAINT*, 15-10
 virtual memory use, *MAINT*, 15-22
 writing output to a file, *MAINT*, 15-8

STOP MONITOR statement
 See also *RMU/STOP MONITOR*
 command
 obsolete, *RDORM*, F-20

Stopping

- security auditing, *MAINT*, 4-11, 4-16; *RDORM*, 6-133
- STOP TRANSFER statement, *RDORM*, 9-456; *SQLRM*, 6-558
- distributing databases, *RDORM*, 9-456
- Storage area, *DESIGN*, 3-7; *SQLRM*, 3-31
 - adding new, *MAINT*, 10-28
 - adjusting parameters, *MAINT*, 16-106, 16-107t
 - calculating size
 - potential problems, *MAINT*, 16-73
 - changing, *MAINT*, 10-31
 - creating, *DESIGN*, 3-7; *SQLRM*, 6-121 to 6-129
 - multiple, *DESIGN*, 4-3
 - deleting, *MAINT*, 10-31
 - distributing tables and indexes
 - among, *DESIGN*, 4-8
 - estimating size for mixed page format, *MAINT*, 16-57
 - estimating sizes, *MAINT*, 16-68e
 - for lists, *SQLRM*, 6-134
 - for segmented strings, *RDORM*, 9-173
 - for table rows, *SQLRM*, 6-132
 - fragmented records, *MAINT*, 16-93
 - hashed indexes
 - shadow pages, *MAINT*, 16-74
 - maps, *MAINT*, 11-10
 - mixed page format, *MAINT*, 11-9
 - estimating number of SPAM pages, *MAINT*, 16-70
 - moving, *DESIGN*, 5-26; *MAINT*, 10-31, 10-35; *RDORM*, 6-85
 - naming, *SQLRM*, 3-31
 - page format, *MAINT*, 16-111
 - parameters
 - default values, *RDORM*, E-1
 - maximum values, *RDORM*, E-1
 - minimum values, *RDORM*, E-1

Storage area (Cont.)

- partitioning data, *RDORM*, 9-171
 - read-only, *MAINT*, 16-30
 - RDB\$SYSTEM, *MAINT*, 16-31
 - updating cardinalities, *MAINT*, 16-32
 - reorganizing databases
 - definition sequence, *MAINT*, 10-58
 - requirements
 - journal files, *MAINT*, 9-40
 - restoring by, *RDORM*, 6-112
 - purpose of, *RDORM*, 6-110
 - SHOW STORAGE AREAS statement, *RDORM*, 9-400
 - specifying on-disk structure for
 - hashed indexes, *DESIGN*, 4-7
 - manipulating SPAM pages, *DESIGN*, 4-8
 - storing lists, *DESIGN*, 4-19
 - structures, *MAINT*, 11-27
 - uniform page format, *MAINT*, 11-6
 - using more than one for
 - a table, *DESIGN*, 4-9
- Storage area file (RDA), *DESIGN*, 4-2
- Storage area parameters
 - default values, *RDORM*, E-1
 - maximum values, *RDORM*, E-1
 - minimum values, *RDORM*, E-1
- Storage map, *SQLRM*, 3-31
 - adjusting parameters, *MAINT*, 16-116, 16-117t
 - changes possible, *MAINT*, 10-45
 - changing, *MAINT*, 10-43, 10-45; *RDORM*, 9-61; *SQLRM*, 6-29 to 6-34
 - changing or creating when indexes change, *MAINT*, 10-43
 - creating, *SQLRM*, 6-130 to 6-137
 - data compression
 - considerations, *MAINT*, 16-128
- DEFINE STORAGE MAP statement, *RDORM*, 9-171

Storage map (Cont.)

