J2X0-2273-01EN

SymfoWARE(R)Server
RDB User's Guide: Database Definition

FUJITSU

Preface

■Purpose

This manual is a user's guide for SymfoWARE Server RDB. The manual explains how to create and define databases.

■Intended reader

This manual is for users who design and define SymfoWARE/RDB databases. Readers should have the following skills and knowledge:

- A general understanding of SymfoWARE/RDB functions and databases
- A working knowledge of applications to which SymfoWARE/RDB is applied
- The ability to develop application programs using C or COBOL
- Working knowledge of Solaris systems or Windows NT systems

■Organization

This manual consists of the following chapters and appendixes.

◆Chapter 1 SymfoWARE/RDB Overview

Provides a general overview of SymfoWARE/RDB functions, databases, and database access methods.

◆Chapter 2 Database Creation

Explains the procedure from designing a SymfoWARE/RDB database up to using it.

◆Chapter 3 Database Definition Alteration and Deletion

Explains how to alter or delete a definition for a database that has already been created.

◆Chapter 4 Storage Structure

Explains the features of data storage structures, which play an important role in determining processing efficiency. This chapter also explains how to allocate database space.

◆Appendix A Quantitative Restrictions

Lists SymfoWARE/RDB limitations.

◆Appendix B Sequential Relationships Among Definition Changes

Indicates basic sequential relationships among changes made to database definitions.

◆Appendix C Operating Environment File Parameters

Lists parameters that can be specified in the operating environment file for tuning the application program operating environment.

◆Appendix D Environment Variables

Explains environment variables for tuning the application program operating environment.

◆Appendix E RDB Command Summary

Lists RDB commands and summarizes the command functions.

◆Appendix F Handling SymfoWARE/RDB Messages

Explains how to reference information about user responses to messages generated by the SymfoWARE/RDB system.

◆Appendix G Exclusive Control Between Application Programs and RDB Commands

Explains exclusive control when an application program and SymfoWARE/RDB command operate simultaneously.

◆Glossary

The glossary defines technical terms used in this manual.

■Reading this manual

The purpose of this manual is to give readers a basic introduction to databases and their creation to make it easier to use SymfoWARE/RDB.

Unless otherwise noted, application programs and SQL statement in this manual are written in C.

■Title Notation of Related Manual

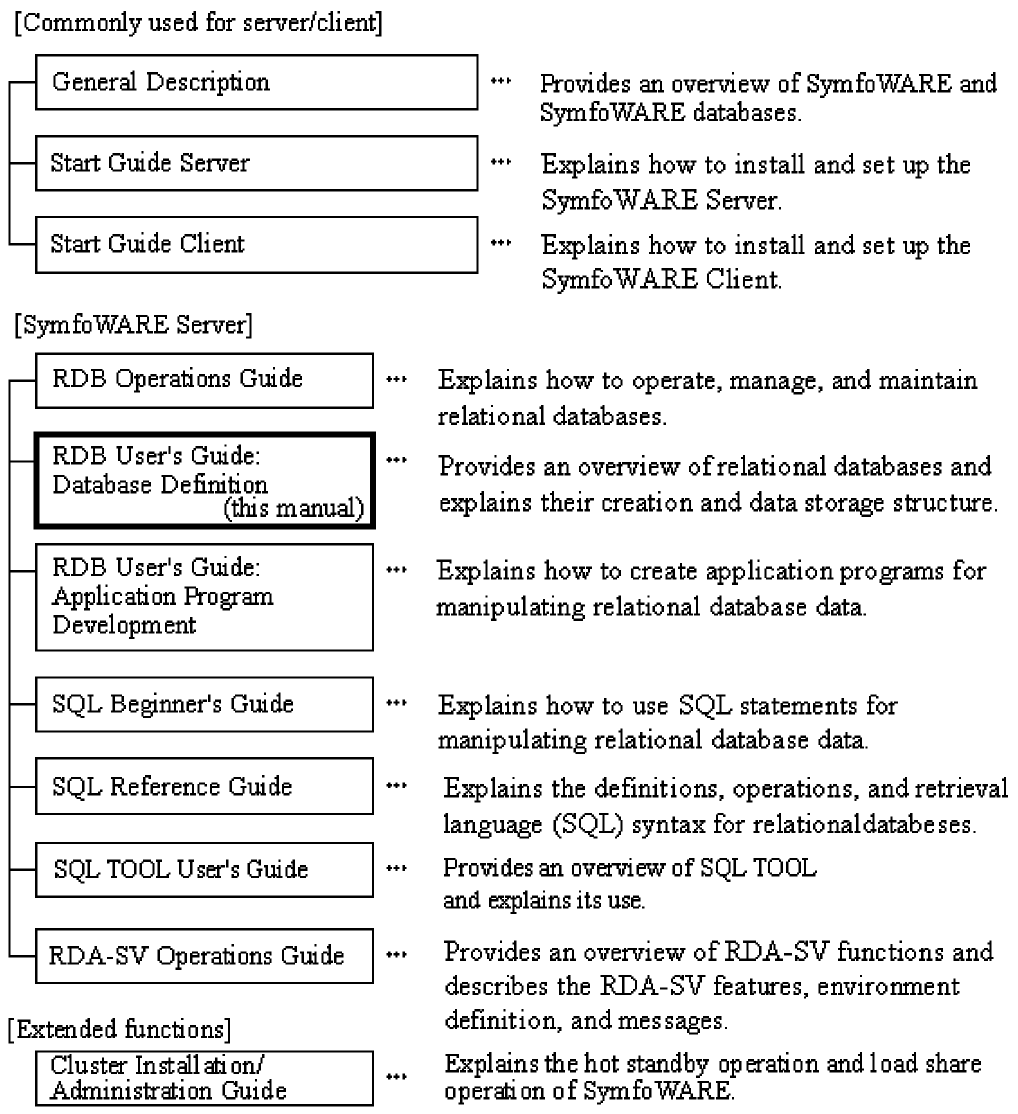
The table below lists the manuals related to this manual and their title notation in this manual.

Notation in this manual	Manual title for each OS	
	Solaris	Windows NT
General Description	FUJITSU SymfoWARE Server General Description	
Start Guide: Client	FUJITSU SymfoWARE Server Start Guide: Client	
Start Guide: Server	–	SymfoWARE Server Start Guide: Server (EE/SE)
RDB Operations Guide	SymfoWARE Server RDB Operations Guide	SymfoWARE Server RDB Operations Guide (EE/SE)
RDB User's Guide: Database Definition (This manual)	FUJITSU SymfoWARE Server RDB User's Guide: Database Definition	
RDB User's Guide: Application Program Development	FUJITSU SymfoWARE Server RDB User's Guide: Application Program Development	
SQL Beginner's Guide	FUJITSU SymfoWARE Server SQL Beginner's Guide	
SQL Reference Guide	FUJITSU SymfoWARE Server SQL Reference Guide	
SQLTOOL User's Guide	FUJITSU SymfoWARE Server SQLTOOL User's Guide	
RDA-SV Operations Guide	SymfoWARE Server RDA-SV Operations Guide	
Cluster Installation/ Administration Guide	SymfoWARE Server Cluster Installation/Administration Guide	–

–: No corresponding manual

■Position of this manual

Manual system and position of this manual in the system



[Commonly used for server/client]

General Description	··· Provides an overview of SymfoWARE and SymfoWARE databases.
Start Guide Server	··· Explains how to install and set up the SymfoWARE Server.
Start Guide Client	··· Explains how to install and set up the SymfoWARE Client.

[SymfoWARE Server]

RDB Operations Guide	··· Explains how to operate, manage, and maintain relational databases.
RDB User's Guide: Database Definition (this manual)	··· Provides an overview of relational databases and explains their creation and data storage structure.
RDB User's Guide: Application Program Development	··· Explains how to create application programs for manipulating relational database data.
SQL Beginner's Guide	··· Explains how to use SQL statements for manipulating relational database data.
SQL Reference Guide	··· Explains the definitions, operations, and retrieval language (SQL) syntax for relationaldatabeses.
SQL TOOL User's Guide	··· Provides an overview of SQL TOOL and explains its use.
RDA-SV Operations Guide	··· Provides an overview of RDA-SV functions and describes the RDA-SV features, environment definition, and messages.

[Extended functions]

Cluster Installation/ Administration Guide	··· Explains the hot standby operation and load share operation of SymfoWARE.

Besides the preceding manuals, SymfoWARE provides an online manual.

◆Command syntax

UNIX

The man command is used to display the syntax of RDB commands.

For details on the man command, refer to AnswerBook2 of the Reference Manual Collection.

The copyright of the online manual is the property of UNIX System Laboratories, Inc. and Fujitsu. Follow the items in the written contract to use the product properly.

Windows NT/2000/XP

The command syntax is included in the Windows NT/2000/XP online help.

◆Action in response to displayed messages

UNIX

The rdbprtmsg command (RDB command) gives the meaning and user response for each displayed message.

Windows NT/2000/XP

Action in response to displayed messages is included in the Windows NT/2000/XP online help.

◆Related manuals

The related manuals are as follows:

- Reference Manual Collection of AnswerBook 2
- Fujitsu COBOL User's Guide for Windows
- COBOL85 User's Guide
- Fujitsu COBOL Language Reference

■Comments on this manual

◆Products covered by this manual

UNIX

- SymfoWARE Server Enterprise Edition 5.0 or later
- SymfoWARE Server Hot Standby Option 5.0 or later

Windows NT/2000/XP

- SymfoWARE Server Enterprise Edition V5.0L10 or later

◆Operating systems supporting SymfoWARE/RDB

- Solaris
- Microsoft(R) Windows NT(R) Server, Enterprise Edition
- Microsoft(R) Windows NT(R) Server network operating system
- Microsoft(R) Windows NT(R) Workstation operating system
- Microsoft(R) Windows NT(R) Server, Terminal Server Edition
- Microsoft(R) Windows(R) 2000 Professional operating system
- Microsoft(R) Windows(R) 2000 Server operating system
- Microsoft(R) Windows(R) 2000 Advanced Server operating system
- Microsoft(R) Windows(R) XP Professional

◆UNIX release version

This system conforms to UNIX System Rel 4.2MP.

◆About the drawings

The drawings covering SymfoWARE/RDB printing in this manual are intended to give the reader only a rough idea of how the printing process works.

◆About the explanatory models

The sample databases in this manual are modeled mainly from inventory control databases of retailers. The database designs and data contents are fictitious and not based on facts.

◆Abbreviated names

This manual uses the following abbreviated names:

Abbreviation	Full name
Windows 95/98/Me	Microsoft(R) Windows(R) 95 operating system Microsoft(R) Windows(R) 98 operating system Microsoft(R) Windows(R) 98 Second Edition Microsoft(R) Windows(R) Millennium Edition
Windows	Microsoft(R) Windows(R) 95 operating system Microsoft(R) Windows(R) 98 operating system Microsoft(R) Windows(R) 98 Second Edition Microsoft(R) Windows(R) Millennium Edition Microsoft(R) Windows (R) XP Professional. Microsoft(R) Windows (R) XP Home Edition. Microsoft(R) Windows NT(R) Server network operating system Microsoft(R) Windows NT(R) Workstation operating system Microsoft(R) Windows NT(R) Server, Enterprise Edition Microsoft(R) Windows NT(R) Server, Terminal Server Edition Microsoft(R) Windows(R) 2000 Professional operating system Microsoft(R) Windows(R) 2000 Server operating system, and Microsoft(R) Windows(R) 2000 Advanced Server operating system
Windows NT	Microsoft(R) Windows NT(R) Server network operating system, Microsoft(R) Windows NT(R) Workstation operating system, Microsoft(R) Windows NT(R) Server, Enterprise Edition, and Microsoft(R) Windows NT(R) Server, Terminal Server Edition
Windows 2000	Microsoft(R) Windows (R) 2000 Professional operating system, Microsoft(R) Windows (R) 2000 Server operating system, and Microsoft(R) Windows (R) 2000 Advanced Server operating system
Windows XP	Microsoft(R) Windows (R) XP Professiona, and Microsoft(R) Windows (R) XP Home Edition
Windows NT/2000/XP	Microsoft(R) Windows NT(R) Server network operating system, Microsoft(R) Windows NT(R) Workstation operating system, Microsoft(R) Windows NT(R) Server, Enterprise Edition, Microsoft(R) Windows NT(R) Server, Terminal Server Edition, Microsoft(R) Windows (R) 2000 Professinal operating system, Microsoft(R) Windows (R) 2000 Server operating system, Microsoft(R) Windows (R) 2000 Advanced Server operating system, and Microsoft(R) Windows (R) XP Professional
MS-DOS	Microsoft(R) MS-DOS(R) 5.0/V or later, and Microsoft(R) MS-DOS(R) 6.2/V or later
SymfoWARE Server	SymfoWARE Server Enterprise Edition

July 2002

Chapter 1 SymfoWARE/RDB Overview

SymfoWARE/RDB provides functions for creating a database, managing a database, and manipulating database data. Before creating a database, the user must design the database structure and define the database based on this database structure design specifications. Then, the user must generate the database based on this database definition. Database management is required for checking database usage conditions and handling database damage. Structured query language (SQL) statements are used to access data within the database.

SymfoWARE/RDB provides functions for building a flexible client server system that includes the networks between systems.

This chapter covers the following topics:

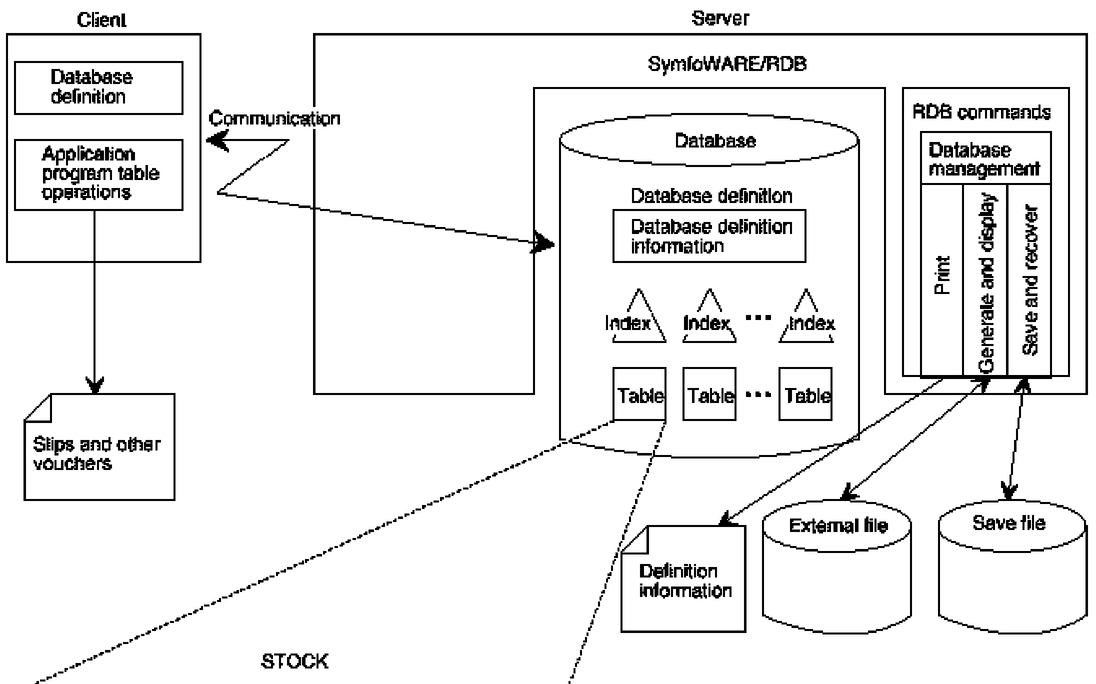
1.1 Overview of SymfoWARE/RDB Functions

SymfoWARE/RDB is a relational database processing system that represents data in table format and processes those tables. The functions of SymfoWARE/RDB can be broadly divided into functions for:

· Defining table formats (database definition)
· Maintaining and managing databases (database management)
· Manipulating tables (table manipulation)

Figure: SymfoWARE/RDB functions configuration shows the configuration of SymfoWARE/RDB functions.

[Figure: SymfoWARE/RDB functions configuration]



STOCK

ITMNO	PRODUCT	STOCKQTY	WHCODE
110	TELEVISION	85	2
111	TELEVISION	90	2
123	REFRIGERATOR	60	1
124	REFRIGERATOR	75	1
140	CASSETTE DECK	120	2
212	TELEVISION	0	2
215	VIDEO CASSETTE PLAYER	5	2
226	REFRIGERATOR	8	1
227	REFRIGERATOR	15	1
240	CASSETTE DECK	25	2
243	CASSETTE DECK	14	2
351	CASSETTE TAPE	2500	2

■Functions for defining table formats (database definition)

To create a database, first define the table formats. 2.2 "Designing a Database," explains the kinds of formats used for tables. RDB commands are used to execute database definitions. For information about how to use actual RDB commands to define a database, see Chapter 2 "Database Creation."

■Functions for maintaining and managing databases (database management)

SymfoWARE/RDB has functions for creating, saving, restoring, maintaining, and managing a database. These functions are invoked by executing specific RDB commands.

For descriptions of the database maintenance and management functions and information about how to execute the functions, refer to the "RDB Operations Guide."

■Functions for manipulating tables (table manipulation)

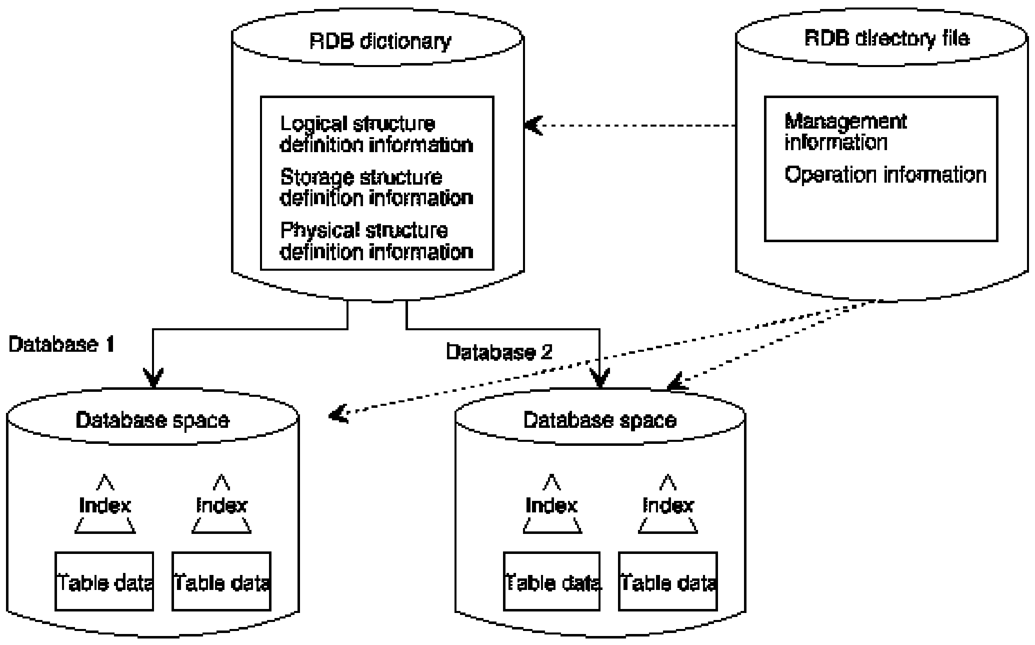
Data manipulation SQL statements are used to insert, alter, delete, and reference data in tables. These SQL statements are used within application programs. For information about how to develop application programs that use data manipulation SQL statements, refer to the "RDB User's Guide: Application Program Development."

For information about how to use data manipulation SQL statements, refer to the "SQL Beginner's Guide."

1.2 Overview of the SymfoWARE/RDB Database Configurat ion

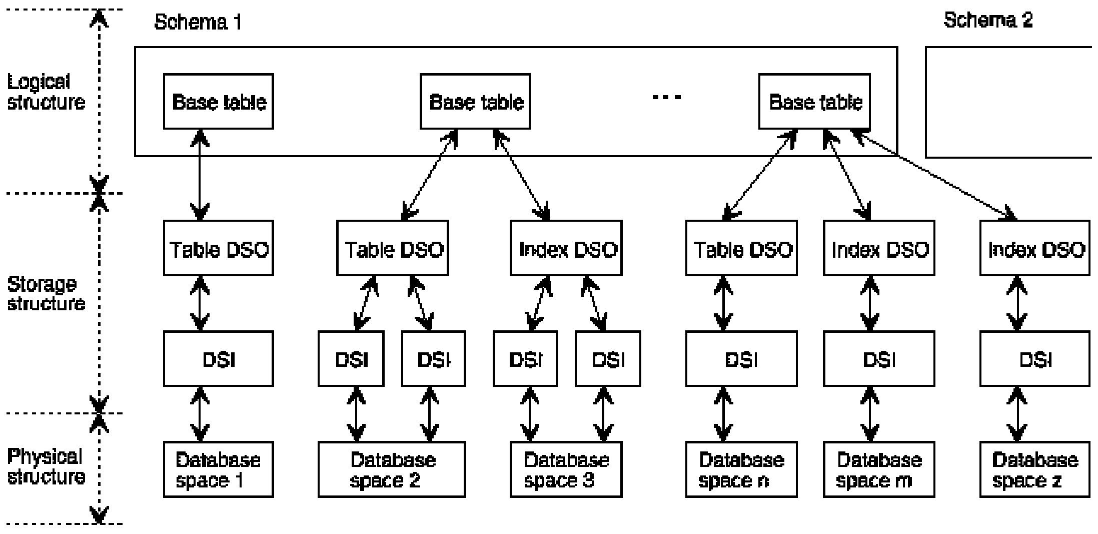
As Figure: SymfoWARE/RDB database configuration shows, a SymfoWARE/RDB database consists of multiple databases and an RDB dictionary and RDB directory file for managing them. The database logical structure, storage structure, and physical structure definition information is stored in the RDB dictionary. Base tables, which are the database data, and indexes are stored in database spaces.

[Figure: SymfoWARE/RDB database configuration]



In addition, as Figure: Relationship of logical, storage, and physical structures within databases shows, each database consists of schemas, base tables, data structure organizations (DSOs), data structure instances (DSIs), and databases spaces. These items are the basic elements of a three-tier hierarchy composed of the logical structure, storage structure, and physical structure.

[Figure: Relationship of logical, storage, and physical structures within databases]



1.2.1 Physical structure

The physical structure consists of database spaces.

■Database space

Under UNIX, a database space is defined on a raw device created on a magnetic disk; under Windows NT/2000/XP, a database space is defined in a local file created on a magnetic disk. A SymfoWARE/RDB system enables multiple database spaces to be defined so that the hard disk I-O load balance can be adjusted. In addition, the base table data or index data of a single schema can be divided and stored in multiple database spaces.

In SymfoWARE/RDB, database spaces become storage structures that enable resources to be managed. Careful consideration is given to processing efficiency, storage efficiency, and operation. The following two functions can be used primarily:

Multi-database space:
A large-scale database can be built by allocating one table or index in multiple database spaces.

Split table operation:
When a single table is split into multiple parts based on specific rules, each subdivision unit can operate independently. This function enables independent creation, update, backup, and recovery in parallel for each subdivision unit of a large-scale database.

1.2.2 Logical structure

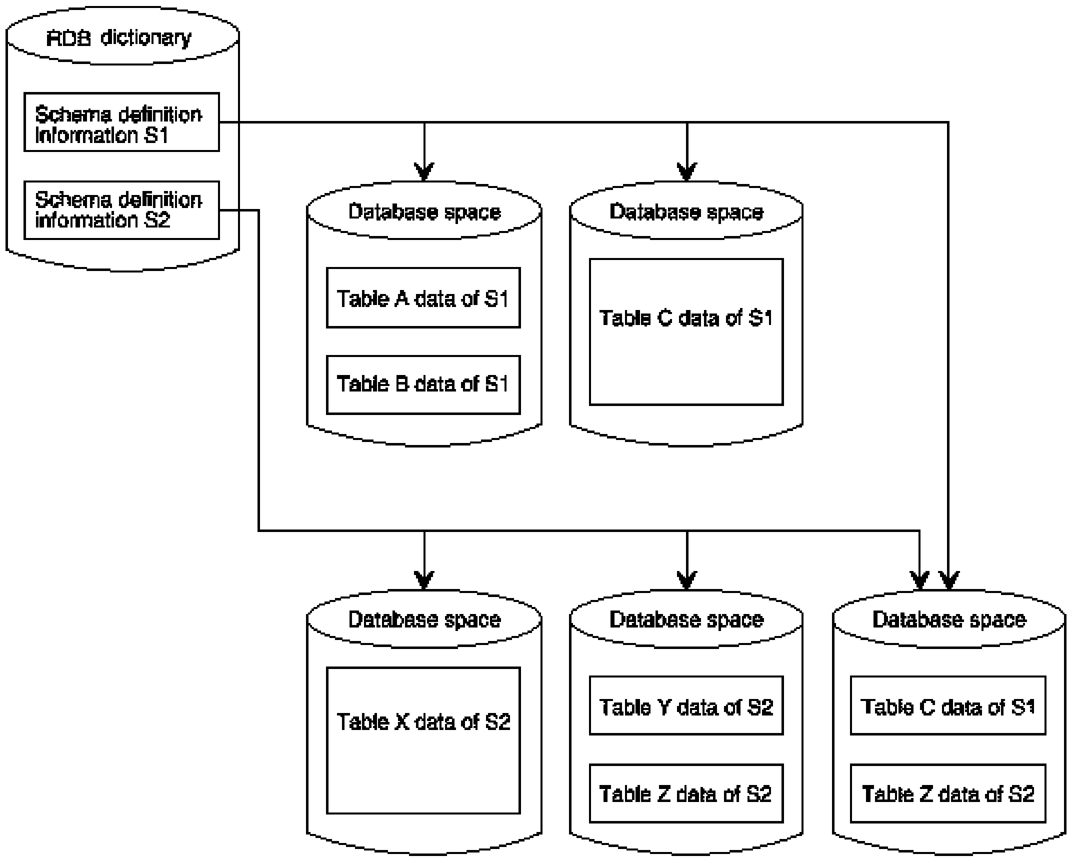
The logical structure consists of schemas and base tables, the elements of the schema.

■Schema

A schema consists of table data and table definition data. The user must carefully consider the following aspects when determining the kinds of base tables that are to form a schema. Consider the applications that are to use the database, the contents of the data to be processed, and the data processing methods. The schema configuration is defined by schema definition statements. These schema definition statements define the following items:

· Schema name
· Schema components
  - Base table name and format
    Schema definition data is entered in the RDB dictionary. Base table data is stored in a database space. Multiple schema definitions can be entered in an RDB dictionary. In addition, base table data belonging to a single schema can be stored in a single database space. Alternatively, base table data can be divided in terms of individual base tables and stored in multiple database spaces.
    Figure: Example of correspondence between schemas and database spaces is an example showing multiple schemas being stored in multiple database spaces.

[Figure: Example of correspondence between schemas and database spaces]



■Base table

A base table consists of columns and rows. Figure: Base table format example is a base table format example. In this figure, one row consists of the data for one product. The data of a single row consists of several columns. A column corresponds to a data item. The data for one product (one row) consists of the four data items (columns): ITMNO, PRODUCT, STOCKQTY, and WHCODE.

The base table configuration is defined by schema definition statements. Table definitions in a schema definition define the data items that form each base table. A table definition defines the following items:

- Table name
- Column
- Table constraint

◆Table name

A table name is the name assigned to each table.

The table name is used to specify the table to be the object of a data manipulation. The table name is also used when adding or deleting a table definition. This table definition specifies the table definition information subject to deletion or addition processing.

◆Column

A column definition contains the following definitions for a column that forms a base table.

Column name:
A column name is the name assigned to each column. The column name is used to specify the column to be processed by a data manipulation. The column name is also used when altering a schema definition to indicate the column to be altered.

Column data type:
The data of each column has a type, such as character, numeric or data-and-time type.

Column default value:
The column default value defines the value to be set if the column data is omitted when data is inserted or

updated.

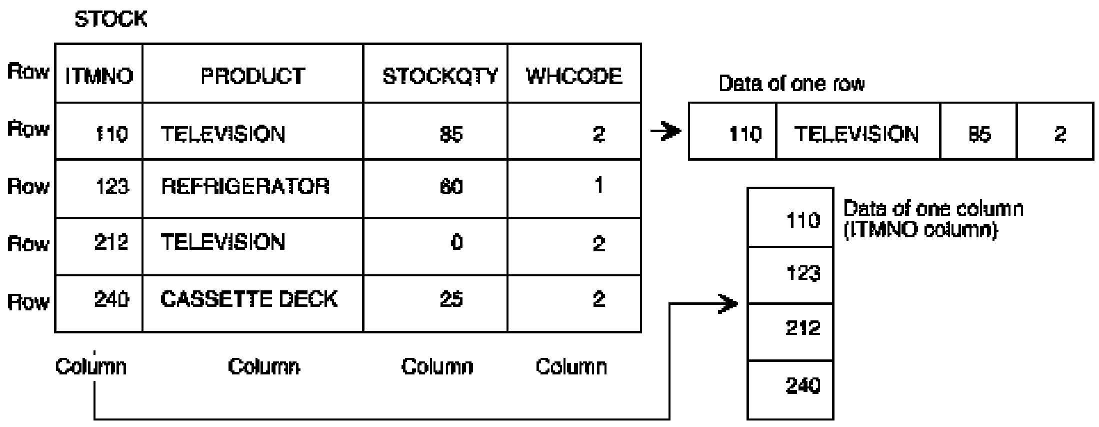
Column constraint:

A column constraint defines a constraint condition on table creation. One such condition may be "each row must have a value stored in the relevant column." Another condition may be "more than one row cannot have the same value in the relevant column."