- defining for tables in multfile databases, *DESIGN*, 4-4
 - deleting, *MAINT*, 10-45, 10-51;
SQLRM, 6-313
 - effect of changes on row placement, *MAINT*, 10-46
 - for lists, *DESIGN*, 4-19
 - modifying, *RDORM*, 9-61; *SQLRM*, 6-29 to 6-34
 - naming, *SQLRM*, 3-31
 - PLACEMENT VIA INDEX option, *MAINT*, 16-117
 - reorganizing databases
 - definition sequence, *MAINT*, 10-58
 - SHOW STORAGE MAPS statement, *RDORM*, 9-404
- Storage methods, *DESIGN*, 1-8
- Storage segment
 - fragmented record, *MAINT*, 11-52
 - structure, *MAINT*, 11-33
- STORE clause
 - in CREATE INDEX statement, *DESIGN*, 4-3
 - in CREATE STORAGE MAP statement, *DESIGN*, 4-3
- STORE statement, *RDMLRM*, 6-140; *GURRR*, 6-1
- adding database records, *RDORM*, 9-457
 - storing segmented strings, *RDMLRM*, 6-151; *RDORM*, 9-464
 - transaction handle restriction, *RDORM*, 9-458
- Storing a value
 - using indicator parameter, *GUSQL*, 10-21
- Storing data, *GUSQL*, 5-3; *SQLRM*, 6-414, 6-560
 - using lists, *GUSQL*, 5-11
 - using segmented strings, *GURRR*, 9-24

Storing row

See INSERT statement

Strategy

- access, *MAINT*, 16-42
- conjunct, *MAINT*, 17-47
- displaying with RDMS\$DEBUG_FLAGS, *MAINT*, 17-20
- query optimizer, *MAINT*, 16-47, 17-2
- retrieval, *MAINT*, 16-47
- saving output with RDMS\$DEBUG_FLAGS_OUTPUT, *MAINT*, 17-20

Stream

See Record stream

String literal

- TODAY, *RDORM*, 3-8
- TOMORROW, *RDORM*, 3-8
- YESTERDAY, *RDORM*, 3-8

Structure

- compared to record, *GUSQL*, 10-2
- space area management (SPAM) page, *MAINT*, 11-55

Structured programming

- BASIC program, *GURRR*, 13-31
- COBOL programs, *GURRR*, 14-32
- C programs, *GURRR*, 17-31
- FORTRAN program, *GURRR*, 15-34
- Pascal programs, *GURRR*, 18-30

Structured query language (SQL)

See SQL

Substring manipulation, *GUSQL*, 3-5, 3-7

SUM function, *GUSQL*, 3-54; *SQLRM*, 3-69

Symbol (DCL)

to invoke precompiler, *GUSQL*, 8-14

Syntax diagrams

reading, *RDORM*, xvii; *SQLRM*, 1-5

SYS\$COMMON logical name, *MAINT*, 18-19

SYS\$LANGUAGE logical name, *RDMLRM*, 1-5

SYS\$PUTMSG call, *GUSQL*, 11-22
 SYSGEN parameters, *MAINT*, 16-132
 SYSMWCNT parameter
 values, *MAINT*, 16-136
 SYSTEM record, *MAINT*, 11-46
 hashed index structure, *MAINT*,
 11-44
 System relation, *RDORM*, 8-1
 RDB\$CONSTRAINTS, *RDORM*, 8-4
 RDB\$CONSTRAINT_RELATIONS,
 RDORM, 8-5
 RDB\$DATABASE, *RDORM*, 8-6
 RDB\$FIELDS, *RDORM*, 8-11
 RDB\$FIELD_VERSIONS, *RDORM*,
 8-9
 RDB\$INDEX_SEGMENTS, *RDORM*,
 8-15
 RDB\$INDICES, *RDORM*, 8-16
 RDB\$RELATIONS, *RDORM*, 8-20
 RDB\$RELATION_FIELDS, *RDORM*,
 8-17
 RDB\$VIEW_RELATIONS, *RDORM*,
 8-23
 RDBVMS\$COLLATIONS, *RDORM*,
 8-24
 RDBVMS\$INTERRELATIONS,
 RDORM, 8-25
 RDBVMS\$PRIVILEGES, *RDORM*,
 8-26
 RDBVMS\$RELATION_
 CONSTRAINTS, *RDORM*,
 8-27
 RDBVMS\$RELATION_
 CONSTRAINT_FLDS,
 RDORM, 8-31
 RDBVMS\$STORAGE_MAPS,
 RDORM, 8-31
 RDBVMS\$STORAGE_MAP_AREAS,
 RDORM, 8-32
 RDBVMS\$TRIGGERS, *RDORM*,
 8-33