◆Table constraint

A table constraint enables the user to define whether or not a constraint is to be applied to a table. Such a constraint may be "more than one row cannot have the same values in one or more columns." This constraint is called a unique constraint.

[Figure: Base table format example]



■View

A view is a virtual table for manipulating data. The view does not actually contain any data. A view table is equivalent to the subtable of a base table as shown in Figure: Concept of a view. A view is defined by a view definition. A view definition defines the following items:

Table name:

Defines the name of the view.

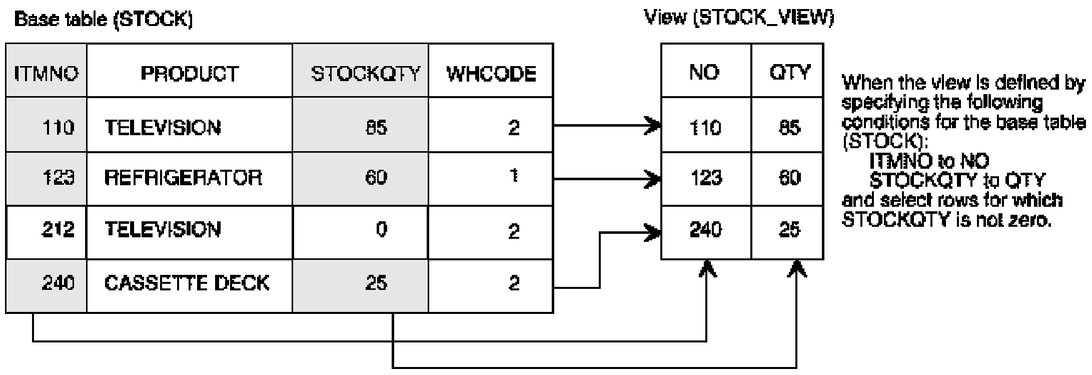
Column names:

Defines the names of the view columns.

View column and row definitions:

Defines which portions of a base table or view are to form a view.

[Figure: Concept of a view]



■Index

An index increases the efficiency of search processing for the database data. A data manipulation is usually accompanied by a data search using column data of a table as the search key. Thus, data search efficiency is an important factor in judging the efficiency of a data manipulation. The user can specify whether or not to create an index for each column of a table. Multiple columns also can be combined and specified as a single index. Searching a column for which no index has been created is less efficient than searching a column having an index. Thus, an index must be created for a column used as a data search key. However, whenever an index is created, additional database capacity is required for the storage. Carefully consider the space required for each index when determining the size of a database space.

Although an index affects the database capacity and data manipulation efficiency, it does not affect the data manipulation. The user need not be concerned with indexes when developing application programs that use SQL statements to manipulate data.

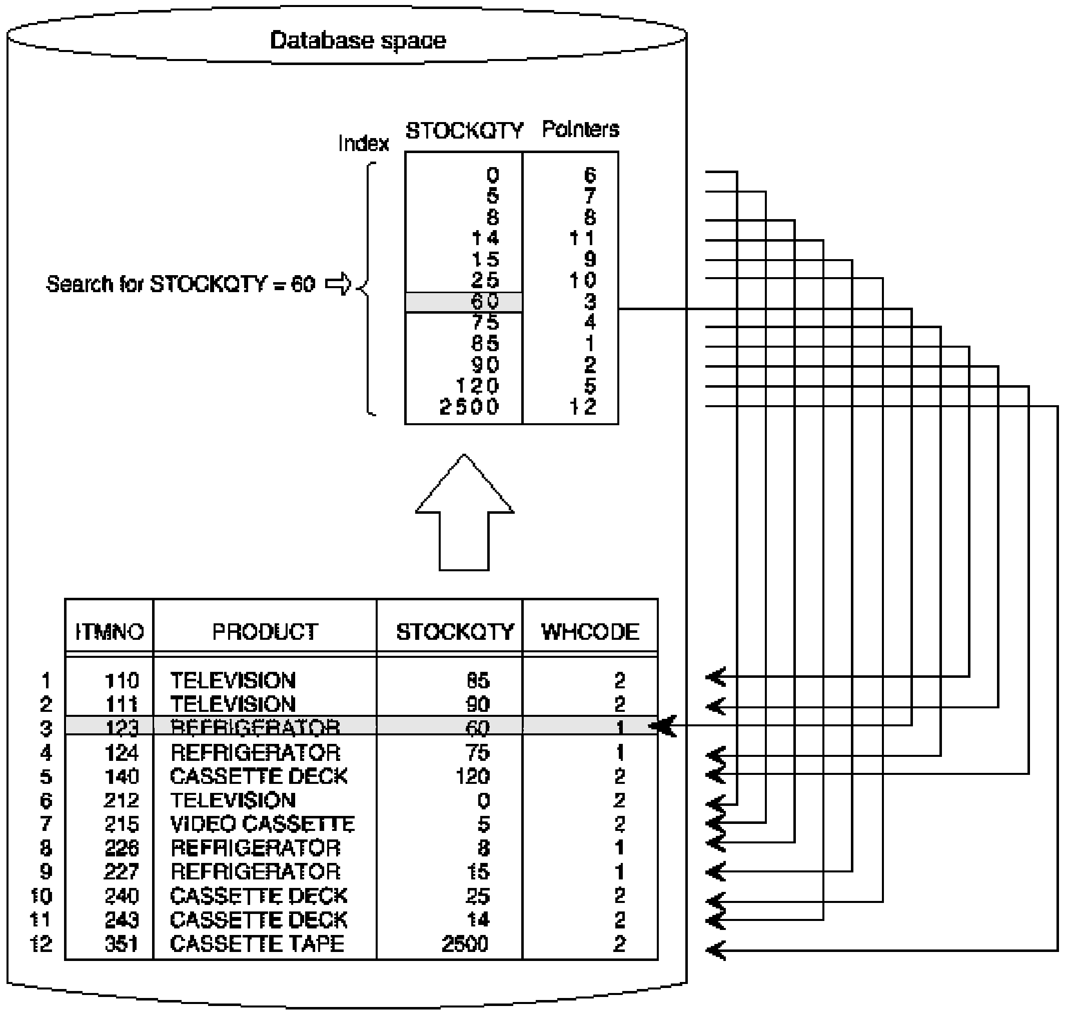
An index is defined by a storage structure definition statement.

An index is defined for a column of a base table. An index cannot be defined for a view.

An index is defined after the schema is defined and before data is stored in the database. However, an index may also be defined after data is stored in the database by database generation or data manipulation. An index is created in a database space following database generation after an index has been defined and when data manipulation (update) is performed.

Figure: Concept of an index shows an index created for the STOCKQTY column of the STOCK table. This figure portrays the concept of an index; it does not accurately represent the database format.

[Figure: Concept of an index]



STOCKQTY	Pointers
0	6
5	7
8	8
14	11
15	9
25	10
60	3
75	4
85	1
90	2
120	5
2500	12

	ITMNO	PRODUCT	STOCKQTY	WHCODE
1	110	TELEVISION	85	2
2	111	TELEVISION	90	2
3	123	REFRIGERATOR	60	1
4	124	REFRIGERATOR	75	1
5	140	CASSETTE DECK	120	2
6	212	TELEVISION	0	2
7	215	VIDEO CASSETTE	5	2
8	226	REFRIGERATOR	8	1
9	227	REFRIGERATOR	15	1
10	240	CASSETTE DECK	25	2
11	243	CASSETTE DECK	14	2
12	351	CASSETTE TAPE	2500	2

The STOCKQTY index consists of STOCKQTY values and pointers indicating the position of the corresponding row in the STOCK table, sorted by STOCKQTY value order. For example, say the user specifies a search for the row of the STOCK table for which the STOCKQTY value is 60. In this case, first the STOCKQTY index is searched. Since the index is arranged by STOCKQTY value order, the search can be performed very quickly. Once the search of the index is completed, the position of the row in the STOCK table is revealed by the corresponding pointer. The specified row can thus be obtained.

1.2.3 Storage structure

The storage structure consists of DSOs and DSIs.

■DSO

A DSO defines the storage structure of the data for a base table. The two types of DSOs are as follows:

- Table
- Index

◆Table DSO

A table DSO defines the type of storage structure for storing data, and, if data is subdivided for storage, the subdivision method.

◆Index DSO

An index DSO defines how the index is created for the table.

■DSI

A DSI defines an area for storing base table data so that it can be allocated in a database space. The two types of DSIs are as follows:

- Table
- Index

◆Table DSI

A table DSI defines an area for storing data so that it can be allocated in a database space.

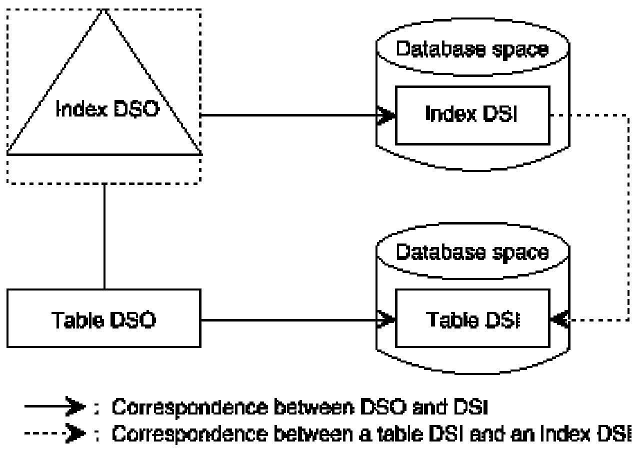
◆Index DSI

An index DSI defines an area for storing index data added to a table so that it can be allocated in a database space.

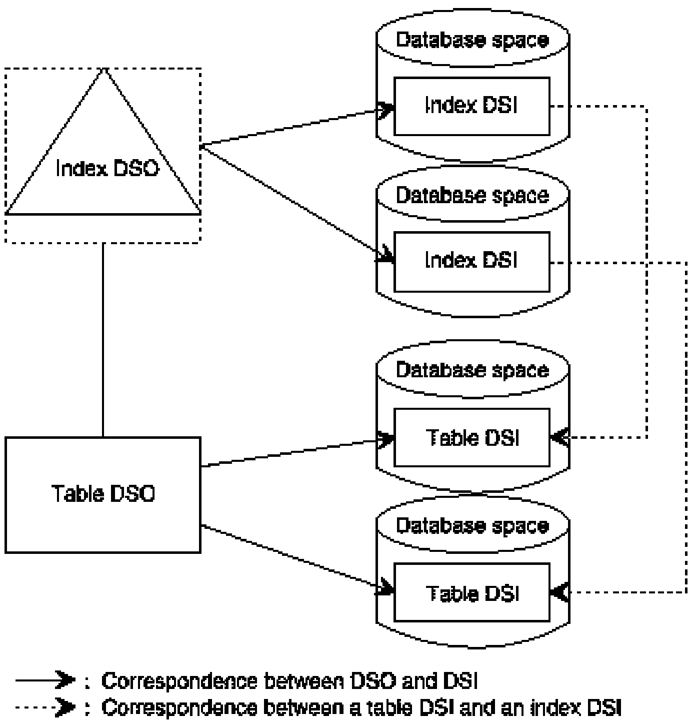
A DSI associates a table or index with a database space.

DSOs and DSIs can be related in either a 1:1 or 1:n correspondence. If a 1:n correspondence exists, table data is subdivided for storage, and rules for splitting the data are defined in the DSO. Figure: Example in which DSOs and DSIs are associated in a 1:1 correspondence is an example in which table data is stored without being split. Figure: Example in which DSOs and DSIs are associated in a 1:n correspondenceis an example in which table data is subdivided for storage. An index DSI is defined for a table DSI. If table data is subdivided for storage, an index DSI must be defined for each table DSI.

[Figure: Example in which DSOs and DSIs are associated in a 1:1 correspondence]

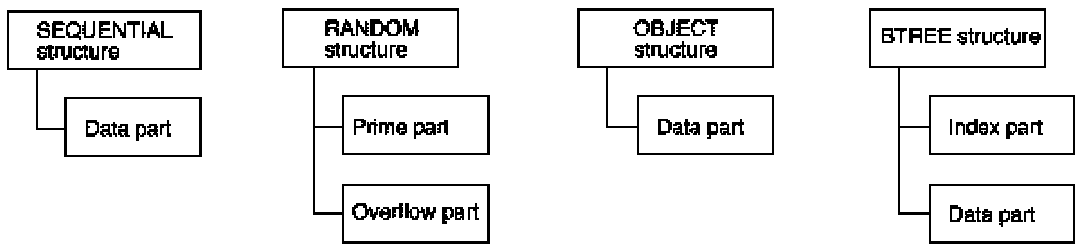

Index DSO
Database space
Index DSI
Database space
Table DSO
Table DSI
: Correspondence between DSO and DSI
: Correspondence between a table DSI and an Index DSI

[Figure: Example in which DSOs and DSIs are associated in a 1:n correspondence]



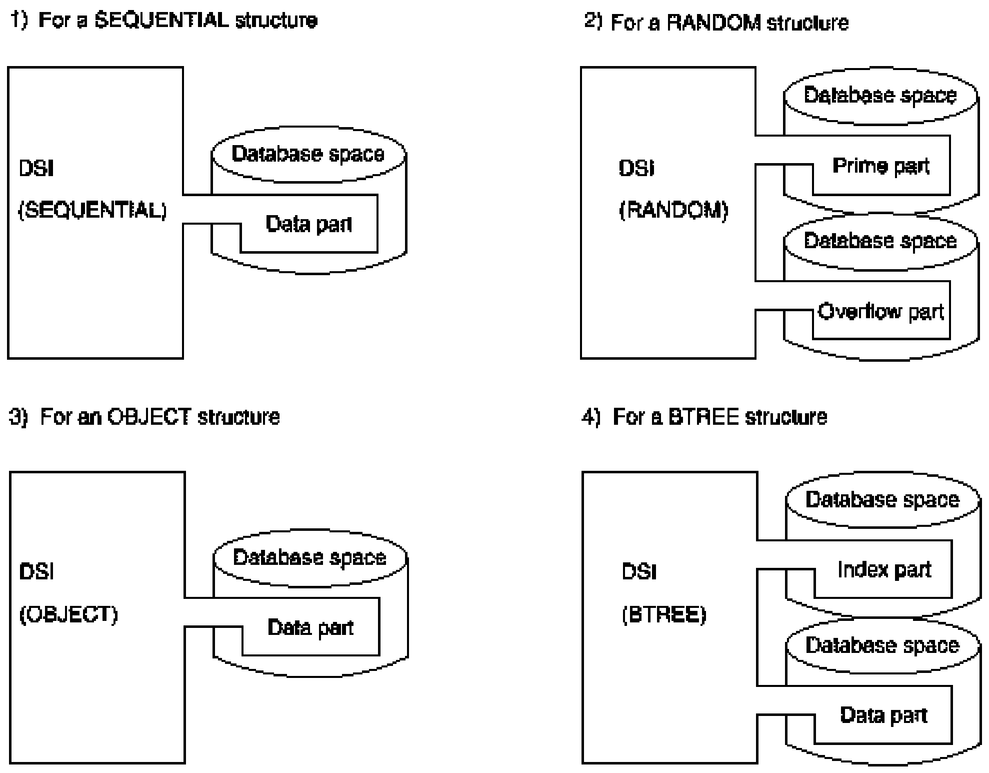
As Figure: Storage structure components shows, the four types of storage structures are SEQUENTIAL, RANDOM, OBJECT, and BTREE. The SEQUENTIAL, RANDOM, and OBJECT structures are used as storage structures for tables. The BTREE structure is used as a storage structure for indexes. Each of these storage structures consists of one or more components as shown in Figure: Storage structure components.

[Figure: Storage structure components]



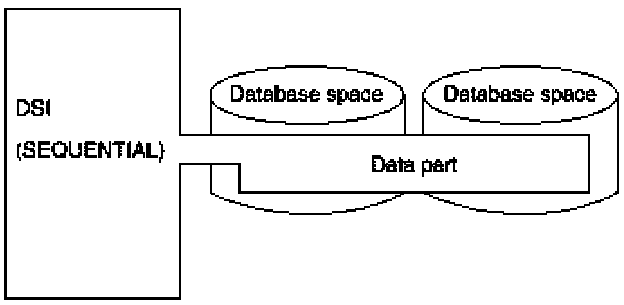
A DSI can consist of multiple database spaces. The configuration of the multiple database spaces is divided into two types. In one configuration, a database space is allocated for each component in the storage structure shown in Figure: Storage structure components. In the other configuration, a database space is allocated to increase the size of each component. Figure: Example in which a database space is allocated to each component is an example in which a database space is allocated to each component. An example of allocating multiple database spaces to increase the size of each component is shown in Figure: Example of allocating multiple database spaces to increase the size of each component.

[Figure: Example in which a database space is allocated to each component]

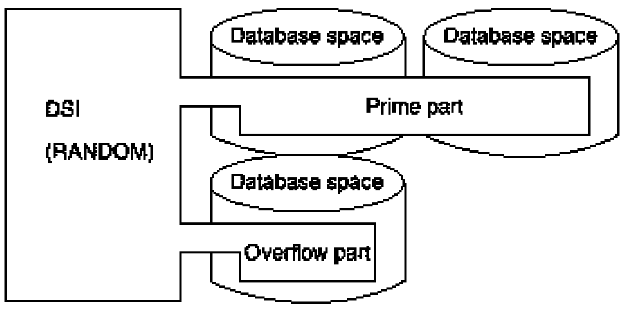


[Figure: Example of allocating multiple database spaces to increase the size of each compon ent]

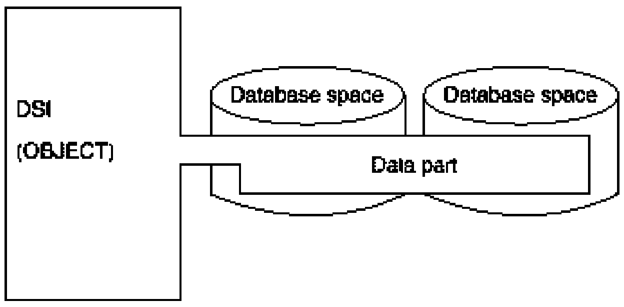
1) For a SEQUENTIAL structure



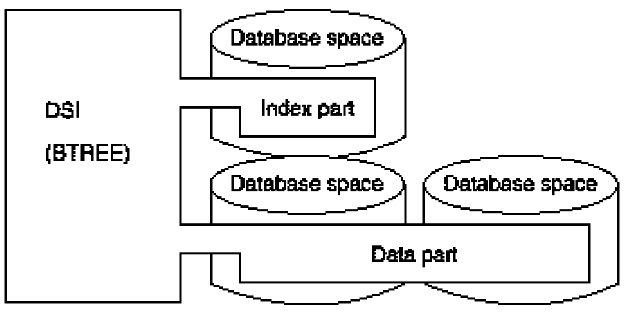
2) For a RANDOM structure



3) For an OBJECT structure



4) For a BTREE structure



1.3 Overview of Database Creation Tasks

A SymfoWARE/RDB database can be created in one of the following two ways:

- Using RDB commands
- Creating a SQL-embedded program

■Using RDB commands

The user can create databases by executing RDB commands at the command prompt of UNIX or Windows NT/2000/XP.

The user can define databases by specifying the file containing various SQLs for defining databases and using the rdbddlex command. The user can also create databases by using the rdbsloader command. This method is suitable for operation in which database logical and storage structures are defined in detail.

For details of how to create databases by using the RDB commands, see Chapter 2 "Database Creation."

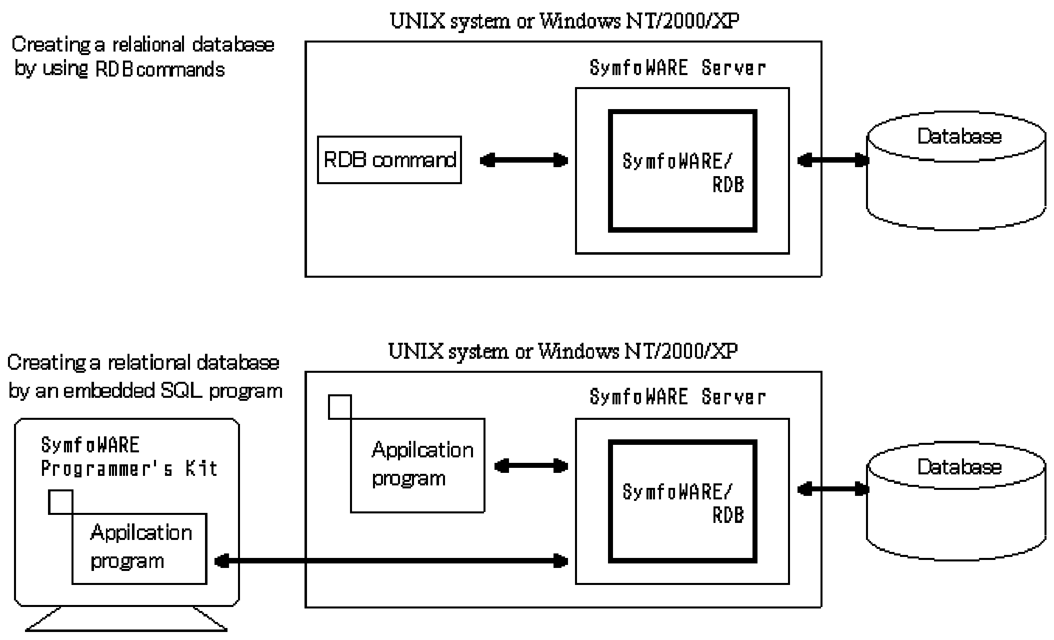
■Creating a SQL-embedded program

Create a SQL-embedded program that uses dynamic SQL to define databases. Create a SQL-embedded program that uses the INSERT statement to create databases.

Details on using a SQL-embedded program to create databases are given in Chapter 2 "Database Creation."

Figure: Overview of tasks involved in database creation provides an overview of the database creation tasks.

[Figure: Overview of tasks involved in database creation]


Creating a relational database by using RDB commands
UNIX system or Windows NT/2000/XP
SymfoWARE Server
RDB command
SymfoWARE/RDB
Database
Creating a relational database by an embedded SQL program
UNIX system or Windows NT/2000/XP
SymfoWARE Server
SymfoWARE Programmer's Kit
Appilcation program
Appilcation program
SymfoWARE/RDB
Database

Chapter 2 Database Creation

This chapter covers procedures ranging from the design and creation of a SymfoWARE/RDB database to database operation.

2.1 Overview of Tasks From Database Design To Operatio n

Database creation tasks are performed after the SymfoWARE/RDB system environment has been created and the SymfoWARE/RDB system has been started. For information about creating the SymfoWARE/RDB system environment and starting the SymfoWARE/RDB system, refer to the "RDB Operations Guide."

See

Refer to the following manuals for more information about the syntax of the SQL statements shown in this manual:

- SQL Reference Guide
- SQL Beginner's Guide

The required procedure for creating and operating a database is as follows.

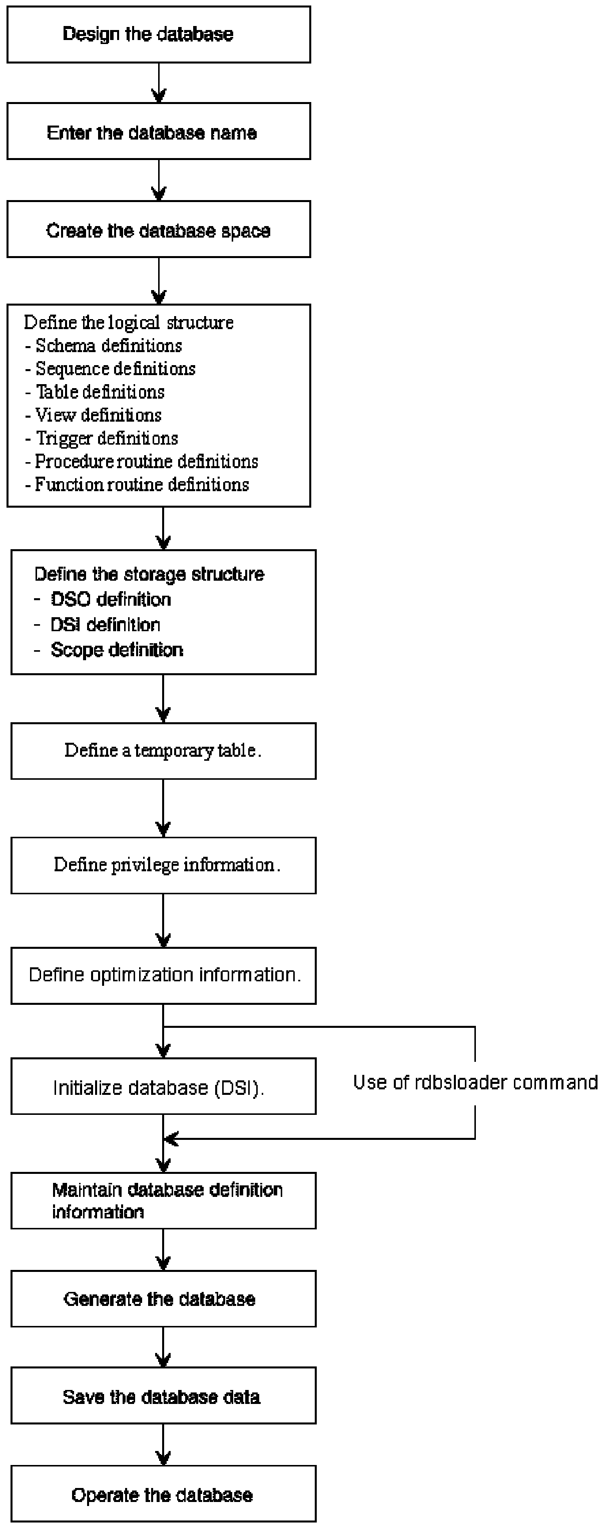
■Database creation and operation

1. Design the database.
2. Enter the database name.
3. Create the database space.
4. Define the logical structures such as schema and table:
   When a logical structure is defined, a simple storage structure can be defined.
5. Define the storage structure (DSO and DSI for table and index).
   To facilitate data retrieval and operation, the storage structure can be defined as follows:
   - Divide a storage table to localize the retrieval range, thus improving retrieval.
   - Divide a storage table to maintain and operate a database without having to stop regular operations during backup and restore if a database failure occurs.
6. Define a temporary table.
7. Define privilege information.
8. Define optimization information.
9. Initialize the database (DSI). If the rdbsloader command was used to create the database, the DSI need not be initialized.
10. To maintain database definition information, save the RDB dictionary data.
11. Generate the database by entering data from external data or an application program.

12. In preparation for using the database, save the database data.
13. Operate the database.

Figure: Procedure from database design to operation shows the flow of tasks from database design to operation.

[Figure: Procedure from database design to operation]


Design the database
Enter the database name
Create the database space
Define the logical structure
- Schema definitions
- Sequence definitions
- Table definitions
- View definitions
- Trigger definitions
- Procedure routine definitions
- Function routine definitions
Define the storage structure
- DSO definition
- DSI definition
- Scope definition
Define a temporary table.
Define privilege information.
Define optimization information.
Initialize database (DSI).
Use of rdbsloader command
Maintain database definition information
Generate the database
Save the database data
Operate the database

2.2 Designing a Database

To create a database, first design the database. When designing the database, be sure to carefully analyze the business applications, types and amounts of data to be processed, and data processing methods.

The database design procedure includes steps for designing tables, attributes, simplified storage structures, and storage structures.

■Designing tables

Design tables that meet the needs of the business application.

■Designing attributes

Determine appropriate data types for designed items, as well as possible column constraints and default values. For details about the attributes that can be used, refer to the "SQL Reference Guide."

■Designing the storage structure

◆Determining the storage structure

From an application program viewpoint, the database is represented in table format. The application program performs data operations according to structured query language (SQL) statements as if it were manipulating rows and columns of data.

The structure for storing the data represented in table format on physical pages is called the storage structure. An appropriate database storage structure is designed according to the analysis data of the access path. To design such a database storage structure, obtain the size of the table from the amount of data and determine the required amount of disk space. The use of a split table is determined by estimating the amount of data to be added and the operation time acceptable for database reorganization.

For details about storage structures, see Chapter 4 "Storage Structure."

- SEQUENTIAL structure
- RANDOM structure
- OBJECT structure
- BTREE structure

◆Estimating the amount of database space required for each storage structure

In this step, allocate database space. Carefully consider the amount of data to be processed and the area access patterns, then allocate database space for each component of the storage structure.

For details about estimating the amount of database space required for each storage structure, see 4.4 "Estimating the Required Amount of Database Space."

■Designing a simplified storage structure

To simplify the process of defining the storage structure, specify the database space for storing data by defining the table or index. SymfoWARE/RDB then automatically defines the storage structure. In this case, the SEQUENTIAL structure is used as the table storage structure. For a table of multimedia data, the SEQUENTIAL or OBJECT structure can be selected as the table storage structure. This process does not allow the use of split storage.

If the storage structure is simplified defined, the DSO and DSI names of the table are automatically assigned from the names generated by the table or index definitions. The data length and allocation are automatically determined at this time.

For simplified definition of the storage structure, the capacity of DSI is dynamically expanded.

The naming prefix, data length, allocation amount, and DSI capacity expansion setting can all be changed by the use of operating environment file parameters. For a table of multimedia data, the storage structure can also be selected. For information about the operating environment file, refer to the "RDB User's Guide: Application Program Development."