T

Table

See also Relation
 and index
 specifying storage area for,
 DESIGN, 4-3
 changing, *DESIGN*, 5-7; *SQLRM*,
 6-35 to 6-51
 constraint, *DESIGN*, 3-31; *SQLRM*,
 6-138
 creating, *DESIGN*, 3-23, 3-26;
 SQLRM, 6-138 to 6-167
 declaring, *SQLRM*, 6-268
 defining protection for, *DESIGN*,
 6-22
 definition of, *INTRO*, 1-2
 deleting, *DESIGN*, 5-7; *SQLRM*,
 6-316
 deleting row from, *GUSQL*, 5-18
 disabling data compression, *MAINT*,
 16-118
 displaying, *GUSQL*, 1-12
 eliminating redundancy from,
 DESIGN, 2-8
 inserting row, *GUSQL*, 5-1
 joining, *GUSQL*, 3-61
 loading, *RDORM*, 6-70
 locking, *GUSQL*, 2-33
 modifying, *DESIGN*, 5-7; *SQLRM*,
 6-35 to 6-51
 naming, *DESIGN*, 5-14; *SQLRM*,
 3-12
 needing separate storage areas,
 DESIGN, 4-8
 normalizing, *DESIGN*, 2-8
 partitioning across multiple storage
 areas, *DESIGN*, 4-9, 4-17
 reserving, *GUSQL*, 2-34, 2-35
 result, *GUSQL*, 1-19, 3-1
 updating, *GUSQL*, 5-12
 Table cursor, *GUSQL*, 4-5
See also Cursor

Table cursor (Cont.)

declaring, *SQLSRV*, 9-16

Table-specific constraint

creating, *SQLRM*, 6-138

definition of, *INTRO*, 1-4

deleting, *SQLRM*, 6-299

modifying, *SQLRM*, 6-35

Tapes

checking labels, *RDORM*, 6-34, 6-66, 6-120

Target page number

determining before storing records, *RDORM*, 9-346

Template

in LSE, *GUSQL*, 1-25

Terminology differences, *INTRO*, 1-11t

Threshold values

selecting for SPAMs, *MAINT*, 16-111

Time format

SET DATE_FORMAT statement, *RDORM*, 9-363

SHOW DATE_FORMAT statement, *RDORM*, 9-381

Time literal

conversion to international format, *RDMLRM*, 1-5

TINYINT data type, *SQLRM*, 3-35

TMPMBX VMS privilege, *MAINT*, 16-141

TODAY string literal

translation of, *RDORM*, 3-8; *SQLRM*, 3-44

TOMORROW string literal

translation of, *RDORM*, 3-8; *SQLRM*, 3-44

Total

See SUM function

TOTAL function, *RDMLRM*, 5-23; *RDORM*, 3-3

TOTAL statistical function, *RDORM*, 3-13

TPU editor

See VAXTPU editor

Tracking

database changes using triggers, *DESIGN*, 3-38

Tracking entities, *DESIGN*, 7-1

Tradeoffs of relational design, *DESIGN*, 1-1

Transaction, *GUSQL*, 2-1; *GURRR*, 2-4

abnormal termination, *MAINT*, 9-2

access modes, *GURRR*, 2-6

analysis, *DESIGN*, 2-9

batch-update, *MAINT*, 16-26;

GUSQL, 2-28; *GURRR*, 2-7, 2-9; *DIST_TRANS*, 2-1

COMMIT statement, *RDORM*, 9-69

committing, *GUSQL*, 2-46; *SQLRM*, 6-60

context, *GUSQL*, 2-1

declaring, *SQLRM*, 6-273

definition of, *INTRO*, 1-6

distributed, *RDMLRM*, 6-130

See also Distributed transaction completing, *DIST_TRANS*, 1-7, 2-17, 6-1 to 6-7

duration statistics, *MAINT*, 15-33

ending, *GUSQL*, 2-46; *GURRR*, 2-25; *SQLRM*, 6-472

failure to start, *GUSQL*, 11-38

in after-image journals, *MAINT*, 9-3

involving multiple databases, *GUSQL*, 2-23

mapping, *DESIGN*, 2-10

normal completion, *MAINT*, 9-2

prototype, *DESIGN*, 2-12

read-only, *GUSQL*, 2-24; *GURRR*, 2-5, 2-7

read/write, *GUSQL*, 2-26; *GURRR*, 2-5, 2-7, 2-8

ROLLBACK statement, *RDORM*, 9-359

rolling back, *GUSQL*, 2-46

scope, *MAINT*, 16-27; *GUSQL*, 2-42; *GURRR*, 2-24

Transaction

scope (Cont.)