■Examples of designing the database

Figure: Contents of the inventory management database shows an inventory management database for a retail store. This inventory management database is used as an example for explaining database design in this section. This database is used to implement the inventory management applications of the retail store. The inventory management database consists of three tables, as follows.

STOCK table:
: Contains information about products handled and quantities of those products in stock

ORDER table:
: Contains information related to products, quantities ordered, and purchase prices for each customer.

COMPANY table:
: Contains information about company names, telephone numbers, and addresses for each customer

The usage examples use the inventory management database mainly to explain database creation and data manipulations. The table and column names above are used in usage examples throughout the remainder of the section. Refer to them as necessary.

■STOCK table

Figure: Contents of the inventory management database a) shows the contents of the STOCK table, which consists of these four columns:

ITMNO:
: Column of code number data assigned to products

PRODUCT:
: Column of product type data

STOCKQTY:
: Column of data indicating the quantity of the product in stock

WHCODE:
: Column of number data for warehouses where the products are stored

■ORDER table

Figure: Contents of the inventory management database b) shows the contents of the ORDER table, which consists of these four columns:

CUSTOMER:
: Column of company number data for customers

PRODNO:
: Column of code numbers assigned to products (corresponds to the ITMNO column of the STOCK table)

PRICE:
: Column of data indicating product purchase prices

ORDERQTY:
: Column of data indicating quantities of products ordered

■COMPANY table

Figure: Contents of the inventory management database c) shows the contents of the COMPANY table, which consists of these four columns:

COMPNO:
: Column of code number data assigned to companies (corresponds to the CUSTOMER column of the ORDER table)

COMPANY:
: Column of company name data

PHONE:
: Column of company telephone number data

ADDRESS:
: Column of company address data

[Figure: Contents of the inventory management database]

a) STOCK table

ITMNO	PRODUCT	STOCKQTY	WHCODE
110	TELEVISION	85	2
111	TELEVISION	90	2
123	REFRIGERATOR	60	1
124	REFRIGERATOR	75	1
140	CASSETTE DECK	120	2
212	TELEVISION	0	2
215	VIDEO CASSETTE PLAYER	5	2
226	REFRIGERATOR	8	1
227	REFRIGERATOR	15	1
240	CASSETTE DECK	25	2
243	CASSETTE DECK	14	2
351	CASSETTE TAPE	2500	2

b) ORDER table

CUSTOMER	PRODNO	PRICE	ORDERQTY
61	123	48000	60
61	124	64000	40
61	140	8000	80
61	215	240000	10
61	240	80000	20
62	110	37500	120
62	226	112500	20
62	351	375	800
63	111	57400	80
63	212	205000	30
63	215	246000	10
71	140	7800	50
71	351	390	600
72	140	7000	70
72	215	210000	10
72	226	105000	20
72	243	84000	10
72	351	350	1000
74	110	39000	120
74	111	54000	120
74	226	117000	20
74	227	140400	10
74	351	390	700

c) COMPANY table

COMPNO	COMPANY	PHONE	ADDRESS
61	IDEA INC.	433-2222	LONDON W.C. 2 ENGLAND 1-2-3
62	ADAM LTD.	731-1111	SANTA CLARA CA USA 7-8-9
63	MOON CO.	143-3333	FIFTH AVENUE NY USA 1-1-1
71	RIVER CO.	344-1212	SAKAI OSAKA JAPAN 4-5-6
72	DRAGON CO.	373-7777	HARAJUKU TOKYO JAPAN 2-3-7
74	FIRST CO.	255-9955	SYDNEY AUSTRALIA 4-16-16

■Column attributes of each table of the inventory management database

Table: Column attributes of each table of the inventory management database shows the column attributes of each table.

[Table: Column attributes of each table of the inventory management database]

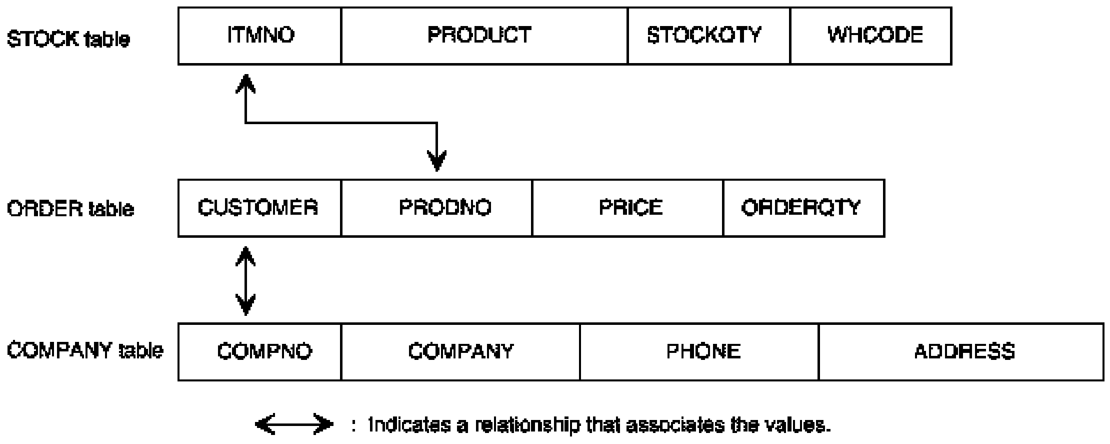
Table name	Column name	Column data type	Column constraint	Notes
STOCK	ITMNO	SMALLINT	NOT NULL	Product code number
	PRODUCT	CHARACTER(25)	NOT NULL	Product type
	STOCKQTY	INTEGER	—	Quantity of product in stock
	WHCODE	SMALLINT	—	Warehouse code
ORDER	CUSTOMER	SMALLINT	NOT NULL	Customer company number
	PRODNO	SMALLINT	NOT NULL	Product code number
	PRICE	INTEGER	—	Product purchase price
	ORDERQTY	SMALLINT	—	Quantities of the product ordered
COMPANY	COMPNO	SMALLINT	NOT NULL	Company code number
	COMPANY	CHARACTER(10)	NOT NULL	Company name
	PHONE	CHARACTER(14)	—	Telephone number
	ADDRESS	CHARACTER(30)	—	Company address

—: NULL is permitted.

■Relationships among the STOCK table, ORDER table, and COMPANY table

Figure: Relationships among the STOCK table, ORDER table, and COMPANY table shows the relationships among the three tables. The STOCK table and ORDER table are related according to ITMNO and PRODNO. In addition, the ORDER table and COMPANY table are related according to CUSTOMER and COMPNO. For example, the product having ITMNO 123 in the STOCK table is the PRODUCT named REFRIGERATOR. Further, the STOCKQTY is 60, and the number of the warehouse in which this product is stored is 1. From the row of the ORDER table in which PRODNO is 123, the PRICE and ORDERQTY of this product are 48000 and 60, respectively. Moreover, since the company number of the CUSTOMER is 61, the following data can be gleaned from the COMPANY table row in which COMPNO is 61. Users can learn the company name, telephone number, and address of the CUSTOMER of that product.

[Figure: Relationships among the STOCK table, ORDER table, and COMPANY table]



2.3 Creating a Database

This section contains the following topics to explain how to create databases:

- Defining a database by using the rdbddlex command
- Defining a database from an application program

2.3.1 Defining a database by using the rdbddlex command

This section shows how to create databases from a definition file.

Physical, logical, and storage structures can be defined using the rdbddlex command.

The first step is to create an input file to be used by the rdbddlex command. The next step is to execute the rdbddlex command.

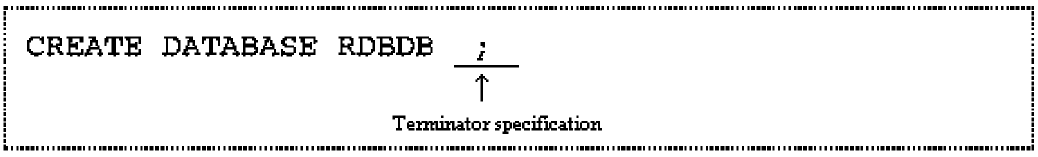
The d option of the rdbddlex command can only be omitted if the first SQL statement of the definition file is the CREATE DATABASE statement.

■Input file format of rdbddlex command

The syntax of for describing an input file of the rdbddlex command has the following general formats:

◆Format 1

Specify a semicolon (;) to terminate each SQL statement.



```
CREATE DATABASE RDBDB ;
```

◆Format 2 (Format for defining a procedure routine)

To define a procedure routine, prefix the input file with "EXEC SQL" and suffix it with "END-EXEC;". The data between "EXEC SQL" and "END-EXEC;" is assumed to be an SQL statement. This format is only valid if the x option has been specified in the rdbddlex command.

```
EXEC SQL
  CREATE PROCEDURE PROC01
     :
END-EXEC;
```

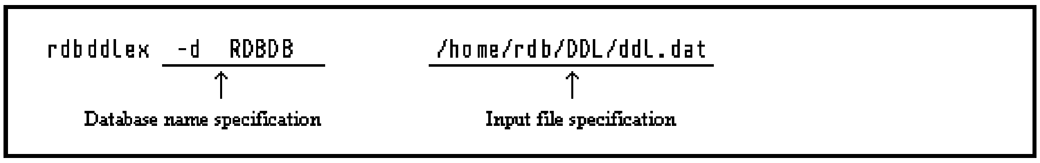
Figure: Sample creation of a database from a definition file is a sample of database creation from a definition file. Figure: Sample definition file is a sample definition file.

These figures are examples for Solaris. For Windows NT, change the input file specification in the rdbddlex command and the database space definition in the input file as shown below.

- Windows NT/2000/XP
  - Input file specification: C:¥USERS¥DEFAULT¥DDL.DAT
  - Database space definition: CREATE DBSPACE DBSPACE1 ALLOCATE FILE C:¥SFWD¥RDB¥USR¥DBSP¥DATABASE_SPACE ATTRIBUTE SPACE(2M)

Use two consecutive hyphens (--) to specify comments. Everything on the line after the two hyphens is treated as a comment.

[Figure: Sample creation of a database from a definition file]



```
rdbddlex -d RDBDB          /home/rdb/DDL/ddl.dat
```

[Figure: Sample definition file]

ddl.dat

```
-- Define a database named "RDBDB"

CREATE DATABASE RDBDB;

-- Define a database space named "DBSPACE 1"

CREATE DBSPACE DBSPACE1 ALLOCATE RAWDEVICE  /dev/rdsk/c0t1d0s1;

-- Define a schema named "STOCKS"

CREATE SCHEMA STOCKS

    -- Define a management table for stocked products.

    CREATE TABLE  STOCK (ITMNO        SMALLINT NOT NULL,
                         PRODUCT      CHARACTER(25) NOT NULL,
                         STOCKQYT     INTEGER,
                         WHCODE       SMALLINT,
                         PRIMARY KEY  (ITMNO)
                        )

    CREATE VIEW   MASS_STOCK (NO, QTY)
                     AS SELECT  ITMNO, STOCKQTY FROM STOCKS.STOCK
                     WHERE STOCKQTY >= 50

    -- Define a management table for ordered products.

    CREATE TABLE  ORDER (CUSTOMER     SMALLINT NOT NULL,
                         PRODNO       SMALLINT NOT NULL,
                         PRICE        INTEGER,
                         ORDERQTY     SMALLINT,
                         PRIMARY KEY  (CUSTOMER, PRODNO)
                        )

    CREATE VIEW   MASS_ORDER (PRODNO, PRICE)
                     AS SELECT  PRODNO, PRICE FROM STOCKS.ORDER
                     WHERE ORDERQTY >= 100

  -- Define a management table for custmer companies.

    CREATE TABLE COMPANY (COMPNO      SMALLINT NOT NULL,
                          COMPANY     CHARACTER(25) NOT NULL,
                          PHONE       CHARACTER(14),
                          ADDRESS     CHARACTER(30),
                          PRIMARY KEY (COMPNO)
                         )

    CREATE VIEW  COMPANY1 AS SELECT COMPNO, COMPANY FROM STOCKS.COMPANY;
```

ddl.dat

```
-- Define a STOCK table storage structure.

CREATE DSO STOCK_DSO                                          -- STOCK table DSO
           FROM STOCKS.STOCK
           TYPE SEQUENTIAL(PAGESIZE(4).ORDER(1));

CREATE DSI STOCK_DSI                                          -- STOCK table DSI
           DSO STOCK_DSO
           ALLOCATE DATA ON DBSPACE1 SIZE 280k;

-- Create an inidex based on PRODUCT of the STOCK table.

CREATE DSO PRODUCT_IXDSO                                      -- Index DSO
           INDEX ON STOCKS.STOCK (PRODUCT)
           TYPE     BTREE ( PAGESIZE1(4), PAGESIZE2(4)) BY ADDRESS;

CREATE DSI PRODUCT_IXDSI                                      – Index DSI
           INDEX
           DSO      PRODUCT_IXDSO
           ALLOCATE INDEX     ON DBSPACE1 SIZE  40K,
                    BASE      ON DBSPACE1 SIZE 200K;

-- Define an ORDER table storage structure.

CREATE DSO ORDER_DSO
           FROM STOCKS.ORDER
           TYPE SEQUENTIAL(PAGESIZE(4).ORDER(1))

           WHERE (CUSTOMER) BETWEEN (?) AND (?);              -- Split and place data for each customer number.

CREATE DSI USA_ORDER_DSI                                      -- ORDER table DSI is for companies located in the USA.
           DSO      ORDER_DSO
           USING    (62, 63)                                  -- Numbers from 62 to 63 are companies in the USA.
           ALLOCATE DATA ON DBSPACE1 SIZE 280k;

CREATE DSI JAPAN_ORDER_DSI                                    -- ORDER table DSI is for companies located in the USA.
           DSO      ORDER_DSO
           USING    (71, 72)                                  -- Numbers from 71 to 72 are companies in the Japan.
           ALLOCATE DATA ON DBSPACE1 SIZE 280k;

-- Create an index for CUSTOMER and MERCHANDISE of the ORDER table.
CREATE DSO BUSINESS_IXDSO                                     -- Index DSO
           INDEX ON STOCKS. ORDER (CUSTOMER, MERCHANDISE)
           TYPE     BTREE ( PAGESIZE1(4),PAGESIZE2(4)) BY ADDRESS;
CREATE DSI EAST_BUSINESS_IXDSI                                -- Index DSI
           INDEX
           DSO      BUSINESS_IXDSO
           BASE     EAST_ORDER_DSI
           ALLOCATE BASE      ON DBSPACE1 SIZE 200K,
                    INDEX     ON DBSPACE1 SIZE  40K;
CREATE DSI WEST_BUSINESS_IXDSI                                -- Index DSI
           INDEX
           DSO      BUSINESS_IXDSO
           BASE     WEST_ORDER_DSI
           ALLOCATE BASE      ON DBSPACE1 SIZE 200K,
                    INDEX     ON DBSPACE1 SIZE  40K;
-- Define a COMPANY table storage structure

CREATE DSO COMPANY_DSO                                        -- COMPANY table DSO
           FROM STOCKS.COMPANY
           TYPE SEQUENTIAL(PAGESIZE(4).ORDER(1));


CREATE DSI COMPANY_DSI                                        -- COMPANY table DSI
           DSO   COMPANY_DSO
           ALLOCATE DATA ON DBSPACE1 SIZE 280k;

-- Create an index for COMPANY NO. of the COMPANY table.
CREATE DSO COMPANY NO. IXDSO                                  -- Index DSO
           INDEX ON STOCKS. COMPANY (COMPANY NO.)
           TYPE     BTREE ( PAGESIZE1(4),PAGESIZE2(4)) BY ADDRESS;
CREATE DSI COMPANY NO. IXDSI                                  -- Index DSI
           INDEX
           DSO      COMPANY NO. IXDSO
           ALLOCATE BASE      ON DBSPACE1 SIZE 200K,
                    INDEX     ON DBSPACE1 SIZE  40K;
```

2.3.2 Defining the database from an application program

This section shows how to use dynamic SQL statements to create a database. Logical and storage structures can be defined from an application program. Register a database name and create a database space in advance by using the rdbddlex command because these tasks cannot be executed from an application program. CMDAREA1 to

CMDAREA3 are set up as SQL statement variables. The programming language used is C. Figure: Sample application program definition is a sample of definition by an application program.

[Figure: Sample application program definition]

```
          :
/* Copy the definition statement to SQL statement variables. */
strcpy(CMDAREA1.sqlvar,"CREATE SCHEMA STOCKS
                        CREATE TABLE  STOCKS (ITMNO    SMALLINT NOT NULL,
                                              PRODUCT   CHARACTER(25) NOT NULL,
                                              STOCKQTY  INTEGER,
                                              WHCODE    SMALLINT,
                                              PRIMARY KEY (ITMNO)
                                            )
                        CREATE VIEW  MASS_STOCK (NO, QTY)
                                     AS SELECT ITMNO, STOCKQTY FROM STOCKS.STOCK
                                     WHERE STOCKQTY >= 50");
CMDAREA1.sqllen = strlen(CMDAREA1.sqlvar);
strcpy(CMDAREA2.sqlvar,"CREATE DSO STOCK_DSO FROM STOCKS.STOCK
                                  TYPE SEQUENTIAL (PAGESIZE(4), ORDER(0))");

CMDAREA2.sqllen = strlen(CMDAREA2.sqlvar);
strcpy(CMDAREA3.sqlvar,"CREATE DSI STOCK_DSI
                            DSO STOCK_DSO
                            ALLOCATE DATA ON DBSPACE1 SIZE 280K");

CMDAREA3.sqllen = strlen(CMDAREA3.sqlvar);
        :
        :
        :
/* Execute the dynamic SQL statements. */
EXEC SQL EXECUTE IMMEDIATE :CMDAREA1;
EXEC SQL EXECUTE IMMEDIATE :CMDAREA2;
EXEC SQL EXECUTE IMMEDIATE :CMDAREA3;
```

If a storage structure is simplified for database definition from an application program, one program covers the steps from definition to creation. However, a COMMIT statement should be specified before data manipulation.

[Figure: Sample definition to creation from an application program]

```
        :
/* Copy the definition statements to SQL statement variables. */
strcpy(CMDAREA1.sqlvar,"CREATE SCHEMA STOCKS
                        CREATE TABLE  STOCK (ITMNO     SMALLINT NOT NULL,
                                             PRODUCT   CHARACTER(25) NOT NULL,
                                             STOCKQTY  INTEGER,
                                             WHCODE    SMALLINT,
                                             PRIMARY KEY(ITMNO)
                                            ) ON DBSPACE1");



CMDAREA1.sqllen = strlen(CMDAREA1.sqlvar);
        :
        :
        :
/* Execute the dynamic SQL statements. */
EXEC SQL EXECUTE IMMEDIATE  :CMDAREA1;
        :
        :
        :
/* Execute COMMIT before beginning data manipulation. */
EXEC SQL COMMIT WORK;

EXEC SQL INSERT INTO STOCKS. STOCK (ITMNO, PRODUCT, STOCKQTY, WHCODE)
                VALUES (:ITMNO, :PRODUCT, :STOCKQTY, :WHCODE);
        :
        :
        :
```

2.3.3 - Omitted -

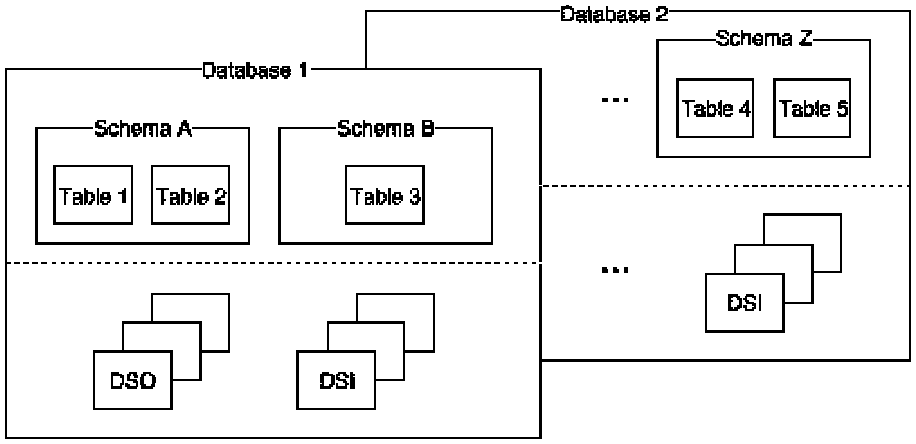
2.4 Entering a Database Name

All logical structure definitions and storage structure definitions belong to a given database environment. Logical structure definitions are the schemas and tables to be created. Such storage structure definitions are the DSOs and DSIs. The user must enter the database name before defining the logical and storage structures.

When a database name is entered, that information is stored in the RDB dictionary.

Figure: Configuration of a database shows the configuration of a database.

[Figure: Configuration of a database]



■CREATE DATABASE statement

Enter a database name using the CREATE DATABASE statement. Specify the database name to be entered in this SQL statement. The specified database name is entered in the RDB dictionary.

Example 1:

Enter MASTER_DB as a database.

```
CREATE DATABASE MASTER_DB
```

Example 2:

Enter STOCKMN_DB as a database.

```
CREATE DATABASE STOCKMN_DB
```

■Database name

For the database name, specify up to 36 alphanumeric characters beginning with an alphabetic character.

2.5 Creating a Database Space

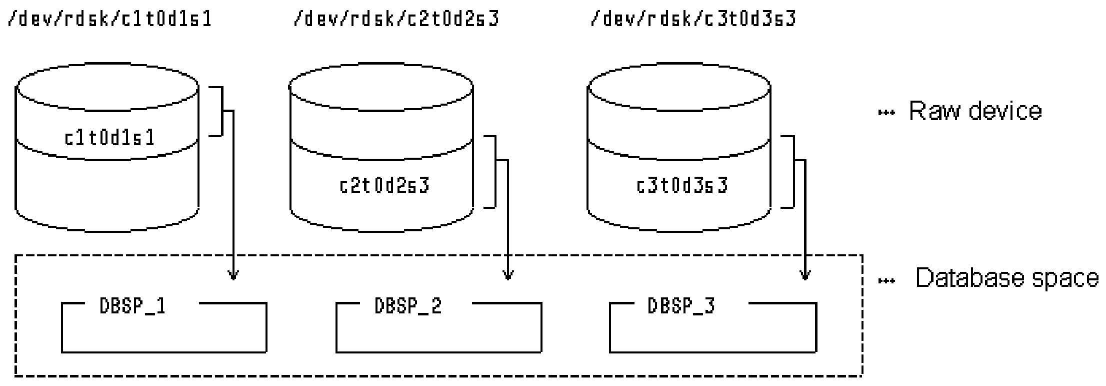
Allocate database space as an area for processing a database. The database space can be reserved on a raw device or as a local file on a magnetic disk. The raw device is used for a database space under UNIX. The local file is used for a database space under Windows NT/2000/XP.

At the creation of a database space, a log environment can be allocated for each database space.

This section explains the relationships between database space and magnetic disk, and the correspondence between the database space and the log environment.

2.5.1 Creating a database space on a raw device

Under UNIX, a partition on a magnetic disk is allocated as a database area; therefore, an actual raw device must be acquired before a database space can be created.



To create a database space, use the CREATE DBSPACE statement.

Executing the CREATE DBSPACE statement associates the database space and an actual raw device as well as the database space and a log environment. The CREATE DBSPACE statement also registers information about the database space in the RDB dictionary.

■CREATE DBSPACE statement

In the CREATE DBSPATE statement, specify the database space name and the name of the raw device in which the database space is to be created.

The following example shows the execution of a CREATE DBSPACE statement to define a database space for a stock management database.

Example:

Create database spaces DBSP_1, DBSP_2, and DBSP_3. Then allocate the following raw devices to their respective database spaces. These raw devices must have been defined in advance.

· DBSP_1.../dev/rdsk/c1t0d1s1
· DBSP_2.../dev/rdsk/c2t0d2s3
· DBSP_3.../dev/rdsk/c3t0d3s3

```
CREATE DBSPACE  DBSP_1  ALLOCATE RAWDEVICE /dev/rdsk/c1t0d1s1
CREATE DBSPACE  DBSP_2  ALLOCATE RAWDEVICE /dev/rdsk/c2t0d2s3
CREATE DBSPACE  DBSP_3  ALLOCATE RAWDEVICE /dev/rdsk/c3t0d3s3
```

◆Database space name

For the database space name, specify up to 36 alphanumeric characters beginning with an alphabetic character.

◆Raw device name

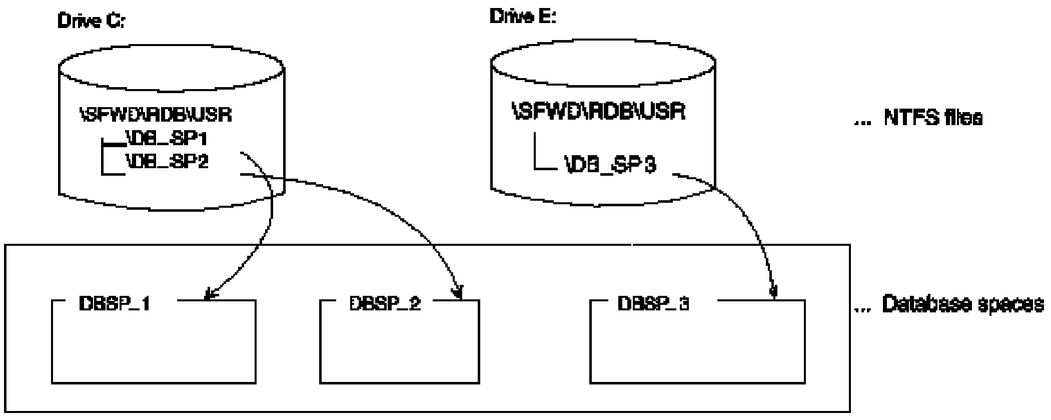
Specify the name of the raw device to be allocated for the database space.

◆Notes on operating multi-RDB

For operation of a multi-RDB, the specified raw device may be shared by another SymfoWARE/RDB environment. After a required raw device is created, use the chown and chmod commands to configure the access rights so that only the activation user of each system can access the raw device. For details on the chown and chmod commands, refer to the commands reference manual of the relevant operating system.

2.5.2 Creating a database space on a local file

Under Windows NT/2000/XP, an NTFS file is allocated to a database area.



Create the database space by using the CREATE DBSPACE statement. When the CREATE DBSPACE statement is executed, the database space is associated with an actual local file. In addition, the database space is associated with a log environment. Information related to the database space is entered in the RDB dictionary.

■CREATE DBSPACE statement

In the CREATE DBSPACE statement, specify the names of the database space and the local file for creating the database space.

Sample CREATE DBSPACE statements for executing database space definitions for STOCKMN_DB follow.

Example:

Create database spaces DBSP_1, DBSP_2, and DBSP_3.

The following database-dedicated NTFS files are allocated to these database spaces:

DBSP_1 :
C:¥SFWD¥RDB¥USR¥DBSP1¥DB_SP1
DBSP_2 :
C:¥SFWD¥RDB¥USR¥DBSP1¥DB_SP2
DBSP_3 :
E:¥SFWD¥RDB¥USR¥DBSP1¥DB_SP3

```
CREATE DBSPACE  DBSP_1  ALLOCATE FILE  C:\SFWD\RDB\USR\DBSP1\DB_SP1
                                              ATTRIBUTE SPACE(2M)
CREATE DBSPACE  DBSP_2  ALLOCATE FILE  C:\SFWD\RDB\USR\DBSP2\DB_SP2
                                              ATTRIBUTE SPACE(1M)
CREATE DBSPACE  DBSP_3  ALLOCATE FILE  E:\SFWD\RDB\USR\DBSP3\DB_SP3
                                              ATTRIBUTE SPACE(3M)
```

■Database space name

For the database name, specify up to 36 alphanumeric characters beginning with an alphabetic character.

■File name

Specify the name of file to be allocated to the database space.

2.5.3 - Omitted -

2.5.4 Operation of a scalable log

When a database space is created, a log environment can be associated with it. The method of specifying this scalable log is as follows.

Example:

Associate log group group1 with database spaces DBSP_1 and DBSP_3, and log group group2 with database space DBSP_2.

```
CREATE DBSPACE  DBSP_1  ALLOCATE RAWDEVICE /dev/rdsk/c1t0d1s1
                        ATTRIBUTE LOG GROUP group1
CREATE DBSPACE  DBSP_2  ALLOCATE RAWDEVICE /dev/rdsk/c2t0d2s3
                        ATTRIBUTE LOG GROUP group2
CREATE DBSPACE  DBSP_3  ALLOCATE RAWDEVICE /dev/rdsk/c3t0d3s3
                        ATTRIBUTE LOG GROUP group1
```

◆Log group name

For the log group name, specify up to 18 characters consisting of alphanumeric characters and underbar (_).

If "system" is specified for the log group name or no log group name is specified, the log group of the system is assumed to have been specified.

For details on scalable log operation, refer to the RDB Operations Guide.