effect on long and short

transactions, *MAINT*, 16-28t

sequence number (TSN), *MAINT*,
11-31, 11-31e

share modes, *GURRR*, 2-6

specifying characteristics of, *GUSQL*,
2-18

starting, *GUSQL*, 1-20, 2-18;
GURRR, 2-4

START_TRANSACTION statement,
RDORM, 9-431

update, *MAINT*, 9-2

abnormal termination, *MAINT*,
9-2

exception using invalid dbkey,
MAINT, 12-8

using deferred snapshots, *MAINT*,
16-104e

using two-phase commit protocol,
GUSQL, 2-2

waiting for lock release, *SQLRM*,
6-507

Transaction handle, *RDORM*, 3-5,
9-83, 9-283, 9-426, 9-458;
GURRR, 9-40

restriction for START_STREAM
statement, *RDORM*, 9-422,
9-426

Transaction identification number

in precompiled SQL statement,
SQLRM, 5-2

Transaction manager, *DIST_TRANS*,
2-2

in distributed transaction, *DIST_*
TRANS, 1-4, 2-3

Transaction sequence number (TSN)
index, *MAINT*, 11-31e

TRANSACTION_HANDLE clause,
RDMLRM, 6-157

Transfers

defining, *RDORM*, 9-180

deleting, *RDORM*, 9-251

Translating requirements analysis,
DESIGN, 2-3

Transportability, *GUSQL*, 2-48

for data retrieval, *GUSQL*, 3-74

of cursor, *GUSQL*, 4-21

of error-handling technique, *GUSQL*,
11-40

of parameter declaration, *GUSQL*,
10-93

using context file, *GUSQL*, 8-11

Trigger, *SQLRM*, 3-19

altering, *DESIGN*, 5-20

and referential integrity, *DESIGN*,
3-37

creating, *DESIGN*, 3-39; *SQLRM*,
6-202 to 6-218

defining

DEFINE TRIGGER statement,
RDORM, 9-204

deleting, *DESIGN*, 5-20; *RDORM*,
9-253; *SQLRM*, 6-322

deleting data, *GURRR*, 6-8

displaying, *GUSQL*, 1-9

dropping, *DESIGN*, 5-20

modifying, *DESIGN*, 5-20

naming, *SQLRM*, 3-19

SHOW statement, *RDORM*, 9-413

using, *GURRR*, 6-16, 6-17

using to track changes to a database,
DESIGN, 3-38

Truncation

of file name during RMU/BACKUP
command, *RDORM*, 6-33

Tuning

See also Maintenance; Performance
adjusting buffer values, *MAINT*,
16-91

allocation size, *MAINT*, 16-110

buffer size, *MAINT*, 16-89

correcting record fragmentation,
MAINT, 16-93

defined, *TUNING*, 1-1

due to added capacity, *TUNING*, 1-3

Tuning (Cont.)

- due to workload changes, *TUNING*, 1-2
 - flushing buffers, *MAINT*, 16-92
 - general guidelines, *MAINT*, 13-4
 - methodology, *TUNING*, 1-5
 - number of buffers, *MAINT*, 16-91
 - page size, *MAINT*, 16-109
 - process quotas, *MAINT*, 13-10
 - Rdb/VMS logical names, *MAINT*, 16-141, 16-141t
 - what to tune, *TUNING*, 1-10
 - when to tune, *TUNING*, 1-1
 - working set parameters, *MAINT*, 13-3
- Two-phase commit protocol, *GUSQL*, 2-2; *DIST_TRANS*, 1-1 to 1-7, 2-6
- commit phase, *DIST_TRANS*, 1-6, 2-12
 - defined, *DIST_TRANS*, 1-1
 - designing database for, *DIST_TRANS*, 3-1 to 3-9
 - prepare phase, *DIST_TRANS*, 1-6, 2-7
 - using with callable RDO, *DIST_TRANS*, 5-1
 - using with interactive RDO, *DIST_TRANS*, 5-1
 - using with RDBPRE, *DIST_TRANS*, 5-1 to 5-18
 - using with RDML, *DIST_TRANS*, 5-2
 - using with SQL, *DIST_TRANS*, 4-1 to 4-24
 - when to use, *DIST_TRANS*, 1-2
- Two-phase commit syntax
- START_TRANSACTION* statement, *RDORM*, 9-431