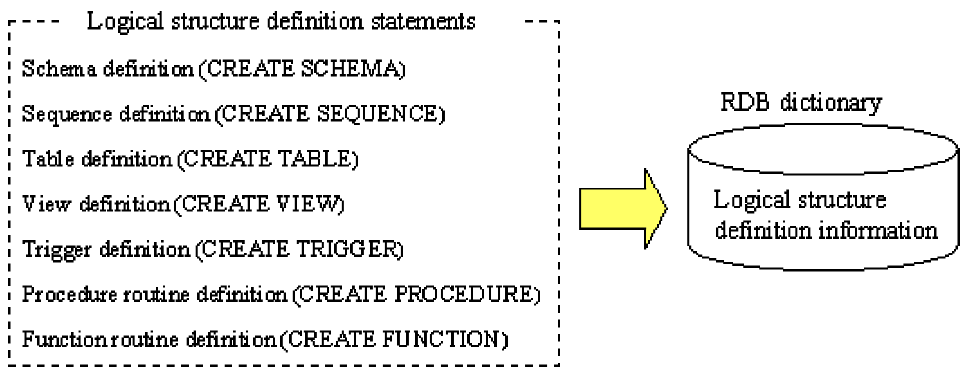
2.6 Defining a Logical Structure

After a database space has been created, define the logical structure. When the logical structure is defined, the definition information is stored in the RDB dictionary.

The logical structure definitions include schema definitions, sequence definitions, table definitions, view definitions, trigger definitions, procedure routine definitions, and function routine definitions.

Figure: Logical structure definition procedure shows the logical structure definition procedure.

[Figure: Logical structure definition procedure]


Logical structure definition statements
Schema definition (CREATE SCHEMA)
Sequence definition (CREATE SEQUENCE)
Table definition (CREATE TABLE)
View definition (CREATE VIEW)
Trigger definition (CREATE TRIGGER)
Procedure routine definition (CREATE PROCEDURE)
Function routine definition (CREATE FUNCTION)
RDB dictionary
Logical structure definition information

■Schema definition

Schemas are managed according to schema names. Multiple schemas can be created for a single database.

Define a schema using a CREATE SCHEMA statement. A schema definition includes definitions of the base tables and views, the elements that form the schema.

■Sequence definition

A sequence can be defined to automatically generate values within the sequence. The user can use a sequence to create primary key values.

Define a sequence using the CREATE SEQUENCE statement. The CREATE SEQUENCE statement can also be used to add a sequence to a previously defined schema.

■Table definition

A table definition defines a base table name and the columns that form the base table. Multiple base tables can be created for a single schema. Define a base table using the CREATE TABLE statement.

The CREATE TABLE statement can also be used to add a base table to a previously defined schema.

■View definition

A view definition defines a view name and the columns that form the view. Multiple views can be created for a single schema. Define a view using the CREATE VIEW statement.

The CREATE VIEW statement can also be used to add a view to a previously defined schema.

■Trigger definition

If a trigger definition is made, data can be automatically inserted into another table when an application program updates a table.

Use the CREATE TRIGGER statement to make a trigger definition. The CREATE TRIGGER statement can also be used to add a trigger definition to a previously defined schema.

■Procedure routine definition

The database operation tasks include tasks that always process data according to a fixed pattern. These fixed-pattern processes can be defined in a schema as a processing procedure called a procedure routine. In the procedure routine definition, specify a procedure routine that belongs to the schema. Multiple procedure routines can be created for one schema. Define a procedure routine using the CREATE PROCEDURE statement.

The CREATE PROCEDURE statement can also be used to add a procedure routine definition for a defined schema.

■Function routine definition

A function routine definition defines a user-created application program written in C as a function. A function routine can be defined to specify a function in an SQL statement and process it. Define a function routine using the CREATE FUNCTION statement.

The CREATE FUNCTION statement can also be used to add a function routine to a previously defined schema.

2.6.1 Schema definition

A schema definition defines a schema name, a schema comment definition, and the following elements that form the schema:

- · Sequences
- · Base tables

· View tables
· Triggers
· Procedure routines
· Function routines

Define the schema name using the schema definition statement (CREATE SCHEMA statement).

A sample schema definition for the inventory management database follows. The schema named STOCKS and the tables that belong to it, such as the STOCK table, are defined for STOCKMN_DB.

Example:

Define a schema for STOCKMN_DB.

```
CREATE SCHEMA  STOCKS   COMMENT 'FOR STOCK MANAGEMENT'
                 ↑                     ↑
            Schema name          Comment definition

CREATE SEQUENCE  SEQUENCE 1  ···                                (1)

CREATE TABLE  STOCK TABLE  (ITMNO···, ··· PRIMARY KEY (ITM NO))  (2)
CREATE TABLE  ···
    :

CREATE VIEW  MASS-STOCK(NO, QTY)  AS SELECT  ···                (3)
CREATE VIEW  ···
    :

  CREATE TRIGGER  STOCK TABLE  TRIGGER  ···                     (4)
    :

CREATE  PROCEDURE  PROC001  ···                                 (5)
  BEGIN
    :
  END

CREATE  FUNCTION  SUER001  ···                                  (6)
```

(1) Sequence definition
(2) Table definitions
(3) View definitions
(4) Trigger definition
(5) Procedure routine definition
    When using the rdbddlex command to specify a procedure routine definition, do not specify the definition in a definition file that has another definition. Specify the procedure routine definition in another definition file by using the rdbddlex command with the x option.
(6) Function routine definition
    When using the rdbddlex command to specify a function routine definition, do not specify the definition in a definition file that has another definition. Specify the function routine definition in another definition file by using the rdbddlex command (without the x option).

■Schema name

For the schema name, specify up to 36 alphanumeric characters beginning with an alphabetic character. The schema name must be unique within the database.

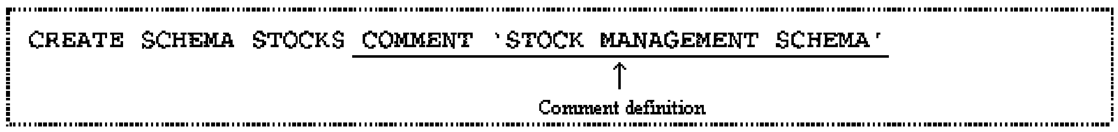
When sequences, base tables, and view tables are specified in SQL statements, the schema name is used to qualify the sequence and table names. The schema name is also used to qualify the table names if an index is specified in an index definition.

■Schema comment definition

A comment consisting of a character string can be specified for the schema. A character string of up to 256 bytes can be specified. If no comment is necessary, omit the specification. An example follows.

Example:

Specify a comment for the STOCKS schema.



```
CREATE SCHEMA STOCKS COMMENT 'STOCK MANAGEMENT SCHEMA'
```

2.6.2 Sequence definition

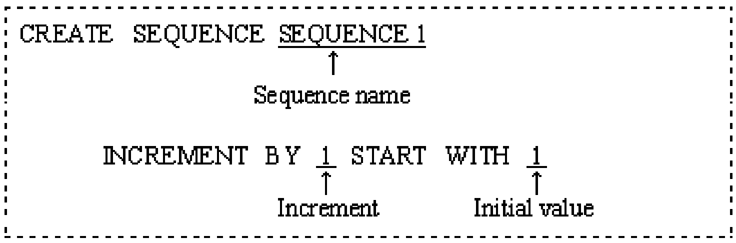
Define a sequence using the CREATE SEQUENCE statement.

A sequence can be defined to automatically generate unique names within the sequence. The user can use a sequence to create primary key values.

A sample sequence definition for a stock management database follows. This sequence definition defines a sequence that belongs to a schema named STOCKS.

Example:

Sample sequence definition



```
CREATE SEQUENCE SEQUENCE 1

      INCREMENT BY 1 START WITH 1
```

■Sequence name

Specify a name to be assigned to a sequence. Specify up to 36 alphanumeric characters for a sequence name, whose first character must be an alphabetic character. Each sequence name within a schema must be unique. A sequence with the same sequence name may be defined in another schema.

2.6.3 Table definition

Define a base table using the CREATE TABLE statement.

The table definition defines the following items:

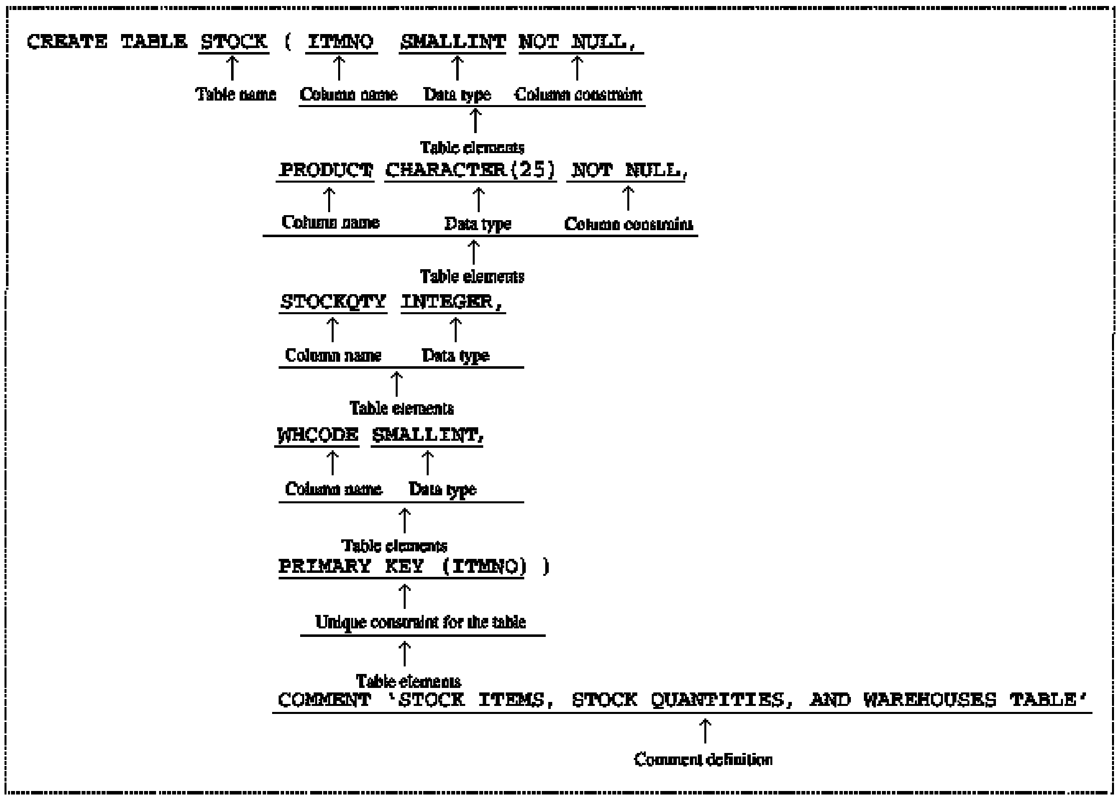
- Table name
- Columns that form the base table
  - Column name
  - Column data type (such as character, integer or date-and-time type)
  - Column constraint (such as a unique constraint or allowing or disallowing NULL values)
  - Column comment definition
- Table constraint for the base table
- Table comment definition

A sample base table definition for the inventory management database follows. This table definition defines the STOCK table that belongs to the schema STOCKS.

Example:

Figure: CREATE TABLE statement that defines the STOCK table shows the CREATE TABLE statement that defines the STOCK table.

[Figure: CREATE TABLE statement that defines the STOCK table]



```
CREATE TABLE STOCK ( ITMNO  SMALLINT NOT NULL,
                     PRODUCT CHARACTER(25) NOT NULL,
                     STOCKQTY INTEGER,
                     WHCODE SMALLINT,
                     PRIMARY KEY (ITMNO) )
                     COMMENT 'STOCK ITEMS, STOCK QUANTITIES, AND WAREHOUSES TABLE'
```

■Table name

Specify a name to be assigned to the base table using up to 36 alphanumeric characters beginning with an alphabetic character.

The table name must be unique within the schema. The same table name can be defined in other schemas.

When a table definition is specified as a schema definition element and the table name is modified by a schema name, the table name must be same as the schema name specified in the schema definition. When the table name is not modified by the schema name, the table name is considered to be modified by the schema name specified in the schema definition.

The table name is used to specify the table to be manipulated by a data manipulation SQL statement.

Example 1:

Sample table names

```
CREATE TABLE  STOCK ( ... )
CREATE TABLE  A1234 ( ... )
```

Example 2:

Invalid table name specification

```
CREATE SCHEMA   STOCKS
CREATE TABLE S.STOCK ( ... )   ⇐ The schema name is not STOCKS.
```

Example 3:

Valid table name specifications

```
CREATE SCHEMA   STOCKS
CREATE TABLE STOCKS.STOCK ( ... )     ⇐ The schema name is valid.
CREATE TABLE ORDER ( ... )            ⇐ The schema name is omitted.
                                        STOCKS is used as the table name.
```

■Column definition

Define the following items for each column that forms the table:

- Column name
- Column data type
- Default value
- Column constraint
- Column comment definition

The column name and column data type must be specified in a column definition. The other items can be specified as required.

◆Column name

Specify a name to be assigned to the column. For the column name, specify up to 36 alphanumeric characters beginning with an alphabetic character. The column name must be unique within a table.

Example:

Sample column names

```
CREATE TABLE S1.STOCK  ( PNO     SMALLINT ...
CREATE TABLE S1.STOCK  ( COL1    SMALLINT ...
```

◆Column data type

Specify the data type of the column. Table: Column data types shows the types that can be specified. The data type is determined by the type of data to be stored and the data size (length).

Example 1:

Let the data type of the PRODUCT column of the STOCK table be a 10-character fixed length character string.

```
CREATE TABLE  STOCK  (PRODUCT  CHARACTER(10) ...)
```

Example 2:

Same definition as example 1

```
CREATE TABLE  STOCK  (PRODUCT  CHAR(10) ...)
```

Example 3:

Let the data type of the PRODUCT column of the STOCK table be a 10-character variable length character string.

```
CREATE TABLE  STOCK  (PRODUCT  CHARACTER VARYING(10) ...)
```

Example 4:

Same definition as example 3

```
CREATE TABLE  STOCK  (PRODUCT  VARCHAR(10) ...)
```

Example 5:

Let the data type of the STOCKQTY column of the STOCK table be a 10-digit external decimal number with two digits to the right of the decimal point.

```
CREATE TABLE  STOCK  (STOCKQTY  NUMERIC(10,2) ...)
```

Example 6:

Let the data type of the STOCKQTY column of the STOCK table be a 10-digit internal decimal number with two digits to the right of the decimal point.

```
CREATE TABLE  STOCK  (STOCKQTY  DECIMAL(10,2) ...)
```

Example 7:

The data type of the STOCKQTY column of the STOCK table is as follows:

Integer in range of -2^{31} to $2^{31}-1$

```
CREATE TABLE  STOCK  (STOCKQTY  INTEGER ...)
```

Integer in range of -2^{15} to $2^{15}-1$

```
CREATE TABLE  STOCK  (STOCKQTY  SMALLINT ...)
```

Example 8:

Let the data type of the STOCKQTY column of the STOCK table be an approximate numeric value with precision 22.

```
CREATE TABLE  STOCK  (STOCKQTY  FLOAT(22) ...)
```

Example 9:

Let the data type of the STOCKQTY column of the STOCK table be a double-precision approximate numeric value.

```
CREATE TABLE  STOCK  (STOCKQTY  DOUBLE PRECISION ...)
```

[Table: Column data types]

Type	Data type specification format	Explanation of specification

Character string type	CHARACTER(n)
	CHAR(n)
		Fixed length character string of length n
		If (n) is omitted, the length defaults to one.
		n: 1 to 32000

	CHARACTER VARYING(n) CHAR VARYING(n) VARCHAR(n)	Variable length character string of maximum length n If (n) is omitted, the maximum length defaults to one. n: 1 to 32000
Exact numeric type	NUMERIC(p, q)	p-digit zoned decimal number with q digits to the right of the decimal point p: 1 to 18; q: 0 to p

	DECIMAL(p, q)
	DEC(p, q)
		p-digit packed decimal number with q digits to the right of the decimal point
		p: 1 to 18; q: 0 to p

	INTEGER INT	Integer from -2^{31} to $2^{31} - 1$
	SMALLINT	Integer from -2^{15} to $2^{15} - 1$
Approximate numeric type	FLOAT(p)	Approximate numeric value with mantissa from -2^{p} to 2^{p} p: 1 to 52 When p = 1 to 23, treated as REAL When p = 24 to 52, treated as DOUBLE PRECISION
	REAL	4-byte floating point number
	DOUBLE PRECISION	8-byte floating point number
Date time type	DATE	10-character date from years to days
	TIME	8-character time from hours to seconds
	TIMESTAMP	19-character time stamp from years to seconds
Interval type	INTERVAL start-field TO end-field	Interval (years to months, or days to times) indicated by the field specifications (For details, see Table: Time interval specifications.)
BLOB type	BINARY LARGE OBJECT (n units) BLOB (n units)	Binary attribute data The unit specification is K, M, or G and cannot be omitted. If the unit specification is K: n = 1 to 2097152 If the unit specification is M: n = 1 to 2048 If the unit specification is G: n = 1 or 2

n:	Number of characters
p:	Precision
q:	Scale

[Table: Time interval specifications]

Type	Start field	End field	Explanation of specification
Year and month type	YEAR(p)	-	p-digit time interval indicating years
		MONTH	p-digit time interval indicating years and months
	MONTH(p)	-	p-digit time interval indicating months
Day and hour type	DAY(p)	SECOND	p-digit time interval indicating days, hours, minutes, and seconds
		MINUTE	p-digit time interval indicating days, hours, and minutes
		HOUR	p-digit time interval indicating days and hours
		-	p-digit time interval indicating days
	HOUR(p)	SECOND	p-digit time interval indicating hours, minutes, and seconds
		MINUTE	p-digit time interval indicating hours and minutes
		-	p-digit time interval indicating hours
	MINUTE(p)	SECOND	p-digit time interval indicating minutes and seconds
		-	p-digit time interval indicating minutes
	SECOND(p)	-	p-digit time interval indicating seconds

p: Precision of start date and time field (Specify an integer from 1 to 9.)

◆Default value

A value can be specified as a default value for a column. Specify a value to be set in the column if no value is specified when a row is inserted in the table. The defaults can be specified with a constant, login name (under UNIX) or logon name (under Windows NT/2000/XP), NULL, the current date, the current time, and the current timestamp.

Example 1:

Sample column definition for the ITMNO column of the STOCK table

```
CREATE TABLE  STOCK TABLE  (ITMNO  INTEGER  DEFAULT 10
                                                ↑
                                   Default value specification
                             :
```

Example 2:

Sample column definition for using a sequence for the ITMNO column of the STOCK table

```
CREATE TABLE  STOCK TABLE  (ITMNO  INTEGER  DEFAULT SEQUENCE 1.NEXTVAL
                                                       ↑
                                          Default value specification
                             :
```

◆Column constraint

A constraint on the data to be stored can be specified for a column. Specify a constraint after the data type specification. The following two kinds of column constraints can be specified:

NOT NULL constraint:

Specify this constraint when NULL is not permitted as column data. Specify NOT NULL.

Unique constraint:

Specify this constraint when duplicate values are not permitted as column data. Specify UNIQUE or PRIMARY KEY.

The unique constraint is detailed later on.

A sample column definition for the ITMNO column of the STOCK table follows. The following conditions are assumed for ITMNO:

- ITMNO is an integer having up to eight digits.
- ITMNO is unique for each product, and a row of the STOCK table is uniquely identified by ITMNO.
- A row cannot be inserted unless an ITMNO value is entered.

Example:

Sample column definition for the ITMNO column of the STOCK table

```
CREATE TABLE   STOCK (ITMNO   INTEGER
                              NOT NULL
                              PRIMARY KEY
                        :
```

◆Column comment definition

A comment consisting of a character string can be specified for a column. A character string of up to 256 bytes can be specified. If no comment is necessary, omit the specification. An example follows.

Example:

Specify a comment for the ITMNO column in the STOCK table.

```
CREATE TABLE   STOCK (ITMNO  SMALLINT...COMMENT 'PRODUCT-NO' )
                        :                          ↑
                                           comment definition
```

■Unique constraint

The unique constraint can be specified as a constraint for a group of several columns within a table. This type of specification is called a table constraint. A unique constraint can also be specified as a column constraint for a single column of a table.

With a unique constraint specification, the specified column or group of columns cannot have the same value or group of values in more than one row. The value of the specified column or values of the group of columns are determined uniquely within the table. The unique constraint is specified by UNIQUE or PRIMARY KEY.

◆UNIQUE

Specify UNIQUE in the following situation. The table is not permitted to have more than one row with the same value or values for the specified column or group of columns. The specification format is as follows.

```
UNIQUE (column-name [{, column-name} ...])
```

NOT NULL must already be specified in the column definition for any column for which UNIQUE is specified.

The next example defines the STOCK table with the constraint that two or more rows cannot have identical values in both the ITMNO and PRODUCT columns.

Example:

Sample table constraint specification for a group of columns

```
CREATE TABLE  STOCK ( ...,

                         UNIQUE (ITMNO , PRODUCT)
                                   ↑        ↑
                              Column name  Column name
                                      ↑
                          Unique constraint for the table
```

The STOCK table for which the unique constraint of this example has been specified cannot have rows such as [3] and [4] in Figure: Sample data that violates the unique constraint. Rows [3] and [4] in Figure: Sample data that violates the unique constraint violate the unique constraint because they both have 123 as the ITMNO and they both have REFRIGERATOR as the PRODUCT. Rows [1] and [2] do not violate the unique constraint because the ITMNO values differ even though the PRODUCT value is the same. Similarly, rows [5] and [6] and rows [7] and [8] do not violate the unique constraint. If the unique constraint were specified only for the ITMNO column, then rows [3] and [4] and rows [5] and [6] in Figure: Sample data that violates the unique constraint would violate the unique constraint.

[Figure: Sample data that violates the unique constraint]

	ITMNO	PRODUCT	STOCKQTY	WHCODE
[1]→	110	TELEVISION	85	2
[2]→	111	TELEVISION	90	2
[3]→	123	REFRIGERATOR	60	1
[4]→	123	REFRIGERATOR	75	1
[5]→	140	CASSETTE DECK	120	2
[6]→	140	TELEVISION	0	2
[7]→	226	REFRIGERATOR	8	1
[8]→	227	REFRIGERATOR	15	1

◆PRIMARY KEY

One or a combination of columns used for determining that a row in a table is unique is called a primary key. The value specified for primary keys must be unique for each row in a table. The specification format is as follows.

```
PRIMARY KEY (column-name [{, column-name} ...])
```

NOT NULL must already be specified in the column definition for any column for which PRIMARY KEY is specified. PRIMARY KEY can only be specified once within a table definition.

The next example specifies the unique constraint related to the ITMNO column of the STOCK table as a table constraint.

Example:

Sample unique constraint specification for the ITMNO column as a table constraint

```
CREATE TABLE  STOCK (...,

                           PRIMARY KEY (ITMNO)
                                          ↑
                                     Column name
                           ────────────────────
                                    ↑
                       Unique constraint for the table
```

■Table comment definition

A comment consisting of a character string can be specified for a table. A character string of up to 256 bytes can be specified. If no comment is necessary, omit the specification. An example follows.

Example:

Specify a character string comment for the STOCK table.

```
CREATE TABLE STOCK (ITMNO SMALLINT NOT NULL, ...
                    COMMENT  'STOCK ITEMS, STOCK QUANTITIES, AND WAREHOUSES TABLE'
                    ───────────────────────────────────────────────────────────────
                                             ↑
                                      Comment definition
```

2.6.4 Table definition for multimedia data storage

This section explains how to define a table that stores data types such as image and voice. This type of data is stored in a BLOB-type column.

To define a BLOB-type column of 31 kilobytes or more, specify SEQUENTIAL or OBJECT as the table storage structure.

If OBJECT is used as the data storage structure, the following conditions are added to the definition of a table for storing numeric values and characters.

1. Only one BLOB-type column for data exceeding 31 kilobytes can be specified, and the column must be specified as the last column in the table.
2. The NOT NULL constraint must be specified for the column described in item 1.
3. The data type for columns other than the column described in item 1. must be fixed length.
4. An ALTER TABLE statement for changing a table definition cannot be described in item 1.

For more information on storage structure, refer to "2.7 Definition of Storage Structure."

Example:

The following is an example in which the PRODPHOT table is defined in schema S1 when SEQUENTIAL is used as the data storage structute. In this sample, ITMNO is defined as a column for non-BLOB-type data. Next, PRODPHOTO is defined as a column for one-megabyte BLOB-type data.

```
CREATE TABLE  S1.PRODPHOT (ITMNO     SMALLINT PRIMARY KEY NOT NULL,
                           PRODPHOTO BLOB(1M))
```

2.6.5 View definition

Define a view using the CREATE VIEW statement. Views are used to simplify data manipulations by application programs and to join multiple tables and process them as a single table. Views are also used to increase the independence of application programs and data.

A view definition defines the following items:

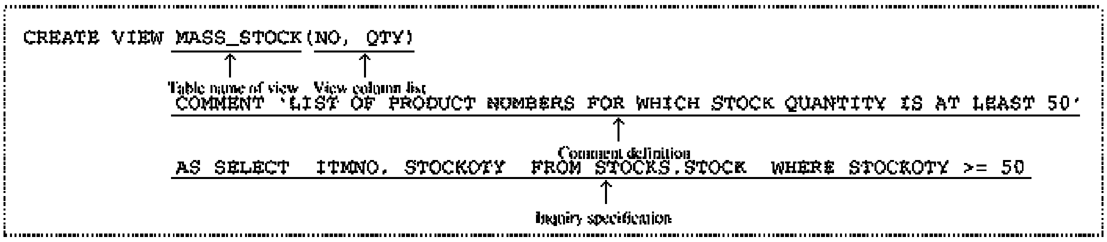
· Table name (view name)
· View column list

  - Column name
  - Column comment definition
- Query specification
- Table (view) comment definition

A sample view definition for the inventory management database follows. This view definition defines a view consisting of the rows of the ITMNO and STOCKQTY columns of the STOCK table for which STOCKQTY is at least 50.

Example:

CREATE VIEW statement that defines the MASS_STOCK view



```
CREATE VIEW MASS_STOCK(NO, QTY)
            COMMENT 'LIST OF PRODUCT NUMBERS FOR WHICH STOCK QUANTITY IS AT LEAST 50'
            AS SELECT  ITMNO, STOCKQTY  FROM STOCKS.STOCK  WHERE STOCKQTY >= 50
```

■Table name (view name)

Specify a name to be assigned to the view. For the table name, specify up to 36 alphanumeric characters beginning with an alphabetic character.

A view name is unique in a schema.

Example:

Sample view name specification

```
CREATE VIEW  MASS_STOCK ( ...
CREATE VIEW  L.STOCKS(...
```

■View column list

Specify column names for the columns that form the view.

◆Column name

Specify names for each of the columns that form the view. For the column name, specify up to 36 alphanumeric characters beginning with an alphabetic character.

A column name is unique in a view.

Example:

Define a view having column names NO and QTY.

```
CREATE VIEW  MASS_STOCK (NO, QTY)
```

◆Column comment definition

A comment consisting of a character string can be specified for each column in the view. A character string of up to 256 bytes can be specified. If no comment is necessary, omit the specification. An example follows.

Example:

Specify a comment for the NO column in the MASS_STOCK view.

```
CREATE VIEW MASS_STOCK (NO COMMENT ‘PRODUCT-NO’, ...)
                                     ↑
                            Comment definition
```

■Query specification

The query specification indicates which portion of the base table forms the view.

Example:

Define the view named MASS_STOCK. Let the ITMNO and STOCKQTY columns of the STOCK table be the NO and QTY columns of the view, respectively.

```
CREATE VIEW  MASS_STOCK (NO, QTY)
             AS SELECT  ITMNO, STOCKQTY  FROM STOCKS.STOCK
```

■Table (view) comment definition

A comment consisting of a character string can be specified for a view. A character string of up to 256 bytes can be specified. If no comment is necessary, omit the specification. An example follows.

Example:

Specify a character string comment for the MASS_STOCK view.

```
CREATE VIEW MASS_STOCK (NO, QTY) ...

         COMMENT ‘LIST OF PRODUCT NUMBERS FOR WHICH STOCK QUANTITY IS AT LEAST 50’
                                         ↑
                                 Comment definition
```

2.6.6 Trigger definition

Define a trigger using the CREATE TRIGGER statement. In a trigger definition, specify a trigger event and a procedure to be started. As the trigger event, specify a table data update method that starts the trigger. As the procedure to be started, specify a procedure to be processed by a triggered SQL statement.

Triggers are classified into the following groups according to the purpose of the trigger:

- Simple trigger
- Update-and-add trigger
- Trigger for calling a procedure routine

■Simple trigger

The user can define a simple operation, such as insertion of data into another table, deletion of data from another table, or updating of data in another table, so that the operation is automatically executed when a table is updated. Specify a simple INSERT, UPDATE, or DELETE statement as the triggered SQL statement.

The use can only define simple operations. However, the definition is easy because it is not required to define a procedure routine.

The table updated using the SQL statement that is the source of the trigger cannot be updated using the triggered SQL statement.

■Update-and-add trigger

For a row added to a database using the INSERT statement or updated using the UPDATE statement, the user can use a trigger to automatically arrange data of the row into columns. That is, a table updated by an SQL statement that causes the start of a trigger can be updated in processing of the triggered SQL statement.

For example, the timestamp indicating the time when a column in a row has been updated can be placed in another column, and the name of the user who updated the row can be placed in another column too. This type of trigger is called an update-and-add trigger.

■Trigger for calling a procedure routine

The user can call a procedure routine by specifying the CALL statement in the triggered SQL statement.

By using the procedure routine, the user can define a procedure that consists of multiple SQL statements as a triggered SQL statement. In addition, the user can use a trigger to define data integrity and consistency that is customized according to the user's desired application requirements.

Using triggers can simplify application programs and construct a highly reliable system because the system can automatically set information in tables and databases as described above.

Sample trigger definitions follow.

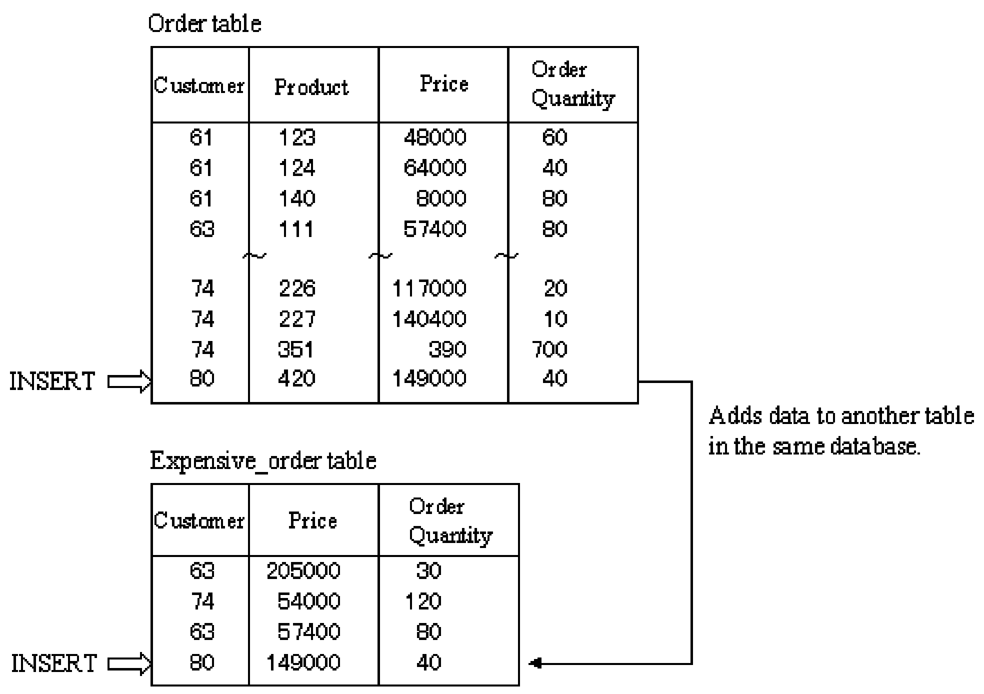
Example 1: Simple trigger

If the order price in a row added to the ORDER table exceeds five million, the ORDER_TRIGGER trigger adds the customer name, product price, and order quantity to the EXPENSIVE_ORDER table.

Sample trigger definition

```
CREATE TRIGGER  ORDER_TRIGGER
  AFTER INSERT ON  ORDER
  REFERENCING NEW AS NEWREC
  FOR EACH ROW
    WHEN(NEWREC.PRICE*NEWREC.ORDERQTY > 5000000)
    INSERT INTO  EXPENSIVE_ORDER
    VALUES(NEWREC.CUSTOMER, NEWREC.PRICE,NEWREC.ORDERQTY)
```

Operation



Order table

Customer	Product	Price	Order Quantity
61	123	48000	60
61	124	64000	40
61	140	8000	80
63	111	57400	80
~	~	~	~
74	226	117000	20
74	227	140400	10
74	351	390	700
80	420	149000	40

Expensive_order table

Customer	Price	Order Quantity
63	205000	30
74	54000	120
63	57400	80
80	149000	40

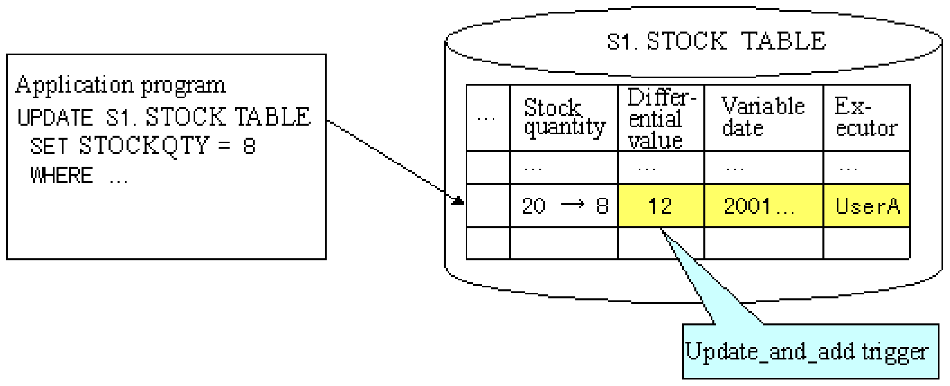
Example 2: Update-and-add trigger

This trigger sets the differential value, variable date and time, and executor name if a stock quantity in the STOCK table decreases by 10 or more.

Sample trigger definition

```
CREATE TRIGGER S1. UPDATE_ADD_TRIGGER
 AFTER UPDATE OF STOCKQTY  ON S1. STOCK TABLE
 REFERENCING NEW AS NEWREC OLD AS OLDREC
 FOR EACH ROW
 WHEN (OLDREC. STOCKQTY-NEWREC. STOCKQTY >= 10)
 UPDATE S1. STOCK TABLE  SET  DEFFERENTIAL_VALUE=OLDREC. STOCKQTY-NEWREC. STOCKQTY
              , VARIABLE_DATE = CURRENT_TIMESTAMP
              , EXECUTOR = CURRENT_USER
WHERE ROW_ID = NEWREC.ROW_ID
```

Operation



Application program
UPDATE S1. STOCK TABLE
 SET STOCKQTY = 8
 WHERE …

S1. STOCK TABLE

…	Stock quantity	Differ-ential value	Variable date	Ex-ecutor
	…	…	…	…
	20 → 8	12	2001 …	UserA

If this trigger is defined and an SQL statement that updates the STOCK table is executed, the update operation specified in the triggered SQL statement is automatically executed. To define an update-and-add trigger, specify ROW_ID in the WHERE clause in the triggered SQL statement.

Example 3: Trigger for calling a procedure routine

When a line is entered to the ORDER table, the trigger "ORDER trigger 2" calls a procedure routine "ORDER routine." The ORDER routine checks the consistency of the entered data. Then, it changes a stock quantity in the STOCK table according to the ordered quantity.

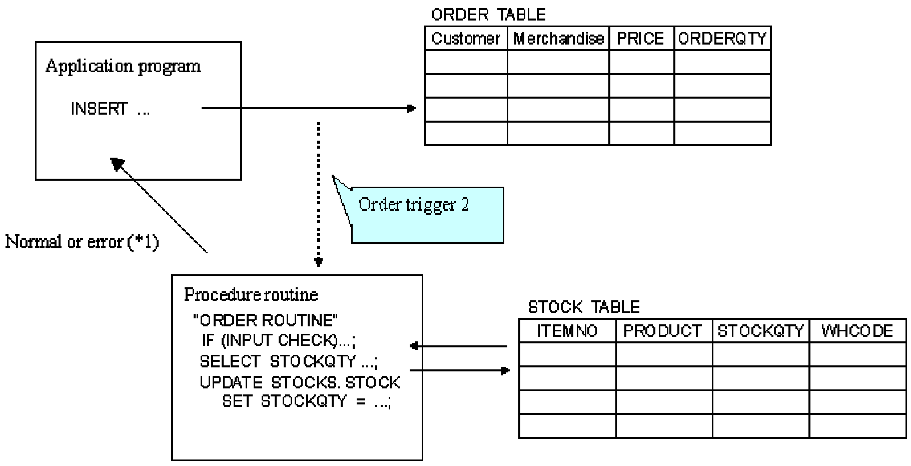
Sample procedure routine definition

```
CREATE PROCEDURE STOCKS.ORDER ROUTINE  (
    IN NEW MERCHANDISE    SMALLINT,
    IN NEW ORDERQTY       SMALLINT
)
BEGIN
    -- SQL VARIABLE DECLARATION
    DECLARE SQLSTATE     CHAR(5);
    DECLARE SQLMSG       CHAR(256);
    DECLARE POINT        SMALLINT;
    DECLARE STOCKQTY   INTEGER;
    -- CONDITION DECLARATION
    DECLARE INPUT ORDERQTY_INVALID   CONDITION FOR SQLSTATE' 60001';
    DECLARE INPUT INTEMNO_INVALID    CONDITION FOR SQLSTATE' 60002';
    DECLARE STOCK_INSUFFICIENT       CONDITION FOR SQLSTATE' 60003';
    DECLARE OTHER                    CONDITION FOR SQLSTATE' 60999';
    -- HANDLER DECLARATION
    DECLARE EXIT HANDLER FOR NOT FOUND
    BEGIN
        IF (POINT = 20 OR POINT = 30)  THEN
            RESIGNAL INPUT ITEMNO_INVALID
                     'ORDER: INPUT ERROR: VALUE OF MERCHANDISE IS INVALID';
        ELSE
            RESIGNAL OTHER ERROR 'ORDER: ERROR:
                              AN UNEXPECTED DATA_NOT_FOUND ERROR OCCURRED';
        END IF;
    END;
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN



        RESIGNAL; -- RETURN THE OCCURRED ERROR AS IT IS.
    END;
    -- THIS PROCESSING
    -- (1) INPUT CHECK
    SET POINT = 10;
    IF (NEWORDERQTY < 1)  THEN
        SIGNAL INPUT ORDERQTY_INVALID 'ORDER: INPUT ERROR: ORDERQTY_INVALID'
    END IF;
    -- (2) STOCKQTY CHECK
    SET POINT = 20;
    SELECT STOCKQTY INTO STOCKQTY FROM STOCKS. STOCK
        WHERE ITEMNO = NEW MERCHANDISE
        WITH OPTION LOCK_MODE(EXCLUSIVE LOCK);
    IF (NEW ORDERQTY > STOCKQTY)  THEN
        SIGNAL STOCK_INSUFFICIENT 'ORDER: ERROR: STOCK_INSUFFICIENT';
    END IF;
    -- (3) ORDER
    SET POINT = 30;
    UPDATE STOCKS. STOCK
        SET STOCKQTY = STOCKQTY - NEW ORDERQTY
        WHERE ITEMNO = NEW MERCHANDISE;
END
```

Sample trigger definition

```
CREATE TRIGGER STOCKS. ORDER TRIGGER 2
    AFTER INSERT ON STOCKS. ORDER
    FOR EACH ROW
    CALL STOCKS. ORDER ROUTINE (NEW. MERCHANDISE, NEW. ORDERQTY);
```

Operation



*1 If the order routine detects an invalid input value, the following error occurs in the INSER statement.
JYP1065E An error occurred in the triggered SQL statement of the trigger "order trigger 2" of the schema "STOCKS." Detail message = JYP2550E An exception is sent out from the SIGNAL statement.
An exception message = "ORDER: ERROR: STOCK_INSUFFICIENT."

As described above, if a procedure routine and a trigger are defined when an SQL statement that updates the table is executed, the procedure routine specified in the triggered SQL statement is automatically executed to suppress any updating that may impair data consistency. The user can define a trigger that checks table for every linked row of the rows that reference one another in the procedure routine sot that data integrity between the tables can be ensured. The user can also define a trigger that automatically deletes corresponding data in child tables when a row in the parent table is deleted.

However, do not define any complex transaction logic with a trigger. A trigger operates as an extension of table updates that start the trigger. No transaction can be controlled in any trigger. Create complex transaction logic with stored procedures, and specify that application program are to call procedure routine directly.

■Trigger operation

A trigger operates as follows.

- Triggers are executed in the following sequence. Each chained trigger is executed in units of rows, following this same rule.
    1. Executes an SQL statement that updates the table for which a trigger is defined.
    2. Loops the row affected by the SQL statement.
    3. Moves to the row to be updated or deleted by the SQL statement.
    4. Executes the defined BEFORE trigger.
    5. Updates, deletes, or inserts the row according to the SQL statement in step 1).
    6. Executes the defined AFTER trigger.
- If a trigger event is defined more than once for the same table, triggers are executed in no particular sequence. Consider the execution sequence to not have an effect on triggers specified with the same trigger operation point.
- When data load (rdbsloader command) or database reinitialization (rdbfmt command) is executed using the utility function, any trigger defined for the execution target table does not operate.
- To update the table updated by an SQL statement that starts a trigger in the triggered SQL statement, use an update-and-add trigger. To define such an update-and-add trigger, specify the UPDATE statement in which ROW_ID is specified in the WHERE condition of the triggered SQL statement. For this trigger, specify AFTER as the trigger operation point and INSERT or UPDATE as the trigger event. With another type of trigger, the table updated by an SQL statement that starts the trigger cannot be updated in processing of the triggered SQL statement.

- Chained triggers cannot be executed by updating the triggered SQL statement specified by an update-and-add trigger.
- If the trigger defined for the table updated by the triggered SQL statement is a trigger other than an update-and-add trigger, a chain of triggers is executed. If a trigger is executed again as an extension of its own execution processing, an error occurs.
- A triggered SQL statement operates as the transaction that executes the SQL statement starting the trigger. If LOCK_MODE is specified in the SQL statement that starts a trigger, the triggered SQL statement operates, assuming that the same LOCK_MODE is implicitly specified in each SQL data manipulation statement executed by the triggered SQL statement.
- When a database is updated by a triggered SQL statement, the unique and NOT NULL constraints defined for an updated table are checked in each SQL statement. If the CALL statement is specified, the constraints are checked in each SQL statement defined in the CALL statement.
- If an error occurs in a triggered SQL statement, execution of the SQL statement that has started the trigger is canceled with an error. If the CALL statement is specified as the triggered SQL statement, an error may occur during execution of an SQL statement in the called procedure routine. In this event, execution of SQL statements can continue in accordance with the error handling method specified in the procedure routine. Alternatively, execution of the CALL statement can be assumed to cause the error, and execution of the SQL statement that starts the trigger can be canceled with an error.
- If the CALL statement is specified as a triggered SQL statement, a transaction rollback exception may occur in the called procedure routine (SQLSTATE exception code: 40). In this event, the transaction containing the SQL statement that starts the trigger is automatically rolled back.

■Notes on defining a trigger

Note the following points about defining a trigger.

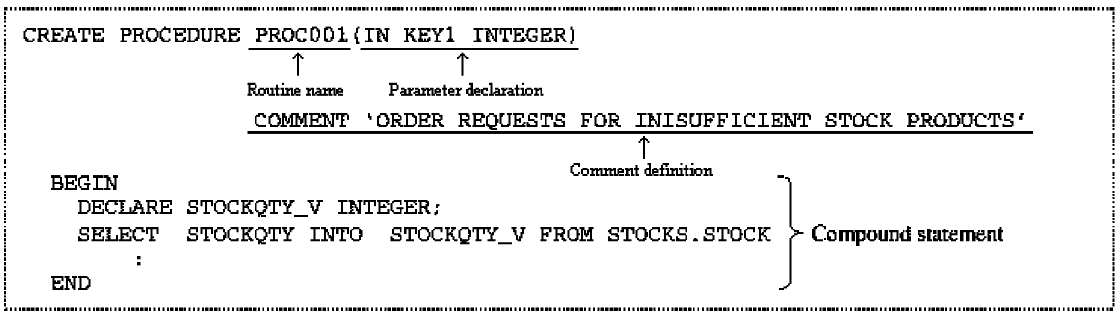
- If a subquery is specified in an SQL statement that starts a trigger, the table specified by the subquery cannot be updated by the triggered SQL statement.
- If a row with a unique constraint in the table has been updated using the search routine of the UPDATE statement, using an SQL statement for updating multiple rows may temporarily cause some data to not be unique. When data is not unique, the row whose updating causes the start of the trigger cannot be referenced temporarily in the extension of the CALL statement specified as the triggered SQL statement of the executed trigger. For this reason, if the procedure routine called by the CALL statement must reference the row whose update causes the start of the trigger, specify the appropriate arguments of the CALL statement to pass the necessary row values.
- Privileges required when a trigger is defined
  The user who wants to define a trigger must have the CREATE privilege for the schema for which the trigger is defined, TRIGGER privilege for the table for which the trigger is defined, and privilege corresponding to the SQL operation specified in the triggered SQL statement.
  - To specify the INSERT statement: User must have the INSERT privilege for the table specified in the triggered SQL statement.
  - To specify the DELETE statement: User must have the DELETE privilege for the table specified in the triggered SQL statement.
  - To specify the UPDATE statement: User must have the UPDATE privilege for the table specified in the triggered SQL statement.
  - To specify the CALL statement: User must have the execution privilege for the procedure routine specified in the triggered SQL statement.
- Privilege check when a triggered SQL statement is executed
  When an SQL statement starts a trigger and the triggered SQL statement is executed, no privilege check is performed for the triggered SQL statement. The user who executes the application program does not need the privilege for the table specified in the triggered operation.

2.6.7 Procedure routine definition

Define a procedure routine using the CREATE PROCEDURE statement. For details about the procedure function, refer to the "RDB User's Guide: Application Program Development." A sample definition that sets procedure PROC001 in the STOCKMN_DB database follows.

Example:

Define PROC001.



```
CREATE PROCEDURE PROC001(IN KEY1 INTEGER)
                 COMMENT 'ORDER REQUESTS FOR INISUFFICIENT STOCK PRODUCTS'
  BEGIN
    DECLARE STOCKQTY_V INTEGER;
    SELECT  STOCKQTY INTO  STOCKQTY_V FROM STOCKS.STOCK
        :
  END
```

◆**Procedure comment definition**

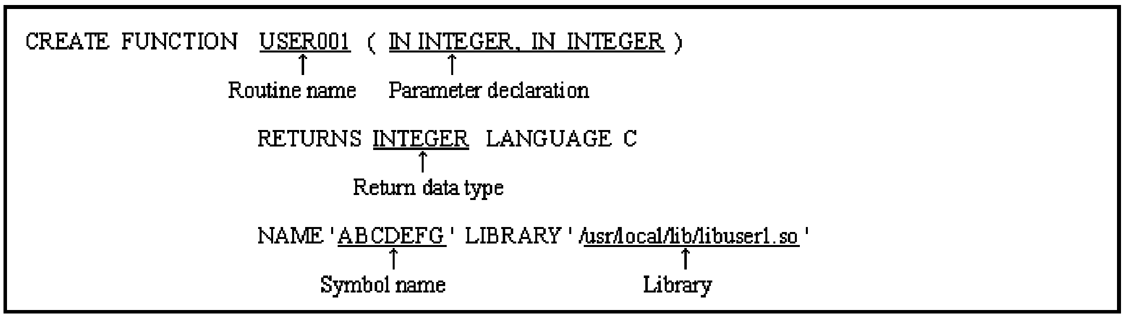
A comment consisting of a character string can be specified for a procedure routine. A character string of up to 256 bytes can be specified. If no comment is necessary, omit the specification.

2.6.8 Function routine definition

Define a function routine using the CREATE FUNCTION statement. For details of the function routine function, refer to the "RDB User's Guide: Application Program Development." A sample definition for user-created function routine USER001 follows.

Example:

Sample definition for user-created function routine USER001



```
CREATE FUNCTION  USER001  ( IN INTEGER, IN INTEGER )
                  RETURNS INTEGER  LANGUAGE C
                  NAME 'ABCDEFG' LIBRARY '/usr/local/lib/libuser1.so'
```

2.7 Defining a Storage Structure

After the logical structure has been defined, define the storage structure. The storage structure definition is then stored in the RDB dictionary.

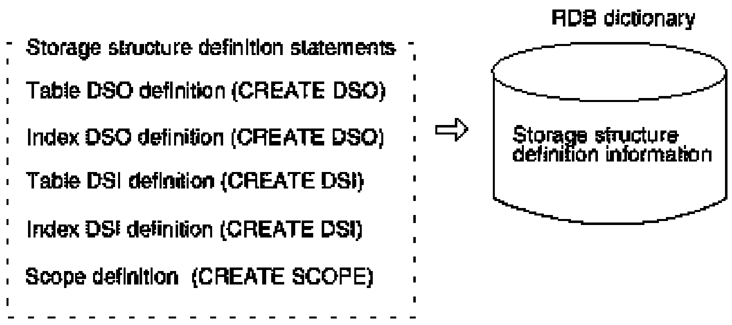
The two kinds of storage structure definitions are DSO definitions and DSI definitions. A table DSO and a table DSI are defined for a table created by a logical structure definition. In addition, if an index is to be defined for a table, an index DSO and an index DSI are defined.

A storage structure is defined by executing a DSO definition statement and a DSI definition statement.

For details about storage structures, see Chapter 4 "Storage Structure."

Figure: Storage structure definition procedure shows the storage structure definition procedure.

[Figure: Storage structure definition procedure]


Storage structure definition statements
Table DSO definition (CREATE DSO)
Index DSO definition (CREATE DSO)
Table DSI definition (CREATE DSI)
Index DSI definition (CREATE DSI)
Scope definition (CREATE SCOPE)
RDB dictionary
Storage structure definition information

■DSO definition

The DSO definition specifies the type of storage structure for storing data and rules such as whether to apply split table operation.

The two types of DSO definitions are table DSO definitions and index DSO definitions.

◆Table DSO definition

A table DSO definition specifies the type of storage structure for storing data and rules such as whether to apply split table operation for a base table. Use the CREATE DSO statement to specify a table DSO definition using.

The storage structures for base tables can be categorized according to the size of the stored data. The storage structures for handling character and numeric data are different from the storage structures for handling multimedia data. For information about handling multimedia data, see 2.7.2 "Table DSO definition for multimedia data storage."

◆Index DSO definition

An index DSO definition defines the columns that form the index and information such as the type of storage structure for storing index data. Use the CREATE DSO statement to specify an index DSO definition.

■DSI definition

A DSI definition specifies an association with the database space where the data is actually stored and information such as split key values for performing split table operation.

The two types of DSI definitions are table DSI definitions and index DSI definitions.

◆Table DSI definition

A table DSI definition specifies an association with a database space according to a table DSO definition. The table DSI definition also specifies information such as split key values for performing split table operation. Use the CREATE DSI statement to specify a table DSI definition.

◆Index DSI definition

An index DSI definition specifies an association with a database space according to an index DSO definition. Use the CREATE DSI statement to specify an index DSI definition.

Storage structure definition statements consist of table and index DSO definition statements and DSI definition statements. Details about these definition statements are explained in 2.7.1 "Table DSO definition," 2.7.3 "Index DSO definition," 2.7.4 "Table DSI definition," and 2.7.5 "Index DSI definition."

■Scope definition

Scope defines the range of table data that is to be manipulated. The user can specify a range in advance so that only data within that range is processed. Application and release of the scope can be specified for each user.

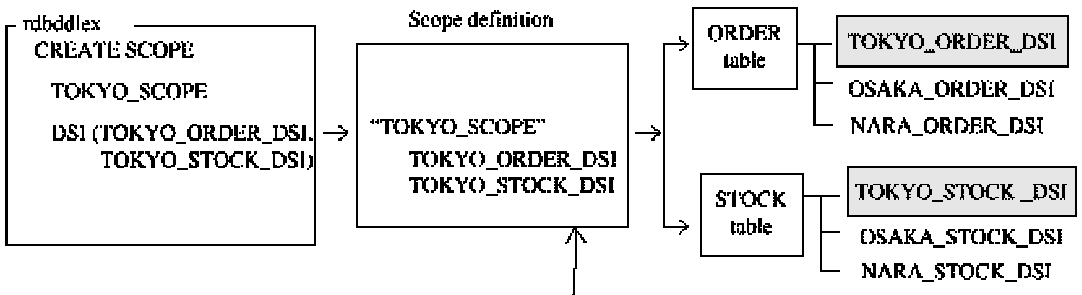
A sample of using scope follows.

Example:

> Define scope TOKYO, apply it to user SUZUKI, and retrieve a limited table from an application program.

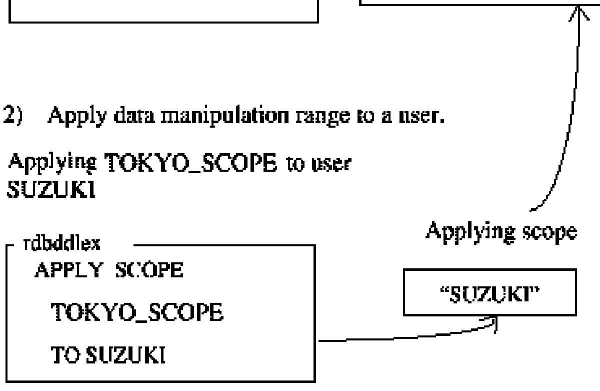
1) Define data manipulation range.

Limiting TOKYO_SCOPE to TOKYO_ORDER_DSI and TOKYO_STOCK_DSI

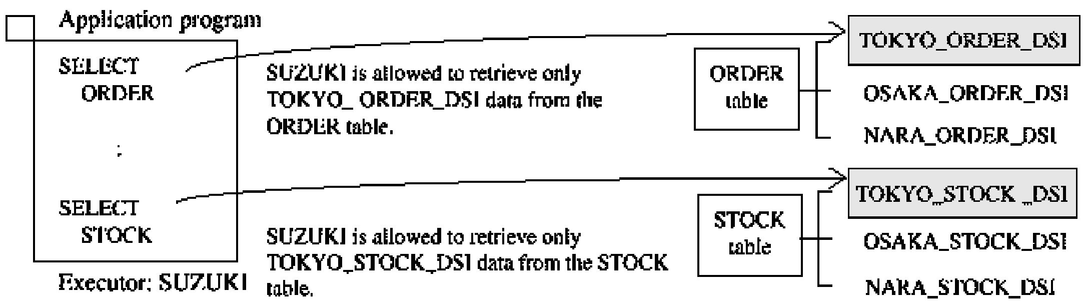


2) Apply data manipulation range to a user.

Applying TOKYO_SCOPE to user SUZUKI



3) Retrieve a range of the table limited by application program.



2.7.1 Table DSO definition

Use the CREATE DSO statement to specify a table DSO definition.

Sample table DSO definitions follow.

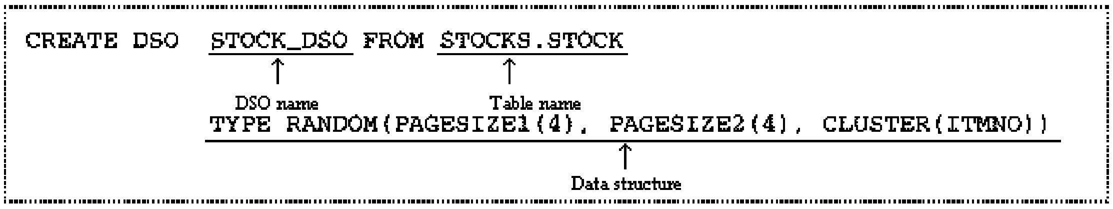
Example:

Table DSO definitions

· When split table operation is not applied (data structure: SEQUENTIAL)

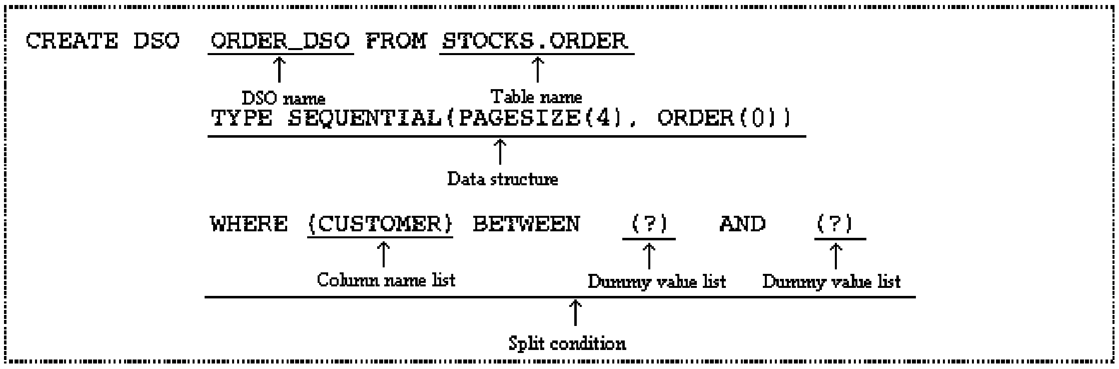
```
CREATE DSO  STOCK_DSO FROM STOCKS.STOCK
                ↑                ↑
            DSO name         Table name
            TYPE SEQUENTIAL(PAGESIZE(4), ORDER(0))
                             ↑
                        Data structure
```

· When split table operation is not applied (data structure: RANDOM)



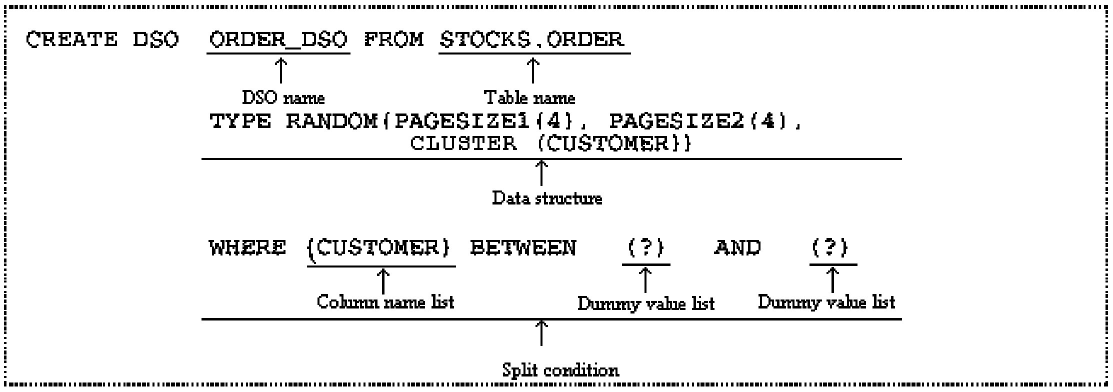
```
CREATE DSO  STOCK_DSO FROM STOCKS.STOCK
            TYPE RANDOM(PAGESIZE1(4), PAGESIZE2(4), CLUSTER(ITMNO))
```

· When split table operation is applied (data structure: SEQUENTIAL)



```
CREATE DSO  ORDER_DSO FROM STOCKS.ORDER
            TYPE SEQUENTIAL(PAGESIZE(4), ORDER(0))
            WHERE {CUSTOMER} BETWEEN  (?)  AND  (?)
```

· When split table operation is applied (data structure: RANDOM)



```
CREATE DSO  ORDER_DSO FROM STOCKS.ORDER
            TYPE RANDOM(PAGESIZE1(4), PAGESIZE2(4),
                      CLUSTER (CUSTOMER))
            WHERE {CUSTOMER} BETWEEN  (?)  AND  (?)
```

■DSO name

For the DSO name, specify up to 36 alphanumeric characters beginning with an alphabetic character. The DSO name must be unique within the database.