U

- UNCORRUPT* command (RdbALTER), *MAINT*, 7-3; *RDORM*, 7-39
- Uniform page format, *DESIGN*, 4-5
- UNION* operator, *GUSQL*, 3-59

- UNIQUE* expression, *RDMLRM*, 3-38
- Uniqueness, *DESIGN*, 3-42
- UNIQUE* operator, *SQLRM*, 3-97
- Unique value
 - retrieving
 - See *DISTINCT* keyword
- Unloading
 - data
 - using RMU/UNLOAD command, *MAINT*, 2-37
 - databases, *DESIGN*, 5-26; *RDORM*, 6-158
 - relations, *RDORM*, 6-158
 - views, *RDORM*, 6-158
- Unresolved database transactions
 - resolving, *RDORM*, 6-100
- Unresolved distributed transaction, *DIST_TRANS*, 6-1
 - completing, *DIST_TRANS*, 6-4
 - completing in corrupt database, *DIST_TRANS*, 6-5
- Unresolved transaction
 - resolving, *RDORM*, 6-103
- Unsupported statements
 - See *Obsolete statements*
- UPDATE* statement, *GUSQL*, 5-12; *SQLRM*, 6-560
 - in SQL module, *GUSQL*, 7-19
- Updating
 - data, *GUSQL*, 5-12; *GURRR*, 6-4
 - database files
 - using dictionary definitions, *DESIGN*, 7-22
 - databases, *GURRR*, 6-1; *SQLRM*, 6-414
 - data dictionary definitions, *SQLRM*, 6-424
 - dictionary, *DESIGN*, 7-15
 - using SQL, *DESIGN*, 7-17
 - problems, *MAINT*, 16-16
 - with *START_STREAM*, *GURRR*, 6-8
- User identification code
 - accessing databases, *RDORM*, 9-44

USER keyword, *SQLRM*, 3-47
 Users
 default number of simultaneous users,
 DESIGN, 3-7
 User-stored records, *MAINT*, 11-35
 User-supplied name, *SQLRM*, 3-3
 aliases, *SQLRM*, 3-15
 authorization identifiers, *SQLRM*,
 3-10
 parameters, *SQLRM*, 3-20
 USING CONTEXT clause, *SQLRM*, 5-2
 Using Rdb/VMS in programs, *GURRR*,
 1-11
 Using Rdb/VMS with other Digital
 products, *INTRO*, 1-34
 Using the RMU/DUMP/AREA command,
MAINT, 11-15e
 Using the RMU/DUMP
 /LAREA=RDB\$AIP command,
 MAINT, 11-19e
 Using the RMU/DUMP/LAREA
 command, *MAINT*, 11-17e
 Using the RMU/DUMP/SHOT
 command, *MAINT*, 11-22e
 Using VAXclusters, *MAINT*, 18-9

V

Validation, *RDORM*, 5-11
 of constraint, *GUSQL*, 11-28
 using index, *MAINT*, 16-84
 VALID IF clause, *RDORM*, 5-11
 Value expression, *GUSQL*, 3-5;
 RDMLRM, 2-2t; *RDORM*, 3-2;
 GURRR, 3-6; *SQLRM*, 3-66
 arithmetic, *SQLRM*, 3-75
 arithmetic expressions, *RDORM*,
 3-3, 3-21
 calculating value, *RDMLRM*, 2-1
 comparing, *GUSQL*, 3-22
 concatenated expressions, *RDORM*,
 3-4, 3-25
 context variable, *RDORM*, 3-3
 database key, *RDORM*, 3-4, 3-32

Value expression (Cont.)
 definition, *RDORM*, 3-1
 field name, *RDORM*, 3-3
 FIRST FROM expression, *RDORM*,
 3-3, 3-31
 host language variable, *RDORM*, 3-2
 literals, *RDORM*, 3-6
 missing value, *RDORM*, 3-3
 numeric string, *RDORM*, 3-3
 quoted string, *RDORM*, 3-3
 RDB\$LENGTH, *RDORM*, 3-34
 RDB\$MISSING, *RDORM*, 3-26
 RDB\$VALUE, *RDORM*, 3-34
 segmented string, *RDORM*, 3-4,
 3-34
 statistical expression, *RDORM*, 3-3
 unary minus, *RDORM*, 3-4
 VARBYTE data type
 See BYTE VARYING data type
 VARCHAR data type, *DESIGN*, 3-19;
 SQLRM, 3-34
 Variable, *SQLRM*, 6-407
 See also Parameter
 comparing literal to numeric,
 GUSQL, 3-23
 environment
 in SQL/Services, *SQLSRV*, B-1
 VAXcluster, *MAINT*, 18-9
 CI-only
 reducing loss of database access,
 MAINT, 18-20
 common system disk, *MAINT*, 18-9
 concepts, *MAINT*, 18-2
 converting from single-node, *MAINT*,
 18-26
 creating MF_PERSONNEL database,
 MAINT, 18-20
 creating the database, *MAINT*,
 18-19
 database access, *MAINT*, 18-19
 dictionary requirements, *MAINT*,
 18-18
 dual-ported MSCP-served disks,
 MAINT, 18-7

VAXcluster (Cont.)

- environment overview, *MAINT*, 1-16
 - exporting database, *MAINT*, 18-31
 - failure, *MAINT*, 18-16
 - file naming conventions, *MAINT*, 18-8
 - file placement, *MAINT*, 18-16
 - importing database, *MAINT*, 18-31
 - journal file placement, *MAINT*, 9-6
 - Local Area VAXcluster
 - AUTOGEN parameters, *MAINT*, 18-31
 - monitoring the database, *MAINT*, 18-30
 - monitor processes, *MAINT*, 18-9
 - performance in a, *MAINT*, 13-9
 - recovery, *MAINT*, 18-19, 18-29
 - shareable images, *MAINT*, 18-9
 - sharing disk files, *MAINT*, 18-6
 - specifying maximum node number, *MAINT*, 18-11
 - starting distributed transaction in, *DIST_TRANS*, 3-6
 - SYSUAF.DAT, *MAINT*, 18-9
 - terms, *MAINT*, 18-2
- ### VAX Data Distributor
- DEFINE TRANSFER statement, *RDORM*, 9-180
 - DELETE SCHEDULE statement, *RDORM*, 9-247
 - DELETE TRANSFER statement, *RDORM*, 9-251
 - REINITIALIZE TRANSFER statement, *RDORM*, 9-357
 - SHOW TRANSFER statement, *RDORM*, 9-409
 - START TRANSFER statement, *RDORM*, 9-453
 - STOP TRANSFER statement, *RDORM*, 9-456
 - summary of statements, *RDORM*, 2-9; *SQLRM*, 2-7
- ### VAX Data Distributor error messages
- explanation files, *RDORM*, B-1

VAX DATATRIEVE

See DATATRIEVE

VAX DATATRIEVE formatting clause, *SQLRM*, 3-49

VAX DBMS

in distributed transaction, *DIST_TRANS*, 4-2

VAX Language-Sensitive Editor

See LSE

VAX Performance Advisor (VPA), *MAINT*, 13-7, 16-132

VAX Software Performance Monitor (SPM), *MAINT*, 13-7, 16-132

VAX Text Processing Utility

See VAXTPU editor

VAXTPU editor

editing in interactive SQL, *SQLRM*, 6-328

using with EDIT statement, *GUSQL*, 1-24

VERIFY command (RdbALTER), *MAINT*, 7-23; *RDORM*, 7-41

Verifying database integrity, *MAINT*, 6-4; *RDORM*, 6-161

full, *MAINT*, 6-4

Verifying databases

after alterations in RdbALTER, *MAINT*, 7-23

after using RdbALTER, *MAINT*, 7-16

Verifying protection, *DESIGN*, 6-34

Version number

displaying information about
SHOW VERSIONS statement, *RDORM*, 9-416

VIDA

differences from current Rdb/VMS version, *SQLRM*, E-1

VIDA database

using with SQL module processor, *GUSQL*, 7-28

View, *GURRR*, 5-1 to 5-6

benefits of using, *GURRR*, 5-1

View (Cont.)