■Table name

Specify the name of the base table corresponding to the structure definition. The table name must be qualified by the schema name.

■Data structure

Specify information about the data structure to be used when storing table data. SEQUENTIAL or RANDOM can be specified when a table storage structure is defined by a DSO definition.

◆SEQUENTIAL

When SEQUENTIAL is specified, added data is stored in the order of addition.

Specify the page length at PAGESIZE. Specify ORDER(0) not to reuse a deleted area or ORDER(1) to reuse it. For more information, see "4.1.1 SEQUENTIAL structure."

◆RANDOM

When RANDOM is specified, added data is stored in a random order.

For RANDOM, use CLUSTER to specify a key for determining the data storage position. If CLUSTER is omitted, the data is stored on the basis of the arrangement of PRIMARY KEY in the table definition.

For PAGESIZE1 and PAGESIZE2, specify the page sizes of the data structure elements (PRIME and OVERFLOW for a RANDOM structure).

A RANDOM structure allows a data storage position to be determined by specifying RULE. For more information, see "4.1.1 RANDOM structure."

■Split condition

When split table operation is to be applied, specify the split condition. The DSI definition statement specifies the actual subdivision units. Here, specify only the condition used for splitting.

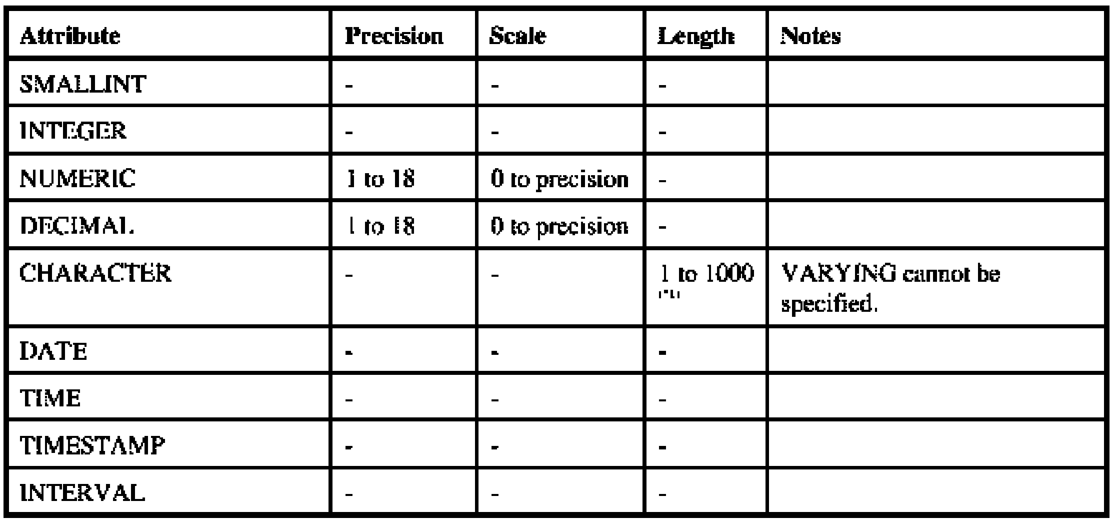
The rows stored in each subdivision unit is determined as follows. The values set in each column specified in the column name list determine these rows. In addition, the result of assigning the following values determines the rows. These values are the values specified by the DSI definition statement for the question marks (?) in the dummy value list.

If a split condition is specified in a DSO definition statement and if the data structure is RANDOM, the column name list specified in the split condition must be included in the column name list specified in CLUSTER.

Up to 64 columns can be specified in the column name list. The number of column names in the column name list must be the same as the number of question marks (?) in each dummy value list.

Table: Data types of columns that can be specified for CLUSTER and column name list shows the data types of the columns that can be specified for CLUSTER and the column name list.

[Table: Data types of columns that can be specified for CLUSTER and column name list]



Attribute	Precision	Scale	Length	Notes
SMALLINT	-	-	-	
INTEGER	-	-	-	
NUMERIC	1 to 18	0 to precision	-	
DECIMAL	1 to 18	0 to precision	-	
CHARACTER	-	-	1 to 1000 [*1]	VARYING cannot be specified.
DATE	-	-	-	
TIME	-	-	-	
TIMESTAMP	-	-	-	
INTERVAL	-	-	-	

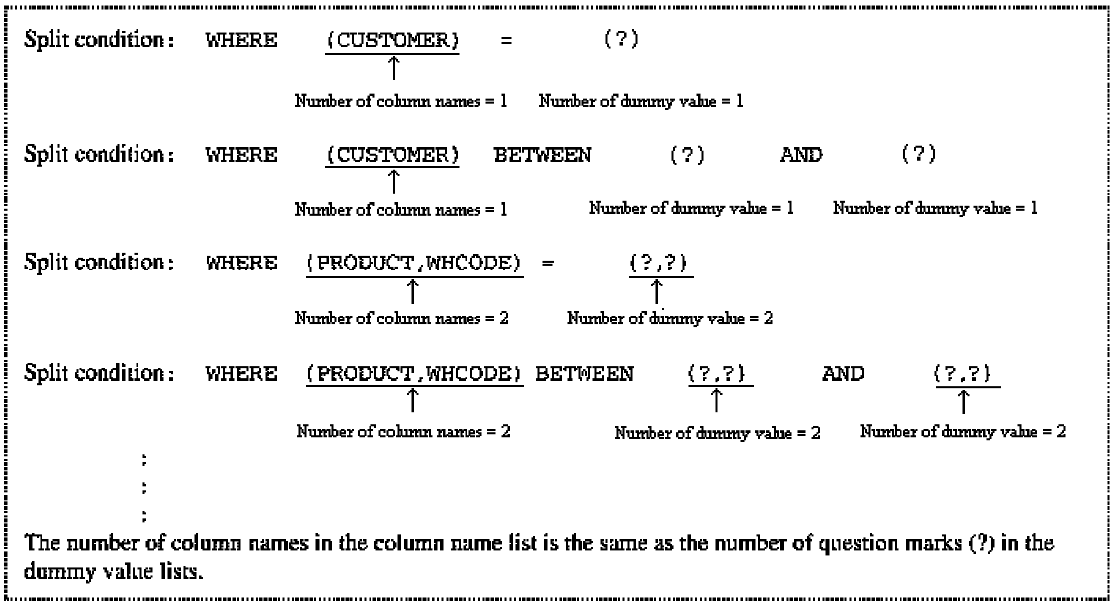
*1 When specifying the split condition, specify a length from 1 to 254.

NOT NULL must be specified for the CLUSTER columns and the column name list. The column name list specified by the split condition is called the split key.

A sample DSO definition for the ORDER table with CUSTOMER and PRODNO of the ORDER table as a composite split key follows.

Example 1:

Relationship between the number of column names in the column name list and the number of question marks (?) in the dummy value lists



```
Split condition:  WHERE  (CUSTOMER)  =  (?)
Split condition:  WHERE  (CUSTOMER)  BETWEEN  (?)  AND  (?)
Split condition:  WHERE  (PRODUCT,WHCODE)  =  (?,?)
Split condition:  WHERE  (PRODUCT,WHCODE) BETWEEN  (?,?)  AND  (?,?)
        :
        :
        :
```

Number of column names = 1
Number of dummy value = 1
Number of column names = 1
Number of dummy value = 1
Number of dummy value = 1
Number of column names = 2
Number of dummy value = 2
Number of column names = 2
Number of dummy value = 2
Number of dummy value = 2

The number of column names in the column name list is the same as the number of question marks (?) in the dummy value lists.

Example 2:

Table DSO definition with multiple columns as the split key

```
CREATE DSO  ORDER_DSO2 FROM STOCKS.ORDER
            TYPE SEQUENTIAL (PAGESIZE1(4), ORDER(0))
            WHERE (CUSTOMER,PRODNO) = (?,?)
```

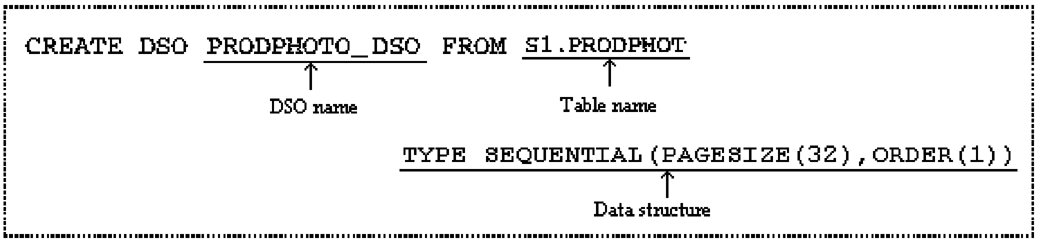
2.7.2 Table DSO definition for multimedia data storage

This section explains how to specify a table DSO definition for storing image or audio data. This data is stored in a BLOB-type column.

For storing data such as image and voice, the record length in a table may exceed the page length. In this case, specify SEQUENTIAL or OBJECT as the data structure. If OBJECT is specified, assign 32 to PAGESIZE. For the OBJECT type, split operation is not available, and the associated table definition is conditional. For more information on table definition conditions, refer to "2.6.4 Table Definition for Multimedia Data Storage." A sample definition follows.

Example:

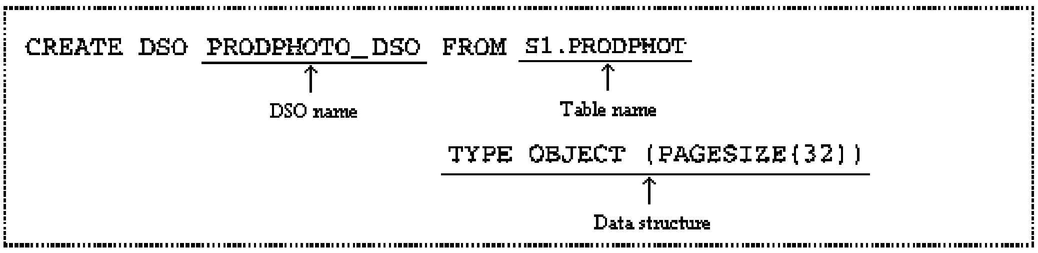
Table DSO definition for storing multimedia data (SEQUENTIAL)



```
CREATE DSO PRODPHOTO_DSO FROM S1.PRODPHOT
                 TYPE SEQUENTIAL(PAGESIZE(32),ORDER(1))
```

Example:

Table DSO definition for storing multimedia data (OBJECT)



```
CREATE DSO PRODPHOTO_DSO FROM S1.PRODPHOT
                   TYPE OBJECT (PAGESIZE(32))
```

2.7.3 Index DSO definition

Use the CREATE DSO statement to specify an index DSO definition. If a column is used for a conditional search, define an index for the column used in the search condition to improve the search efficiency.

If PRIMARY KEY or UNIQUE is specified in a table definition, an index DSO definition with the same column configuration is required. (The order is the same.) If RANDOM is specified for the table data structure, either PRIMARY KEY or UNIQUE can be associated with CLUSTER KEY. In this case, an index DSO definition is unnecessary.

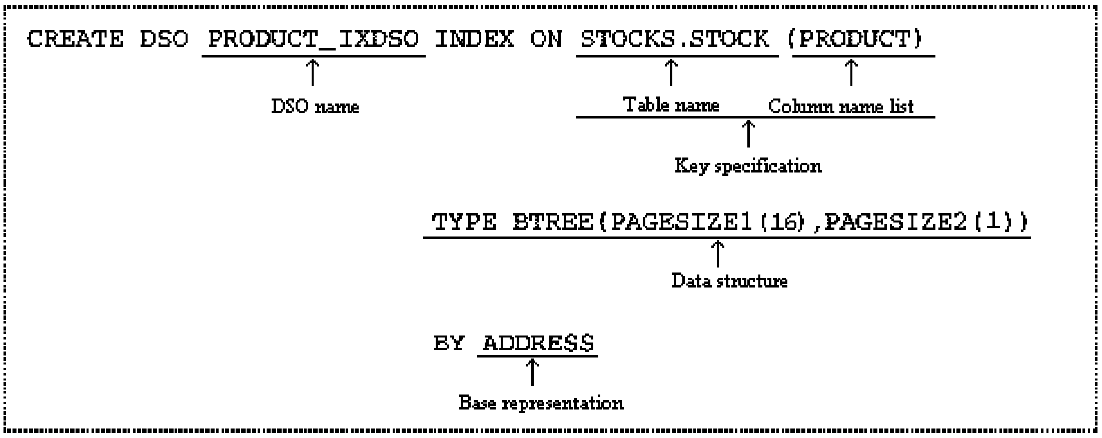
In addition, a new index DSO definition can be added for a table in which data has already been stored.

Sample index DSO definitions follow.

Example:

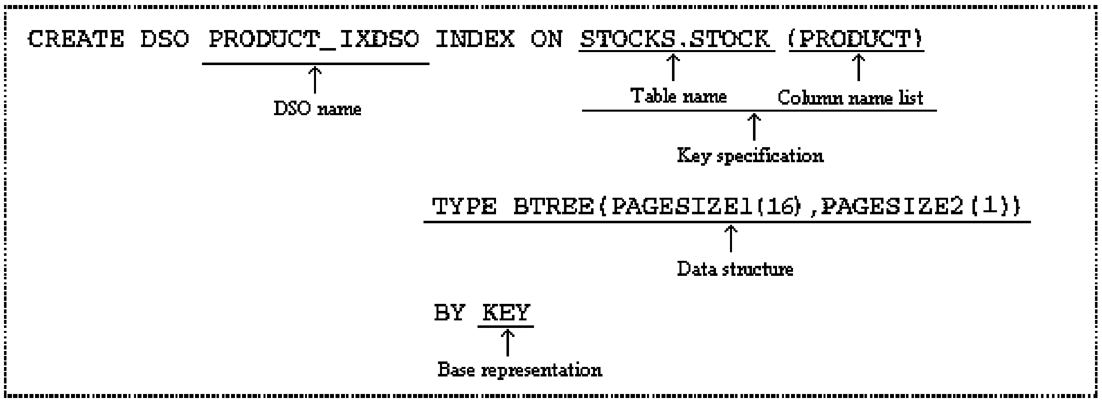
Index DSO definitions

- When the table data structure is SEQUENTIAL



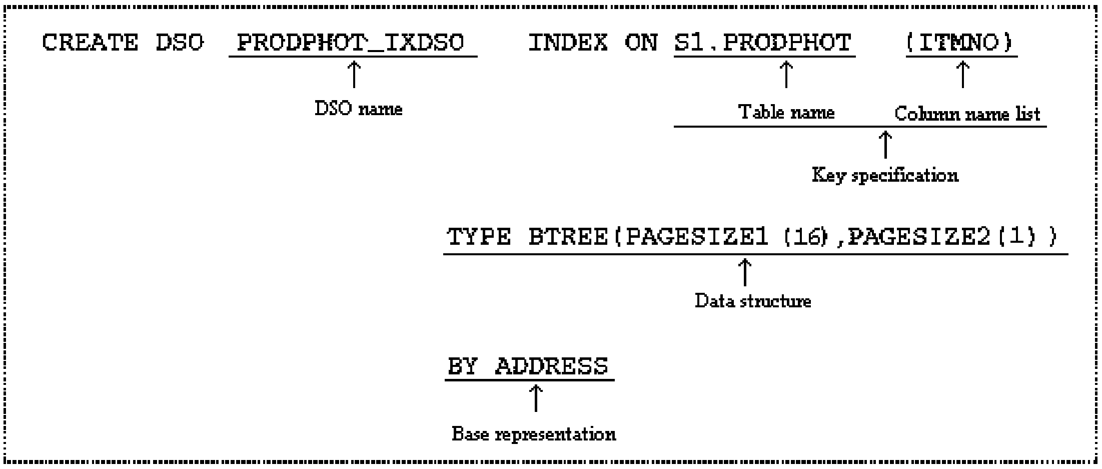
```
CREATE DSO PRODUCT_IXDSO INDEX ON STOCKS.STOCK (PRODUCT)
                     TYPE BTREE(PAGESIZE1(16),PAGESIZE2(1))
                     BY ADDRESS
```

- When the table data structure is RANDOM



```
CREATE DSO PRODUCT_IXDSO INDEX ON STOCKS.STOCK (PRODUCT)
                     TYPE BTREE(PAGESIZE1(16),PAGESIZE2(1))
                     BY KEY
```

- When the table data structure is OBJECT



```
CREATE DSO  PRODPHOT_IXDSO    INDEX ON S1.PRODPHOT    (ITMNO)
                TYPE BTREE(PAGESIZE1 (16),PAGESIZE2(1) )
                BY ADDRESS
```

■DSO name

For the DSO name, specify up to 36 alphanumeric characters beginning with an alphabetic character.

The DSO name must be unique within the database.

■Key specification

Specify the table name for which the index is to be created and the list of column names forming the index.

◆Table name

Specify the name of the base table for which the storage format is to be defined. The table name must be qualified by the schema name.

◆Column name list

Specify the column names for which the index is to be created. Table: Data types of columns that can be specified for the column name list of an index shows the data types of the columns that can be specified for the column name list of an index.

[Table: Data types of columns that can be specified for the column name list of an index]

Attribute	Precision	Scale	Length	Notes
SMALLINT	-	-	-	
INTEGER	-	-	-	
NUMERIC	1 to 18	0 to precision	-	
DECIMAL	1 to 18	0 to precision	-	
CHARACTER	-	-	1 to 1000	VARYING can be specified.
DATE	-	-	-	
TIME	-	-	-	
TIMESTAMP	-	-	-	
INTERVAL	-	-	-	

■Data structure

Specify information about the data structure to be used when index data is stored. Only BTREE can be specified.

◆BTREE

For PAGESIZE1, specify the page size of the data part.

For PAGESIZE2, specify the page size of the index part.

For details, refer to "4.2.1 BTREE Structure."

■Base representation

Specify the way the index and base are associated. If this specification is omitted, SEQUENTIAL or OBJECT default to ADDRESS , and RANDOM defaults to KEY.

ADDRESS:

The index and table records are associated according to the storage addresses of the table records. Specify ADDRESS when the table data structure is SEQUENTIAL or OBJECT.

KEY:

The index and table records are associated according to the cluster key of the table records. Specify KEY when the table data structure is RANDOM.

2.7.4 Table DSI definition

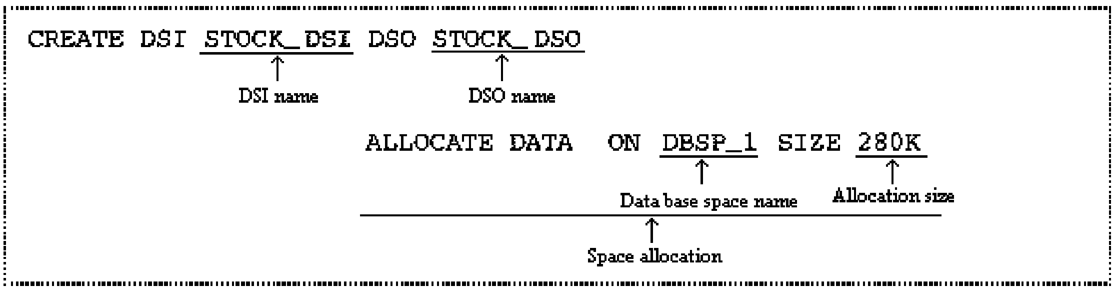
Use the CREATE DSI statement to specify a table DSI definition. The table DSI definition allocates database space according to the table DSO definition.

Sample table DSI definitions follow.

Example:

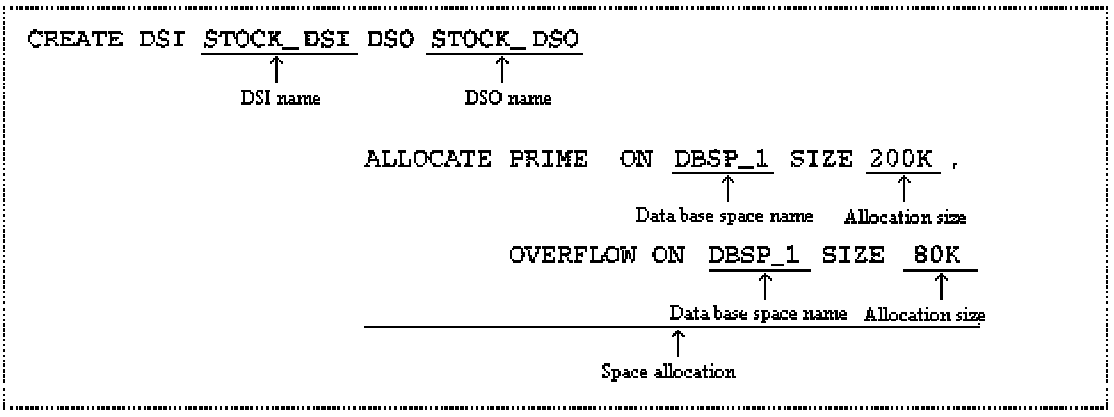
Table DSI definitions

· When split table operation is not applied (data structure: SEQUENTIAL)



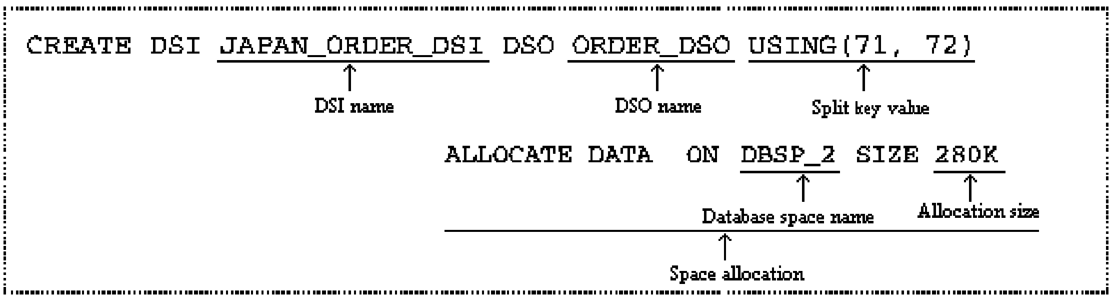
```
CREATE DSI STOCK_DSI DSO STOCK_DSO
                 ALLOCATE DATA  ON DBSP_1 SIZE 280K
```

· When split table operation is not applied (data structure: RANDOM)



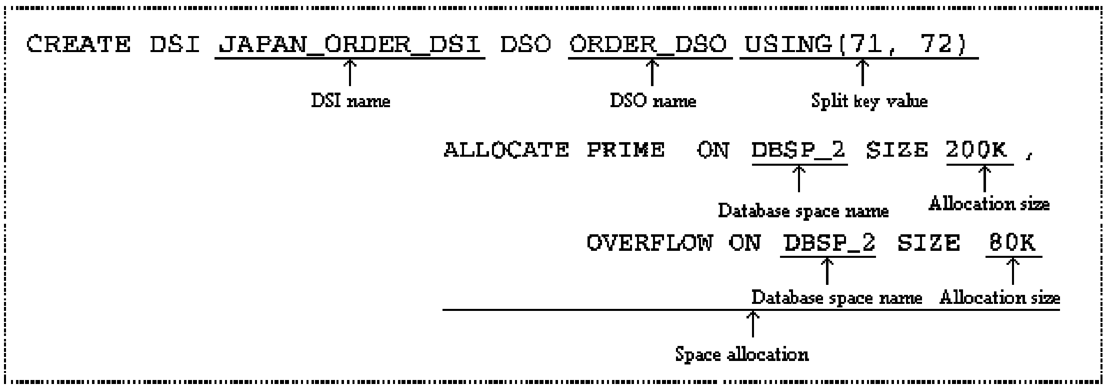
```
CREATE DSI STOCK_DSI DSO STOCK_DSO
                 ALLOCATE PRIME  ON DBSP_1 SIZE 200K ,
                      OVERFLOW ON DBSP_1 SIZE  80K
```

· When split table operation is applied (data structure: SEQUENTIAL)



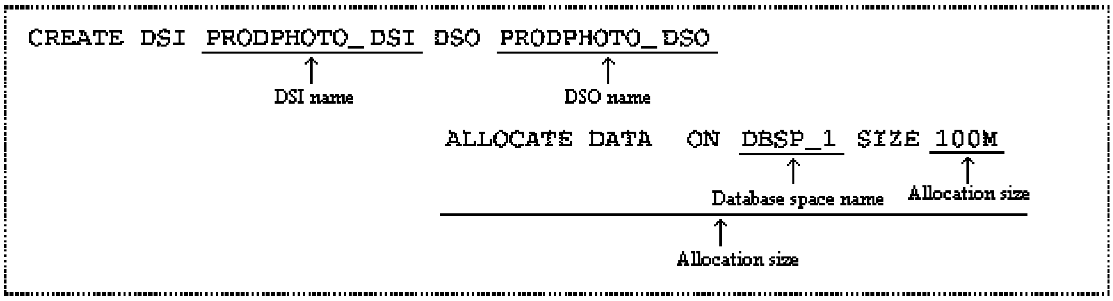
```
CREATE DSI JAPAN_ORDER_DSI DSO ORDER_DSO USING(71, 72)
                 ALLOCATE DATA  ON DBSP_2 SIZE 280K
```

· When split table operation is applied (data structure: RANDOM)



```
CREATE DSI JAPAN_ORDER_DSI DSO ORDER_DSO USING(71, 72)
                 ALLOCATE PRIME  ON DBSP_2 SIZE 200K ,
                      OVERFLOW ON DBSP_2 SIZE  80K
```

· When the table data structure is object



```
CREATE DSI PRODPHOTO_DSI DSO PRODPHOTO_DSO
                 ALLOCATE DATA  ON DBSP_1 SIZE 100M
```

■DSI name

For the DSI name, specify up to 36 alphanumeric characters beginning with an alphabetic character.

The DSI name must be unique within the database.

■DSO name

Specify the table DSO name given in a CREATE DSO statement.

With no split table operation, define only one DSI for one DSO definition. However, if split table operation is to be applied, define multiple DSIs. These DSIs specify split key values. The DSI definitions for split table operation must include all existing data.

■Split values

Specify the values for the split condition when split table operation is to be applied.

Specify the constant value for the question marks (?) in the dummy value list. The dummy value list is specified in the split condition of the table DSO definition. If the split condition includes multiple question marks (?), use commas to delimit the constant values in order of occurrence. The number of constant values must be the same as the number of question marks (?) specified in the split condition.

The user cannot specify split values to have the storage destination of a given row (data) include multiple DSIs. Table: Specification formats of constants that can be specified for split values shows the specification formats of constants that can be specified for split values.

[Table: Specification formats of constants that can be specified for split values]

Attribute of corresponding column	**Specification format of constant that can be specified for split value**
SMALLINT	-32768 to 32767 (no decimal point)
INTEGER	-2147483648 to 2147483647 (no decimal point)
NUMERIC(p, q)	Number of digits in the integer part (to the left of the decimal point) does not exceed p - q. Number of digits in the decimal part (to the right of the decimal point) does not exceed q. (*1)
DECIMAL(p, q)	
CHARACTER(n)	Character string constant having up to n characters
DATE	10-character date from years to days
TIME	8-character time from hours to seconds
TIMESTAMP	19-character time stamp from years to seconds
INTERVAL start-field TO end-field	Interval (years to months, or days to times) indicated by the field specifications (For details, see Table: Specification formats of time interval types that can be specified for split key values.)

n:	Number of characters
p:	Precision
q:	Scale
*1	If a constant does not contain a decimal point, all digits are included in the integer part.

[Table: Specification formats of time interval types that can be specified for split key values]

Type	Start field	End field	Explanation of specification
Year and month type	YEAR(p)	-	p-digit time interval indicating years
		MONTH	p-digit time interval indicating years and months
	MONTH(p)	-	p-digit time interval indicating months
Day and hour type	DAY(p)	SECOND	p-digit time interval indicating days. hours, minutes, and seconds
		MINUTE	p-digit time interval indicating days. hours, and minutes
		HOUR	p-digit time interval indicating days and hours
		-	p-digit time interval indicating days
	HOUR(p)	SECOND	p-digit time interval indicating hours, minutes, and seconds
		MINUTE	p-digit time interval indicating hours and minutes
		-	p-digit time interval indicating hours
	MINUTE(p)	SECOND	p-digit time interval indicating minutes and seconds
		-	p-digit time interval indicating minutes
	SECOND(p)	-	p-digit time interval indicating seconds

p: Precision of start field (Specify an integer from 1 to 9.)

■Space allocation

Specify the physical space to be allocated to the table DSI.

Specify the name of the database space where the data is to be physically stored. In addition, specify the size of the storage area to be acquired for this DSI within the database space. The acquired storage area is accessed according to the page size specified in the DSO definition.