- compared to
 - cursor, *GUSQL*, 4-6
- creating, *DESIGN*, 3-46; *SQLRM*, 6-219 to 6-230
- defining, *RDORM*, 9-219; *GURRR*, 5-2, 5-4
- defining protection for, *DESIGN*, 6-22
- deleting, *DESIGN*, 5-21; *RDORM*, 9-255; *SQLRM*, 6-325
- displaying, *GUSQL*, 1-12
- in retrieving data, *GUSQL*, 3-67
- naming, *SQLRM*, 3-12
- record selection expressions, *RDORM*, 4-15
- specifying default protection for, *RDORM*, 9-144
- unloading, *RDORM*, 6-158

Virtual memory statistics, *MAINT*, 15-22

VIRTUALPAGECNT parameter values, *MAINT*, 16-137

VMS Debugger, *GUSQL*, 9-6; *GURRR*, 11-24

VMS privilege

- TMPMBX privilege, *MAINT*, 16-141

VMS security audit journal, *MAINT*, 4-20

- loading into database, *MAINT*, 4-25; *RDORM*, 6-70

Volume tables, *DESIGN*, 2-14

VPA

- See VAX Performance Advisor

W

- Wait interval, *DIST TRANS*, 4-4
- WAIT option, *GUSQL*, 2-23
- WHENEVER statement, *GUSQL*, 11-5, 11-16; *SQLRM*, 6-564
- WHERE clause, *GUSQL*, 3-22, 5-13
 - specifying alternative condition, *GUSQL*, 3-49
- Wildcard character (*), *GURRR*, 3-7

- WITH clause, *RDMLRM*, 4-50
 - of record selection expression, *RDORM*, 4-8
- Workload information
 - collecting with DECTrace, *MAINT*, 15-60
- WSDEFAULT parameter values, *MAINT*, 16-139
- WSEXTENT parameter values, *MAINT*, 16-139
- WSMAX parameter values, *MAINT*, 16-137
- WSQUOTA parameter values, *MAINT*, 16-139

Y

- YESTERDAY string literal
 - translation of, *RDORM*, 3-8; *SQLRM*, 3-44

How to Order Additional Documentation

Technical Support

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

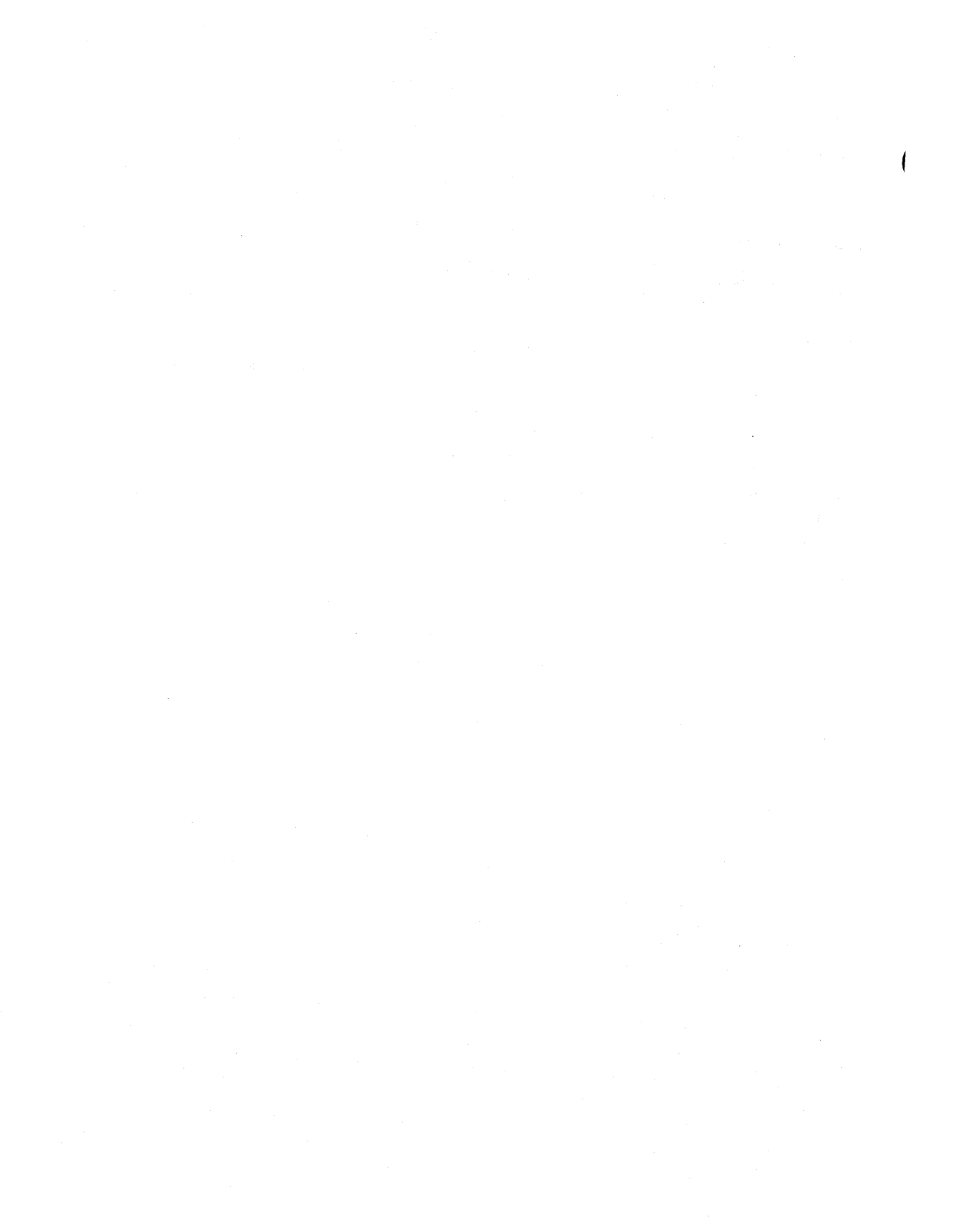
Electronic Orders

To place an order at the Electronic Store, dial 800-DEC-DEMO (800-332-3366) using a 1200- or 2400-baud modem. If you need assistance using the Electronic Store, call 800-DIGITAL (800-344-4825).

Telephone and Direct Mail Orders

Your Location	Call	Contact
Continental USA, Alaska, or Hawaii	800-DIGITAL	Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061
Puerto Rico	809-754-7575	Local Digital subsidiary
Canada	800-267-6215	Digital Equipment of Canada Attn: DECdirect Operations KAO2/2 P.O. Box 13000 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6
International	_____	Local Digital subsidiary or approved distributor
Internal ¹	_____	USASSB Order Processing - WMO/E15 or U.S. Area Software Supply Business Digital Equipment Corporation Westminster, Massachusetts 01473

¹For internal orders, you must submit an Internal Software Order Form (EN-01740-07).



Reader's Comments

VAX Rdb/VMS
Introduction and Master Index
AA-KY66C-TE

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

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Completeness (enough information)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Clarity (easy to understand)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Organization (structure of subject matter)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Figures (useful)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Examples (useful)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Index (ability to find topic)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Page layout (easy to find information)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

I would like to see more/less _____

What I like best about this manual is _____

What I like least about this manual is _____

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

I am using **Version** _____ of the software this manual describes.
Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____
_____ Phone _____

Do Not Tear - Fold Here and Tape

digital



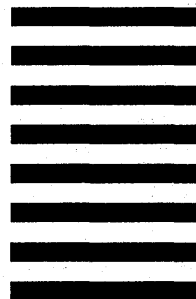
NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

**DIGITAL EQUIPMENT CORPORATION
CORPORATE USER PUBLICATIONS
MRO1-3/L12
P.O. BOX 1001
MARLBOROUGH, MA 01752-9840**



Do Not Tear - Fold Here and Tape