■Database space name

Specify the name of the physical database space where the data is actually stored.

To store data in multiple database spaces for scalable log operation, specify their names in the same log group.

■Allocation size

Specify an unsigned integer combined with a unit symbol (K or M) to denote the size of the storage area acquired in the database space. The unit symbol K indicates kilobytes, and the unit symbol M indicates megabytes.

The units to be used for storage in the database space according to the table DSI are determined as follows. The split condition in the table DSO definition and the split key value specification determine these storage units. Some examples follow.

Example 1:

Relationship between the number of question marks (?) specified in the split condition and the number of constants in the split value specification

```
Split condition:   WHERE    (CUSTOMER)  =      (?)
                                                 ↑
                                        Number of dummy values = 1

Split value:       USING         (61)
                                  ↑
                         Number of constants = 1

Split condition:   WHERE   (CUSTOMER) BETWEEN (?)  AND  (?)
                                                   ↑
                                           Number of dummy values = 2

Split value:       USING        (61,71)
                                   ↑
                          Number of constants = 2

Split condition:   WHERE   (PRODUCT,WHCODE)  =     (?,?)
                                                     ↑
                                           Number of dummy values = 2

Split value:       USING   ('TELEVISION',2)
                                 ↑
                          Number of constants = 2

Split condition:   WHERE   (ITMNO,STOCKQTY)  BETWEEN (?,?)  AND  (?,?)
                                                          ↑
                                                 Number of dummy values = 4

Split value:       USING   (110,80,250,100)
                                  ↑
                           Number of constants = 4
              :
              :
              :
The number of constants is the same as the number of question marks (?) specified in the split condition.
```

Create a television DSI by splitting and storing STOCK table data according to PRODUCT and WHCODE values.

Example 2:

Create a television DSI for the STOCK table.

```
CREATE DSO  STOCK_DSO                    -- STOCK table DSO
       FROM STOCKS.STOCK
       TYPE SEQUENTIAL ( PAGESIZE(4),
                         ORDER(1))
       WHERE (PRODUCT,WHCODE) = (?,?)    -- Split data according to PRODUCT and WHCODE

CREATE DSI  STOCK_DSI_TV_2               -- STOCK table DSI for warehouse code 2
       DSO  STOCK_DSO
       USING  ('TELEVISION',2)           -- STOCK table for product TELEVISION and warehouse
       ALLOCATE DATA ON DBSP_1 SIZE 280K -- code 2
```

Create a JAPAN DSI by splitting and storing ORDER table data according to CUSTOMER region. The ORDER table CUSTOMER numbers are divided by region with the range 71 and 72 representing companies in JAPAN.

Example 3:

Create JAPAN DSI for the ORDER table.

```
CREATE DSO  ORDER_DSO                              -- ORDER table DSO
       FROM STOCKS.ORDER
       TYPE RANDOM  ( PAGESIZE1(4),
                      PAGESIZE2(4),
                      CLUSTER (CUSTOMER,PRODNO))
       WHERE  (CUSTOMER)  BETWEEN (?) AND (?)    -- Data is split and placed according to
                                                   -- CUSTOMER numbers.

CREATE DSI  JAPAN_ORDER_DSI                        -- ORDER table DSI for companies located in JAPAN
       DSO  ORDER_DSO
       USING (71,72)                               -- Number 71 and 72 indicate companies located in
       ALLOCATE PRIME     ON DBSP_2 SIZE 200K,     -- JAPAN.
                OVERFLOW  ON DBSP_2 SIZE  80k
```

Create a television and refrigerator DSI by splitting and storing STOCK table data according to PRODUCT and WHCODE values. Set PRODUCT and WHCODE for the split condition, and specify multiple split values.

Example 4:

Create a DSI for products TELEVISION and REFRIGERATOR for the STOCK table.

```
CREATE DSO  STOCK_DSO                                 -- STOCK table DSO
       FROM STOCKS.STOCK
       TYPE SEQUENTIAL ( PAGESIZE(4),
                         ORDER(0))
       WHERE (PRODUCT,WHCODE) = (?,?)                 -- Split data according to PRODUCT and
                                                      -- WHCODE

CREATE DSI  STOCK_DSI_TV_FRIDG                        -- STOCK table DSI for warehouse codes 2 and 1
       DSO  STOCK_DSO
       USING  ('TELEVISION',2), ('REFRIGERATOR',1)    -- STOCK table for products
       ALLOCATE DATA ON DBSP_1 SIZE 280K              -- TELEVISION and REFRIGERATOR
```

Create a sales amount DSI by splitting and storing SALES table data according to FISCAL YEAR and MONTH. Specify multiple split values.

Example 5:

Create a DSI for sales amount of the second half of fiscal years 1999 and 2000 for the sales table.

```
CREATE DSO  SALES_DSO                                          -- SALES table DSO
       FROM STOCKS.SALES
       TYPE SEQUENTIAL(PAGESIZE(4),ORDER(0))
       WHERE (FISCAL YEAR, MONTH) BETWEEN (?,?) AND (?,?)      -- Split data according to
                                                               -- FISCAL YEAR and
                                                               -- MONTH

CREATE DSI  SECOND_HALF_OF_FISCAL_YEAR_DSI                     -- DSI for second half of fiscal
       DSO  SALES_DSO                                          -- 2000 and 2001
       USING   (2000, 10, 2001, 3),(2001, 10, 2002, 3)         -- SALES table for FISCAL
       ALLOCATE DATA ON DBSP_2   SIZE 280K                     -- YEAR and MONTH
```

SALES table

Fiscal year	Month	Sales amount
2000	4	85
2000	5	70
2000	6	40
2000	7	60
2000	8	55
2000	9	80
2000	10	100
2000	11	85
2000	12	50
2001	1	75
2001	2	60
2001	3	110
2001	4	60
⋮	⋮	⋮
2001	10	120
2001	11	85
2001	12	70
2002	1	80
2002	2	65
2002	3	80

DSI for the second half (Oct. to Mar.) of fiscal year 2000

DSI for the second half (Oct. to Mar.) of fiscal year 2001

DSI for the second half of fiscal year

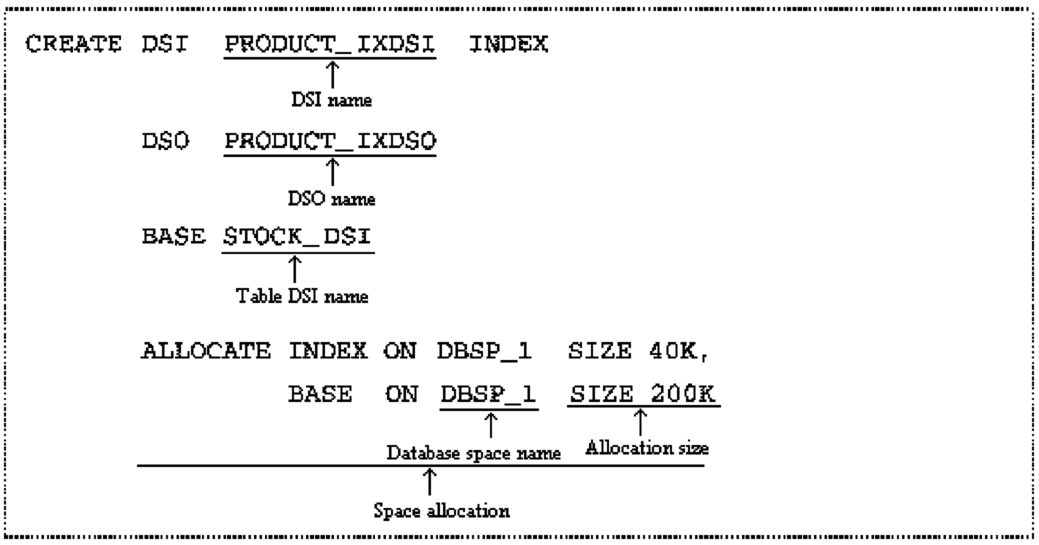
2.7.5 Index DSI definition

Use the CREATE DSI statement to specify an index DSI definition. The index DSI definition allocates database space according to the index DSO definition.

A new index DSI definition can be added for a table DSI in which data has already been stored.

Example:

Create an index DSI definition.



```
CREATE DSI  PRODUCT_IXDSI  INDEX
        DSO  PRODUCT_IXDSO
        BASE STOCK_DSI
        ALLOCATE INDEX ON DBSP_1  SIZE 40K,
                 BASE  ON DBSP_1  SIZE 200K
```

■DSI name

For the database name, specify up to 36 alphanumeric characters beginning with an alphabetic character. The DSI name must be unique within the database.

■DSO name

Specify the index DSO name given in a CREATE DSO statement.

■Table DSI name

Specify a DSI name indicating the corresponding base table. When split table operation is to be applied, create an index for each table DSI. When split table operation is not to be applied, can not specify the table DSI name.

■Space allocation

Specify the physical space to be allocated to the index DSI.

Specify the name of the database space where the data is to be physically stored. In addition, specify the size of the storage area to be acquired for this DSI within the database space.

■Database space name

Specify the name of the physical database space where the index data is actually stored.

To store data in multiple database spaces for scalable log operation, specify their names in the same log group.

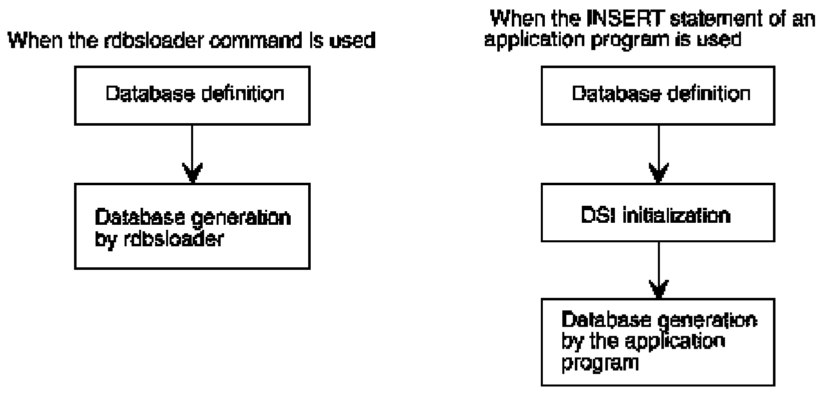
■Allocation size

Specify an unsigned integer combined with a unit symbol (K or M) to denote the size of the storage area acquired in the database space. The unit symbol K indicates kilobytes, and the unit symbol M indicates megabytes.

2.7.6 DSI initialization

A DSI must be initialized before data is stored. This section explains DSI initialization. Figure: Overview of DSI initialization provides an overview of DSI initialization.

[Figure: Overview of DSI initialization]


When the rdbsloader command is used
Database definition
Database generation by rdbsloader
When the INSERT statement of an application program is used
Database definition
DSI initialization
Database generation by the application program

A DSI must be initialized before data is stored. The rdbfmt command performs DSI initialization. However, when the rdbsloader command is used to store data, DSI initialization is unnecessary.

In addition, if the DSI is associated with a shared buffer to improve performance, the rdbconbf command must be executed before the rdbfmt command. For more information on the rdbconbf and rdbfmt commands, refer to the man command (under UNIX) or the SymfoWARE/RDB online manual (under Windows NT).

An example of DSI initialization follows.

Example:

Initialize created DSIs.

```
rdbconbf -i STOCKMN_DB.STOCK_DSI POOL1
rdbconbf -i STOCKMN_DB.JAPAN_ORDER_DSI POOL2
rdbconbf -i STOCKMN_DB.ITMNO_IXDSI POOL3
rdbfmt -mi -i STOCKMN_DB.STOCK_DSI
rdbfmt -mi -i STOCKMN_DB.JAPAN_ORDER_DSI
rdbfmt -mi -i STOCKMN_DB.ITMNO_IXDSI
```

2.7.7 Scope definition

A scope is defined by using the CREATE SCOPE statement.

The CREATE SCOPE must be used by the person who defined table DSI.

Example:

Define scope TOKYO.

$$\text{CREATE SCOPE } \underbrace{\text{TOKYO\_SCOPE}}_{\text{Scope name}}$$

$$\text{DSI } \underbrace{\text{(TOKYO\_STOCK\_DSI, TOKYO\_ORDER\_DSI)}}_{\text{DSI name list}}$$

■Scope name

Specify the name of a scope that limits the range of DSI names using up to 36 alphanumeric characters beginning with an alphabetic character.

A scope name is unique in a database.

■DSI name list

Specify the DSI names of tables whose access range is limited as DSI name lists. Each DSI name must be unique in the database. A DSI that is the same as that of the scope definition statement cannot be specified.

■Caution:

Limiting a data manipulation range with the scope function is only effective for SQL statements of an application program or rdbupt command. It is not effective for RDB commands such as rdbsloader.

2.8 Applying a Storage Structure Definition

This section explains how to apply a storage structure definition. Application of storage structure definition involves application of scope definition.

2.8.1 Scope definition application

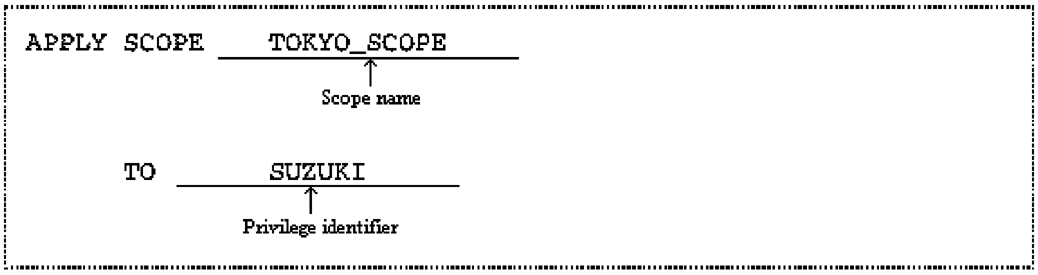
A scope definition is applied by using the APPLY SCOPE statement. A scope definition can be applied to a user to limit the range of items in a table that the user is allowed to manipulate.

The executor of the APPLY SCOPE statement must be the user who defined the scope.

A sample scope application follows.

Example:

Apply scope TOKYO to user SUZUKI.



```
APPLY SCOPE    TOKYO_SCOPE

        TO       SUZUKI
```

■Scope name

Specify the name of the scope applied to the table user with up to 36 alphanumeric characters beginning with an alphabetic character.

A scope name must be defined by the CREATE SCOPE statement.

A scope name is unique in a database.

■Privilege identifier

Specify the user identifier of the scope user.

2.9 Simplifying a Storage Structure Definition

This section explains how to simplify a storage structure definition.

When defining a storage structure in a simplified form, use table or index definition. In a table or index definition, specify a database space to store data. SymfoWARE/RDB automatically defines a storage structure according to the specification. In this case, the table storage structure is set to SEQUENTIAL and the index storage structure is set to BTREE. For a multimedia data storage table, SEQUENTIAL or OBJECT can be selected as the table storage structure; however, split storage is not available.

If a storage structure is defined in a simplified form, DSO and DSI of the table are automatically named in accordance with the table or index definition. The data length and allocation size are then also automatically determined.

A simplified storage structure definition dynamically extends the DSI capacity.

The prefix for naming, the data length, the allocation size, and the DSI capacity extension can be changed by using parameters in the operating environment file. For a multimedia data storage table, the storage structure can be selected. For information about the operating environment file, refer to the "RDB User's Guide: Application Program Development."

2.9.1 Table definition

In a table definition, specify a data space as a storage area for table data. When a storage area is specified, the table DSO and DSI are automatically defined. In addition, DSI is automatically initialized.

■Storage area specification

Specify a database space as a storage area for actual table data. If a storage area is specified, the table and schema names should be specified using up to 8 alphanumeric characters beginning with an alphabetic character.

If DEFAULT_DSI_NAME=CODE is specified in the system operating environment file, however, the table and schema names may be specified using up to 36 alphanumeric characters beginning with an alphabetic character.

The example below defines a base table for an stock management database with a storage structure. Define an STOCK table belonging to schema STOCKS.

Example:

CREATE TABLE statement defining an STOCK table (with storage area specification)

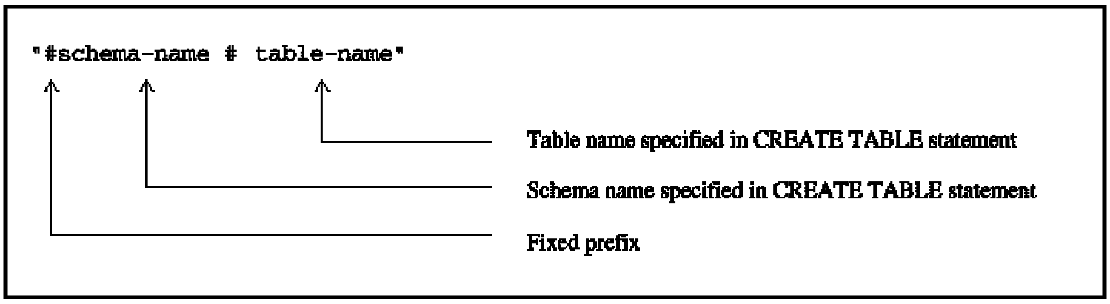
```
CREATE TABLE STOCKS.STOCK  ( ITMNO     SMALLINT  NOT NULL,
                             PRODUCT   CHARACTER(25) NOT NULL,
                             STOCKQTY  INTEGER,
                             WHCODE    SMALLINT,
                             PRIMARY KEY(ITMNO))

                             COMMENT 'Stock item and quantity, warehouse
                                                                 table'
                             ON DBSPACE1
                                   ↑
                             Storage area specification
```

The above specification has the following definitions:

```
CREATE TABLE STOCKS.STOCK  ( ITMNO     SMALLINT  NOT NULL,
                             PRODUCT   CHARACTER(25) NOT NULL,
                             STOCKQTY  INTEGER,
                             WHCODE    SMALLINT,
                             PRIMARY KEY(ITMNO))

                             COMMENT 'Stock item and quantity, warehouse
                                                                 table'
CREATE DSO #STOCKS#STOCK  FROM STOCKS.STOCK

       TYPE SEQUENTIAL(PAGESIZE(4), ORDER(1))

CREATE DSI #STOCKS#STOCK  DSO  #STOCKS#STOCK

       ALLOCATE DATA ON DBSPACE1 SIZE 256K
```

◆Table DSO and DSI names

DSO and DSI of a table are named by combining schema and table names in a table definition. The DSO and DSI names are the same.


"#schema-name # table-name"
Table name specified in CREATE TABLE statement
Schema name specified in CREATE TABLE statement
Fixed prefix

If DEFAULT_DSI_NAME=CODE is specified in the system operating environment file, however, table DSO and DSI are named with a 10-diit figure determined by the system.

Examples are given below.

Example 1:

When the schema and table names are character strings

```
CREATE TABLE STOCKS.ORDERS( ...... ) ON DBSPACE1

⇒ DSO name : #STOCKS#ORDERS
  DSI name : #STOCKS#ORDERS
```

Example 2:

If DEFAULT_DSI_NAME=CODE is specified in the system operating environment file

```
CREATE TABLE STOCKSMASTER.ORDERSMASTER( ...... ) ON DBSPACE1

⇒ DSO name : #0000000155
  DSI name : #0000000155
```

◆Table storage structure

The table storage structure becomes as follows: Note that these are automatically defined by SymfoWARE/RDB.

Storage structure:
SEQUENTIAL structure
Data part page size:
4 kilobytes
Data part allocation size:
256 kilobytes
Area reuse specification:
ORDER(1)

For information about the storage structure, see "2.7 Storage structure definition."

2.9.2 Table definition for multimedia data storage

For a multimedia data storage table, specification of the storage area, and naming a table DSO and DSI are the same as that for a table of characters and numeric values. The following example is a storage area specification for a table that is to storage multimedia data and its table storage structure. For information about specifying the table DSO and table DSI names, see 2.6.2 "Table definition."

Example:

Storage area specification for a table that is to storage multimedia data

```
CREATE TABLE  S1.PRODPHOT (ITMNO     SMALLINT PRIMARY KEY NOT NULL,
                           PRODPHOTO BLOB(1M) NOT NULL)
                            ON DBSPACE1
                                 ↑
                       Storage area specification
```

If DEFAULT_DSI_TYPE is specified for the operating environment file, this specification has the same meaning as of the definition below.

· At DEFAULT_DSI_TYPE = SEQUENTIAL

```
CREATE TABLE S1. PRODPHOT(ITMNO     SMALLINT  PRIMARY KEY  NOT NULL.
                          PRODPHOTO BLOB(1M)  NOT NULL)

CREATE DSD #S1# PRODPHOT  FROM S1.PRODPHOT
       TYPE SEQUENTIAL(PAGESIZE(32).ORDER(1))

CREATE DSI #S1# PRODPHOT  DSD #S1# PRODPHOT
       ALLOCATE DATA ON DBSPACE1 SIZE 700000K
```

Note:
In this example, 700,000 kilobytes are allocated to the table data
storage area in DEFAULT_TABLE_SIZE of the operating environment file.

· At DEFAULT_DSI_TYPE = OBJECT

```
CREATE TABLE  S1.PRODPHOT (ITMNO     SMALLINT PRIMARY KEY NOT NULL,
                           PRODPHOTO BLOB(1M) NOT NULL)

CREATE DSO   #S1#PRODPHOT  FROM S1.PRODPHOT
             TYPE OBJECT(PAGESIZE(32))

CREATE DSI   #S1#PRODPHOT DSO #S1#PRODPHOT
             ALLOCATE DATA ON DBSPACE1 SIZE 700000K
```

Note:
In this example, 700,000 kilobytes are allocated to the table data
storage area in DEFAULT_OBJECT_ TABLE_SIZE of the operating environment file.

The table storage structure is defined as follows.

Storage structure:

If the table format does not satisfy the conditions listed below, specify SEQUENTIAL as the storage structure. If the table format satisfies all the conditions listed below, specify OBJECT as the storage structure. If necessary, the OBJECT structure can be changed to SEQUENTIAL structure by specifying DEFAULT_DSI_TYPE for the operating environment file. If this parameter is omitted, OBJECT is assumed to have been specified.

· Only one BLOB-type column is specified at the end of a table with a size of 32 kilobytes or more.
· Columns other than BLOB-type columns have the fixed-length attribute.
· The NOT NULL restriction is specified for the BLOB-type column.

Page size of data part:

Specify the page size defined in DEFAULT_TABLE_SIZE of the operating environment file in the SEQUENTIAL structure. The page size should be 32 kilobytes.
In the OBJECT structure, specify a page size of 32 kilobytes.

Allocated size of data part:

For the SEQUENTIAL structure, specify the allocated size of the table data storage area defined in DEFAULT_TABLE_SIZE of the operating environment file. Reserve a sufficient value, taking into account the amount of data to be handled.
For the OBJECT structure, specify the allocated size of the table data storage area defined in

DEFAULT_OBJECT_TABLE_SIZE of the operating environment file. Reserve a sufficient value, taking into account the amount of data to be handled.

For information about the storage structure, see 2.7 "Defining a Storage Structure."

2.9.3 Index definition

Use the CREATE INDEX statement to define an index.

In the index definition, specify the columns that form the index key and the database space for holding the table data. SymfoWARE/RDB automatically generates the index DSO and DSI definitions.

■Index name

Define the name (index name) to be assigned to the index, using up to 8 alphanumeric characters beginning with an alphabetic character.

Note that the index name can be specified with character strings of up to 36 alphanumeric characters beginning with an alphabetic character when DEFAULT_DSI_NAME=CODE is specified in the system operating environment file.

To delete the index definition, specify the index definition to be deleted by index name.

■Key specification

Specify the column name of the column that is to be the index key. Multiple columns can be combined to form a single index.

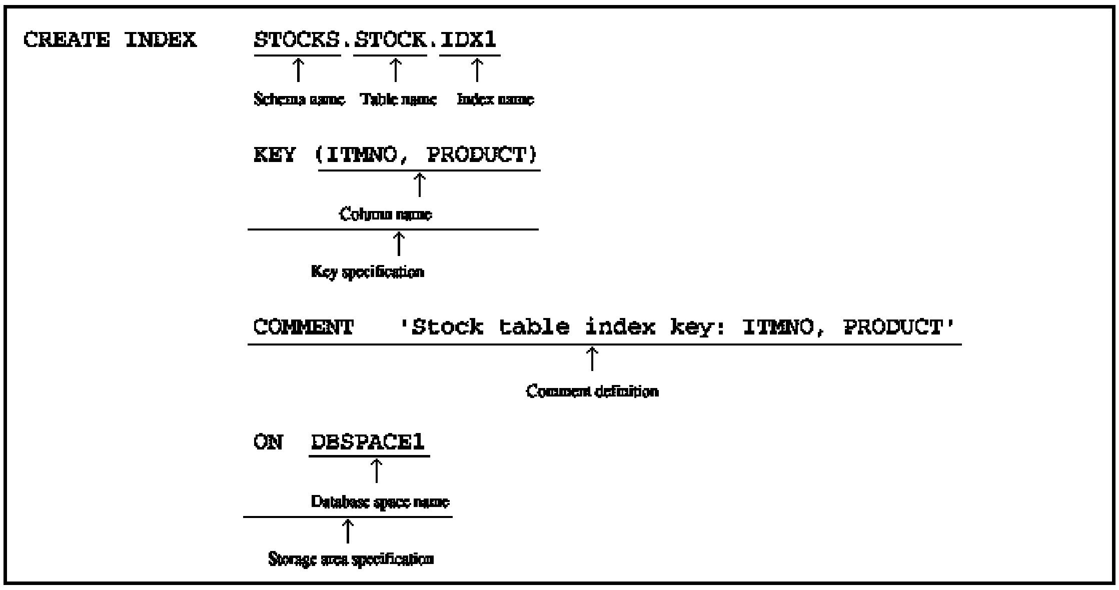
■Storage area specification

After the keyword ON, specify the database space name of the database space for storing the index.

The following example defines the storage structure of an index formed by combining the ITMNO and PRODUCT columns of the stock management database. The index name is IDX1, and the index is stored in the database space named DBSPACE1.

Example:

CREATE INDEX statements for defining the STOCK table



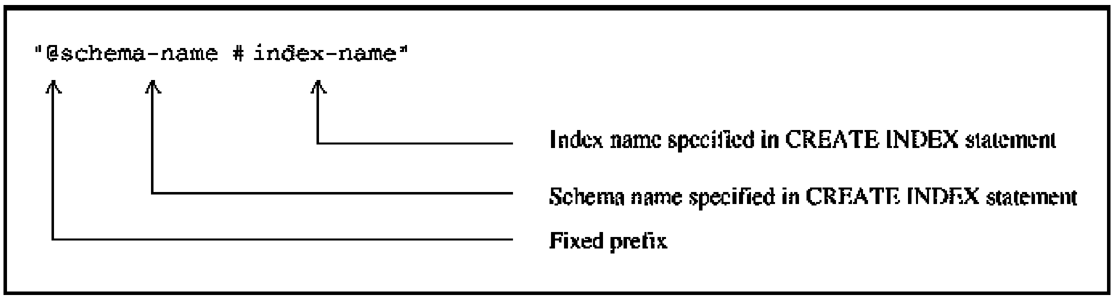
```
CREATE INDEX    STOCKS.STOCK.IDX1
                KEY (ITMNO, PRODUCT)
                COMMENT   'Stock table index key: ITMNO, PRODUCT'
                ON  DBSPACE1
```

The preceding specification has the same meaning as the following definition.

```
CREATE DSO @STOCKS#IDX1

       INDEX ON  STOCKS.STOCK (ITMNO, PRODUCT)

       TYPE BTREE(RAGESIZE1(2), PAGESIZE2(2))  BY ADDRESS

CREATE DSI @STOCKS#IDX1 INDEX  DSO @STOCKS#IDX1

       OPTION (DEGENERATE)

       ALLOCATE BASE ON DBSPACE1 SIZE 168K,

        INDEX ON DBSPACE1 SIZE 32K
```

■Index DSO name and DSI name

The index DSO and index DSI names are generated by combining the schema name and index name from the index definition. The DSO and DSI names are the same.


"@schema-name # index-name"
Index name specified in CREATE INDEX statement
Schema name specified in CREATE INDEX statement
Fixed prefix

If DEFAULT_DSI_NAME=CODE is specified in the system operating environment file, the system assigns 10-digit names for the index DSO and DSI names.

An example follows.

Example 1:

When the schema name and index name are character strings

```
CREATE INDEX STOCKS.STOCK.IDX1 KEY ( ..... ) ON DBSPACE1

⇒ DSO name : @STOCKS#IDX1
   DSI name : @STOCKS#IDX1
```

Example 2:

When DEFAULT_DSI_NAME=CODE is specified in the system operating environment file

```
CREATE INDEX STOCKS.STOCK.IND1MASTER KEY ( ...... ) ON DBSPACE1

⇒ DSO name : @0000000017
   DSI name : @0000000017
```

■Index storage structure

The index storage structure is as follows:

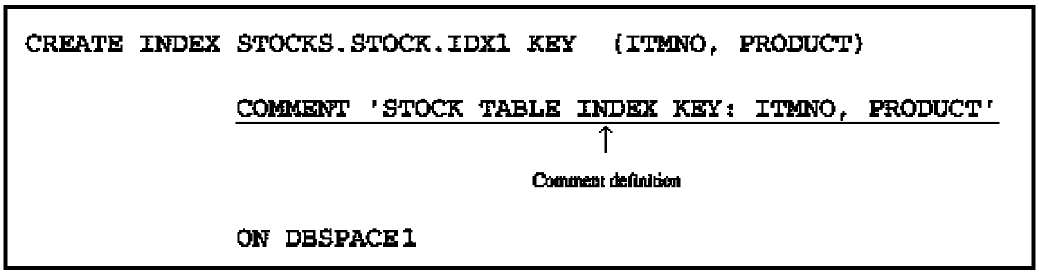
Storage structure:
    BTREE structure
Data part page size:
    2 kilobytes
Index part page size:
    2 kilobytes
Data part allocation size:
    168 kilobytes
Degeneration specification:
    Present
Index part allocation size:
    32 kilobytes

■Index comment definition

A comment consisting of a character string can be specified for an index if the index is defined with an abbreviated storage structure definition. A character string of up to 256 bytes can be specified. If no comment is necessary, omit the specification. An example follows.

Example:

Specifying a comment for an index that has an abbreviated storage structure definition.



```
CREATE INDEX STOCKS.STOCK.IDX1 KEY  (ITMNO, PRODUCT)

             COMMENT 'STOCK TABLE INDEX KEY: ITMNO, PRODUCT'

             ON DBSPACE1
```

Comment definition

2.10 Defining a Temporary Table

Before using a temporary table, define the temporary table and an index.

A temporary table is created specifically for a user of an application program. Multiple users can use temporary tables with the same table name. Uses of temporary tables are listed below.

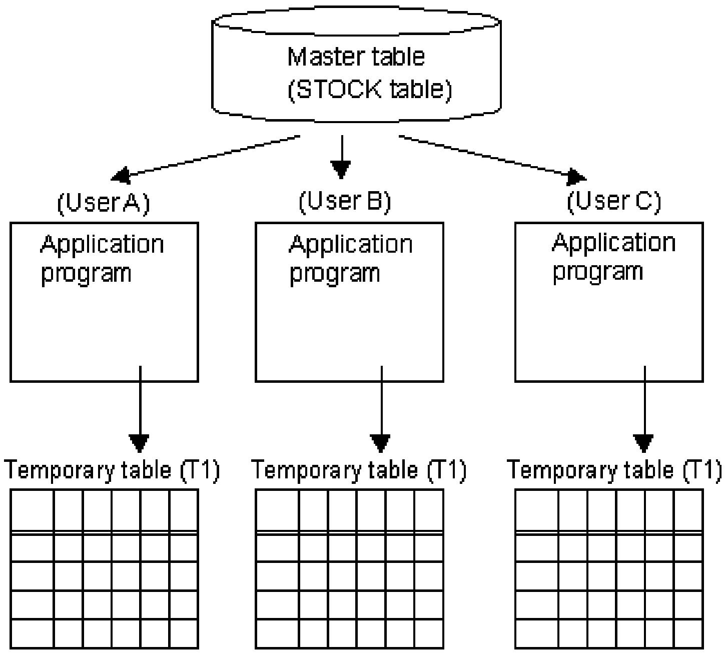
· Extracting data necessary for a user from a database and creating a table containing the data
· Incorporating processing results of a procedure routine into an application program using a temporary table

As described above, a temporary table can be used as a table independent of other application programs to facilitate development of an application program.

Extracting data necessary for a user from a database and creating a table containing the dat a

When data manipulated by a user can be limited or if specific data is frequently referenced, creating in advance a compact table containing necessary data enables smooth processing that is not affected by update operations of another user.

Temporary tables can be used for such processing. To create a compact table, data can be extracted from the master table (STOCK table) according to the conditions set for a user and then stored in a temporary table (T1).

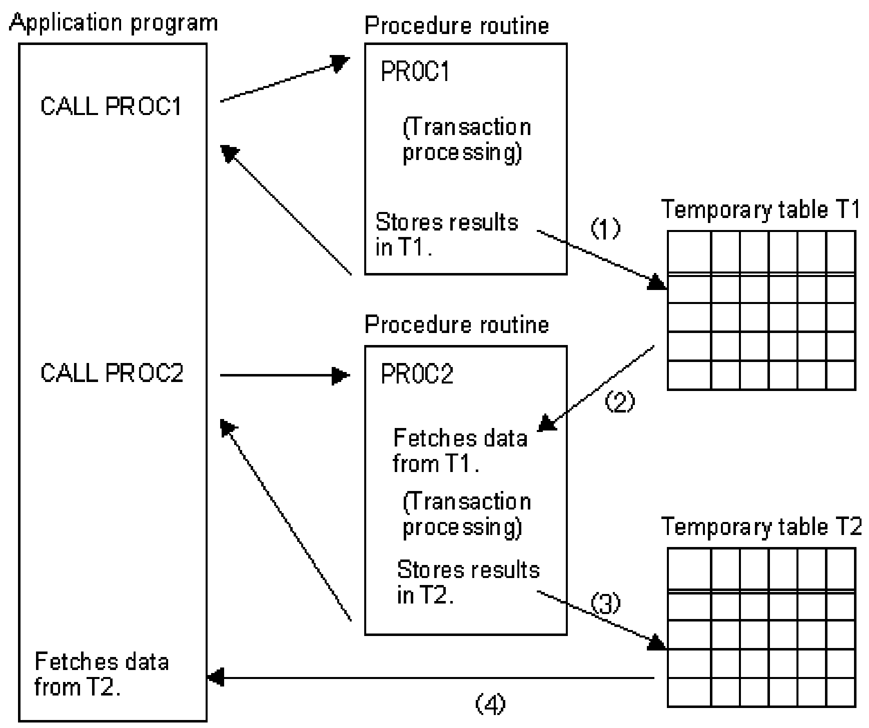


A user can execute the following INSERT statement to create a temporary table (T1) that contains necessary data:

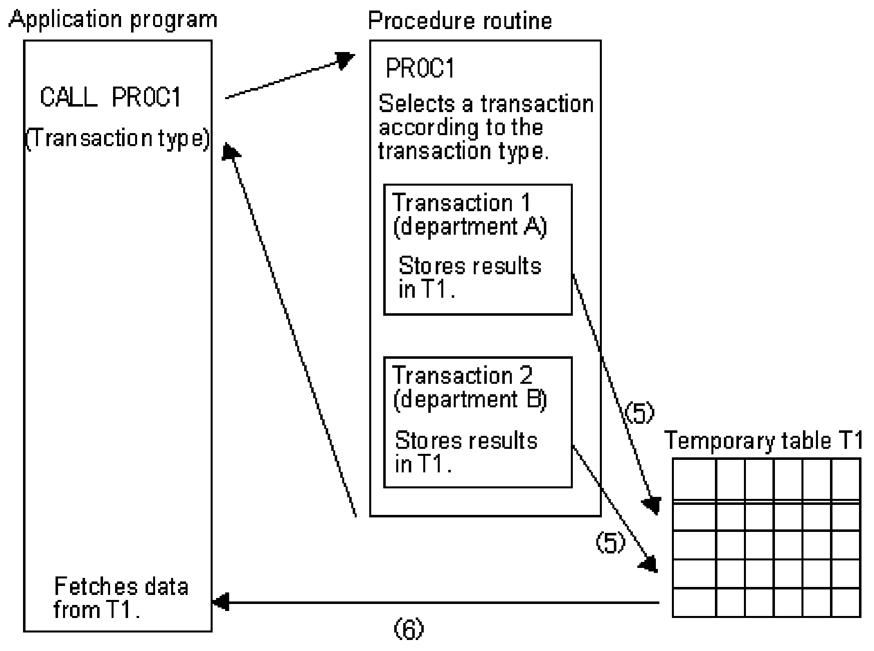
```
INSERT INTO T1
  SELECT C1, C2, C3 FROM STOCK TABLE
  WHERE conditional-expression-that-extracts-data-necessary-for-a-user
```

Incorporating processing results of a procedure routine into an application program using a temporary table

Because a temporary table is created for one user, data can be passed between procedure routines that are executed consecutively ((1) and (2)) and between an application program and procedure routine ((3) and (4)).

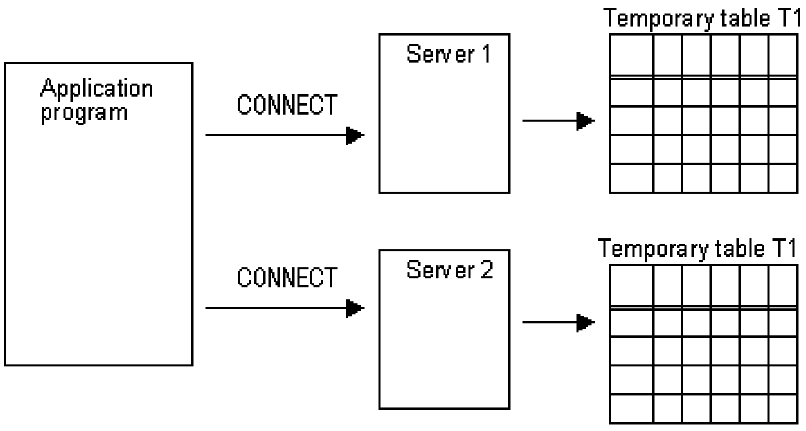


Processing results of a procedure routine for multiple transactions that outputs processing results in the same format can be stored in a temporary table (5) in order to pass the results to an application program (6).



For more specific examples of use, refer to the "RDB User's Guide: Application Program Development."

When multiple connections are established between an application program and servers, a temporary table is created for each connection. These temporary tables are independent, and the connections cannot share the temporary tables.



The user can specify whether to use a temporary table within a session of the application program or within a transaction. Stored data is erased after the session or transaction terminates.

The storage structure of a temporary table is SEQUENTIAL. The storage structure of the index of a temporary table is BTREE. The storage structures are automatically defined.

■Defining a temporary table

Use the CREATE TABLE statement to define a temporary table.

A sample temporary table definition follows.

This sample definition defines the STOCK_TEMPORARY table for database spaces DBSP_1 and DBSP_2.

Example:

CREATE TABLE statement for defining the STOCK_TEMPORARY table

```
CREATE GLOBAL TEMPORARY TABLE STOCKS.STOCK_TEMPORARY
                                   ↑          ↑
                              Schema name  Temporary table name

     (ITMNO SMALLINT NOT NULL, STOCKQTY INTEGER, ...
               ↑                         ↑
          Table element             Table element

     PRIMARY KEY (ITMNO))
               ↑
          Table element

     ON COMMIT DELETE ROWS
                 ↑
         Row deletion specification

     COMMENT 'STOCK_CHECK_TEMPORARY_TABLE'
                          ↑
                   Comment definition

     ON  DBSP_1    2,    DBSP_2    3
           ↑       ↑
   Database space name  Number of users
```

◆Table name

Specify a name to be assigned to a temporary table. Specify up to 36 alphanumeric characters for a table name, whose first character must be an alphabetic character. The table name must be qualified by a schema name.

◆Table elements

Specify the name, data type, default value, and constraint of each column in a temporary table. Specify up to 36 alphanumeric characters for a column name, whose first character must be an alphabetic character. Each column name within the table must be unique.

◆Row deletion specification

Specify when to delete a row in a temporary table. If this argument is omitted, the temporary table is assumed to be used within a transaction, and data stored in the temporary table is deleted when the transaction terminates.

DELETE ROWS:
: The temporary table is assumed to be used within a transaction, and data stored in the temporary table is deleted when the transaction terminates.

PRESERVE ROWS:
: The temporary table is assumed to be used in a session, and data stored in the temporary table is deleted when the session terminates.

◆Comment definition

For a temporary table, a comment can be specified with a character string or national character string. A character string of up to 256 bytes can be specified. Both a character string and national character string can be specified as a comment. If no comment is necessary, omit the specification.

◆Database space name

Specify the name of a physical database space in which to store data.

◆Number of users

Specify the number of temporary tables to create in a database space. Specify the number of users so that the total number of database space users indicates the multiplicity of the application program that uses the temporary tables.

◆DSO name and DSI name of temporary table

The DSO name and DSI name of a temporary table begin with _TEMP.

■Defining an index

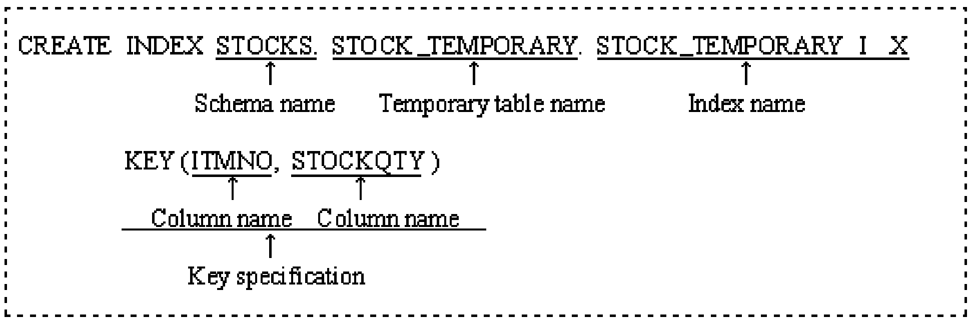
Use the CREATE INDEX statement to define an index. In the index definition, specify the columns that form the index key.

A sample index definition for a temporary table follows.

This sample definition defines an index for the STOCK_TEMPORARY table.

Example:

CREATE INDEX statement for defining an index



```
CREATE INDEX STOCKS. STOCK_TEMPORARY. STOCK_TEMPORARY I X
          KEY (ITMNO, STOCKQTY )
```

Remarks: KEY (ITMNO, STOCKQTY) is a key specification.

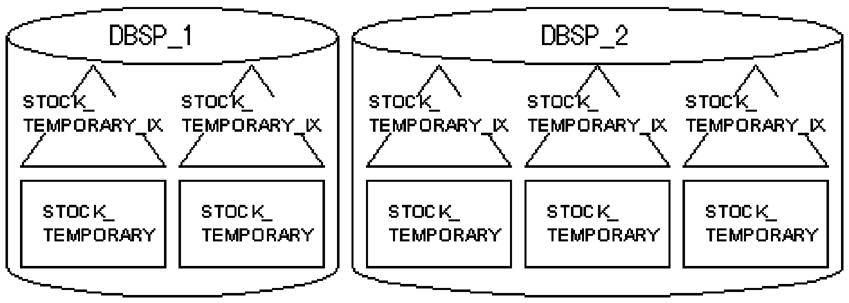
◆Index name

Define a name (index name) to be assigned to an index. Specify up to 36 alphanumeric characters for the index name, whose first character must be an alphabetic character.

◆Key specification

Specify the name of a column to be used as the index key. Multiple columns can be combined to form a single index.

The index is created in the database space specified in the temporary table definition and paired with a temporary table.



◆Index DSO name and DSI name

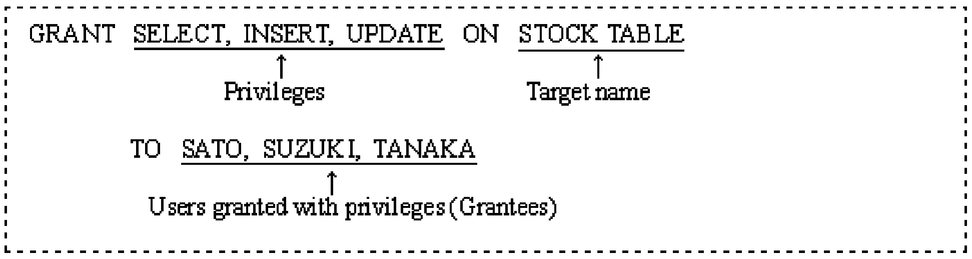
The DSO name and DSI name of an index begin with TEMP.

2.11 Privilege Information Definition

Use the GRANT statement to define privilege information. If a resource is defined, only the user who defined the resource retains all privileges for the resource. In order for another user to access the resource, privilege information must be defined. The GRANT statement can be used to specify which privileges to grant to specific users for a resource. A sample definition that specifies privilege information in the stock management database follows. The definition defines privilege information for the STOCK table belonging to schema STOCKS.

Example:

Grants the reference, storage, and update privileges for the STOCK table.



```
GRANT SELECT, INSERT, UPDATE ON STOCK TABLE

      TO SATO, SUZUKI, TANAKA
```

When defining privilege information, the user may want to specify at one time the privileges necessary for a specific transaction. In this situation, use of roles is convenient for granting privileges.

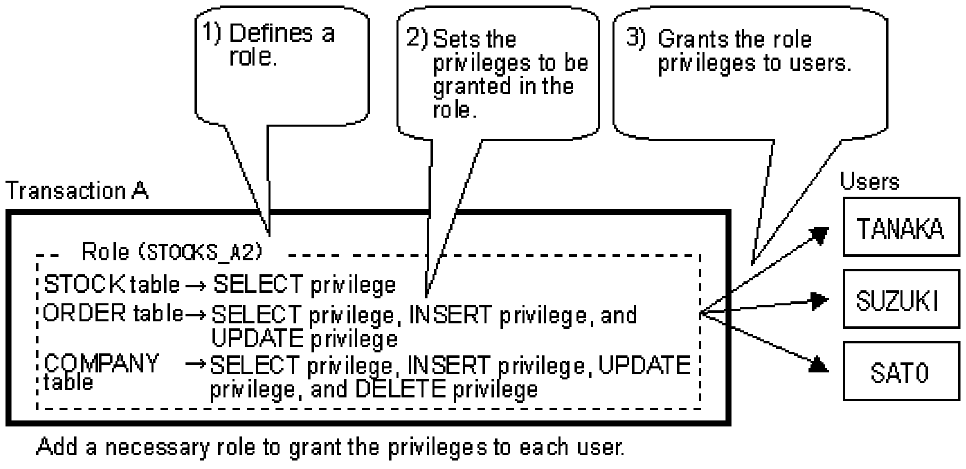
■Defining privileges using a role

A role is a group of privileges necessary for a transaction. Define a role to specify the privileges necessary for a transaction. For efficient privilege management, a role can be defined to grant the role privileges to all users who perform the transaction.

The procedure for granting privileges using the role function is given below.

1. Define a role.
2. Specify the privileges to be granted in the role.
3. Grant the role privileges to users.

The role function is outlined below.


1) Defines a role.
2) Sets the privileges to be granted in the role.
3) Grants the role privileges to users.
Transaction A
Role (STOCKS_A2)
STOCK table → SELECT privilege
ORDER table → SELECT privilege, INSERT privilege, and UPDATE privilege
COMPANY table → SELECT privilege, INSERT privilege, UPDATE privilege, and DELETE privilege
Users
TANAKA
SUZUKI
SATO
Add a necessary role to grant the privileges to each user.

◆1) Define a role.

Use the CREATE ROLE statement to define a role.

A sample role definition for defining role STOCKS_A2 follows.

Example:

```
CREATE  ROLE  STOCKS_A2
                  ↑
              Role name
```

◆2) Specify the privileges to be granted in the role.

Use the GRANT statement to specify the privileges to be granted in the role. In the defined role, specify the privileges granted for accessing a table in a database.

The following table lists the privileges that can be defined in a role by using the GRANT statement.

Privilege	Operation
SELECT privilege	Privilege to reference data in a table in a database
UPDATE privilege	Privilege to update data in a table in a database
DELETE privilege	Privilege to delete data from a table in a database
INSERT privilege	Privilege to insert data into a table in a database
EXECUTE privilege	Privilege to execute a procedure or function routine in a database
TRIGGER privilege	Privilege to define a trigger for a table
CREATE privilege	Privilege to define a table, view table, procedure routine, function routine, or sequence for a schema
ALLOCATE privilege	Privilege to allocate a table area in a database space using a DSI definition
DROP privilege	Privilege to delete a schema, table, view table, procedure routine, function routine, trigger, or sequence
ALTER privilege	Privilege to update a table definition
INDEX privilege	Privilege to define an index for a table

An example of specifying privileges granted for individual tables in role STOCKS_A2 follows.

Example:

```
GRANT  SELECT  ON  STOCKS. STOCK TABLE  TO ROLE  STOCKS_A2
         ↑                 ↑                              ↑
      Privilege        Target name                    Role name

GRANT  SELECT, INSERT, UPDATE  ON  STOCKS. ORDER TO ROLE STOCKS_A2
GRANT  SELECT, UPDATE, INSERT, DELETE  ON  STOCKS. COMPANY  TO  ROLE  STOCKS_A2
```

◆3) Grant the role privileges to users.

Grant the role privileges to users.

Use the GRANT statement to grant the role privileges to users.

An example of granting the privileges granted in role STOCKS_A2 to users SATO, SUZUKI, and TANAKA follows.

Example:

```
GRANT  STOCKS_A2>  TO  SATO, SUZUKI, TANAKA
           ↑                   ↑
       Role name   Users granted with privileges (Grantees)
```

To enable the privileges specified with the GRANT statement in the defined role, execute the SET ROLE statement in an application program. For details on how to execute the SET ROLE statement in an application program, refer to the "RDB User's Guide: Application Program Development."

◆Specifying a default role

After a role is created for defining privilege information, a default role can be specified. A default role is a role that is effective prior to execution of the SET ROLE statement in the application program specified at environment configuration.

Use the ALTER USER statement to specify a default role.

An example of specifying default role STOCKS_A2 for users SATO, SUZUKI, and TANAKA follows.

Example:

```
ALTER  USER  SATO DEFAULT_ROLE=STOCKS_A2
ALTER  USER  SUZUKI DEFAULT_ROLE=STOCKS_A2
ALTER  USER  TANAKA  DEFAULT_ROLE=STOCKS_A2
```

2.12 Defining Optimization Information

For more efficient database access, define optimization information. Determine the optimization information by considering the number of records stored in the database and the changes in the index key caused after the definition of the storage structure is completed.

After defining the optimization information, fetch and analyze an access plan so the database can be tuned appropriately. Once the database is tuned by an access plan, optimization information need not be redefined and updated even if the data status is changed. For details on how to fetch and analyze the access plan, refer to the SQLTOOL User's Guide.

If it is impossible to estimate the number of records to be stored in a database and variations of the index key, optimization information can be defined after the data is actually stored.

■What is optimization information?

Optimization information is used to efficiently process data according to data status. It is defined for base table and index DSIs, and is used to execute application programs. Optimization information contains values that depend on the status of the data in a database, for example, number of data items, and base table and index DSI space requirements.

■Optimization information definition opportunity

Optimization information must be determined in an ordinary way, considering the number of records stored in a database and variations of the index key that will occur after the definition of the storage structure is completed. Additional opportunities for optimization information definition are listed below.

◆When optimization information is defined with values assumed in advance:

- When a database is defined
- When a database is reconfigured
    - When a table DSI is added at split storage
    - When index is added
    - When a large amount of data is added or updated

■Defining optimization information

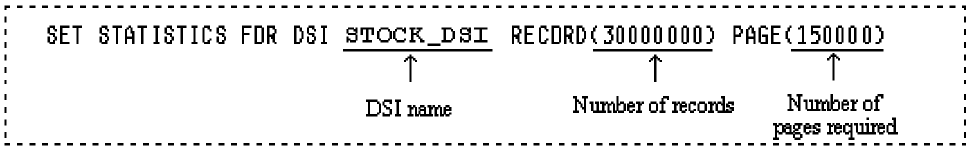
Use the rdbddlex command to specify the SET STATISTICS statement to define optimization information with the values assumed in advance. For more information on the SET STATISTICS statement, refer to the SQL Reference Guide.

Optimization information must be defined for each table or index DSI.

Optimization information can be defined for each table or index DSO; however, the system converts the specified value for each DSO into a value for each DSI.

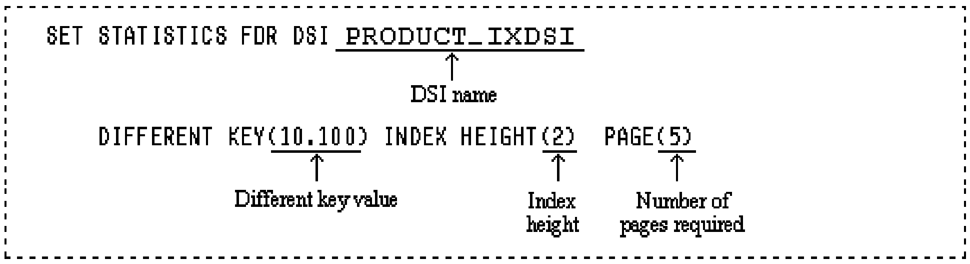
Optimization information should therefore be defined for each DSI.

- Definition for each table DSI (data structure: SEQUENTIAL)



```
SET STATISTICS FOR DSI STOCK_DSI RECORD(30000000) PAGE(150000)
```

- Definition for each index DSI (data structure: BTREE)



```
SET STATISTICS FOR DSI PRODUCT_IXDSI

    DIFFERENT KEY(10.100) INDEX HEIGHT(2)  PAGE(5)
```

The items to be specified depend on the definition method. Details of the optimization information to be defined are listed below.

◆Definition for each DSI

Table DSI (SEQUENTIAL structure)

Set item	Specified contents	Default
Number of records	Specify the number of records to be stored.	Cannot be omitted.
Number of pages required (DATA part)	Specify the value obtained at estimation of the DSI size.	Maximum record size automatically calculated by the system.

Table DSI (RANDOM structure)

Set item	Specified contents	Default
Number of records	Specify the number of records to be stored.	Cannot be omitted.
Number of pages required (DATA part)	Specify the value obtained at estimation of the DSI size.	Maximum record size automatically calculated by the system.
Number of pages required (OVERFLOW part)	(*1)	Set to 0.
Maximum number of pages (OVERFLOW part)	(*1)	Set to 0.
Average number of pages (OVERFLOW part)	(*1)	Set to 0.

Table DSI (OBJECT structure)

Set item	Specified contents	Default
Number of records	Specify the number of records to be stored.	Cannot be omitted.
Number of pages required (DATA part)	Specify the value obtained at estimation of the DSI size.	Maximum record size automatically calculated by the system.

Index DSI (BTREE structure)

Set item	Specified contents	Default
Number of pages required (BASE part)	Specify the value obtained at estimation of the DSI size.	Maximum record size automatically calculated by the system.
Index height	(*1)	Set by the system according to the number of records in the related table
Number of different key values	Specify the number of different key values for each combination of columns making up the index. (*2)	Cannot be omitted.

*1 Specified when the set value is known in advance, e.g., the created database has the same configuration as that of a database defined with optimization information after data was stored.
*2 Specification example:
Suppose that the index consists of the columns "part code" and "section code." Column "part code" consists of ten types of codes, and there are 100 combinations of "part code" with "section code."
In this example, the values to be specified are 10 and 100.

◆Definition for each table

Set item	Specified contents	Default
Number of records	Specify the number of records to be stored.	Cannot be omitted.

Note:
If the table is split into DSIs, (the specified value divided by the number of DSIs) is assigned to each DSI.

◆**Definition for each index DSO**

Set item	Specified contents	Default
Number of different key values	Specify the number of different key values for each combination of columns making up the index. (*1)	Cannot be omitted.

*1 Specification example:
Suppose that the index consists of the columns "part code" and "section code." Column "part code" contains ten types of codes, and there are 100 combinations of "part code" with "section code." In this example, the values to be specified are 10 and 100.

Note:
If the table is split into DSIs, (the specified value divided by the number of DSIs) is assigned to each DSI.
If the calculated result is 1 or less, 1 is assigned to each DSI.

■Output of optimization information

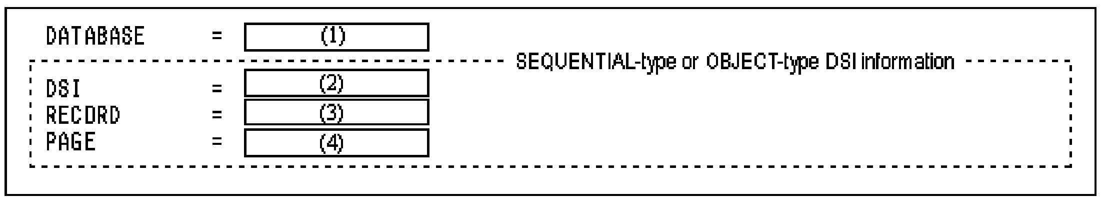
To output the defined optimization information, use the rdbddlex command to specify the PRINT STATISTICS statement. For more information on the PRINT STATISTICS statement, refer to the SQL Reference Guide.

Examples of specification and output given below.

Example 1:

Output example of optimization information defined for each table DSI (data structure: SEQUENTIAL or OBJECT)

```
PRINT STATISTICS FOR DSI STOCK_DSI FILE /rdb2/statistics/DB01/STOCK_DSI
```



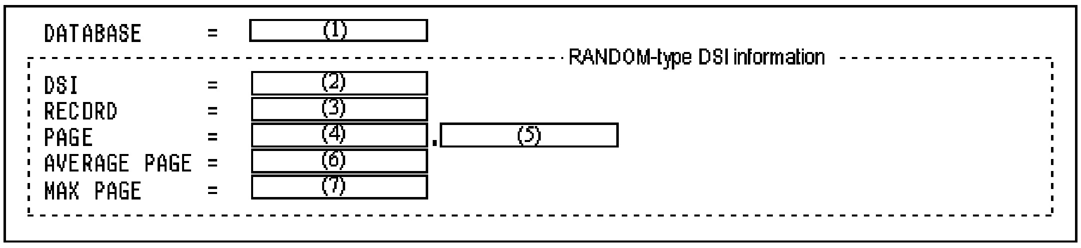
```
DATABASE    =  (1)
DSI         =  (2)
RECORD      =  (3)
PAGE        =  (4)
```

(1) Database name
(2) DSI name
(3) Number of records stored in DSI
(4) Number of pages containing records

Example 2:

Output example of optimization information defined for each table DSI (data structure: RANDOM)

```
PRINT STATISTICS FOR DSI STOCK_DSI FILE /rdb2/statistics/DB01/ STOCK_DSI
```



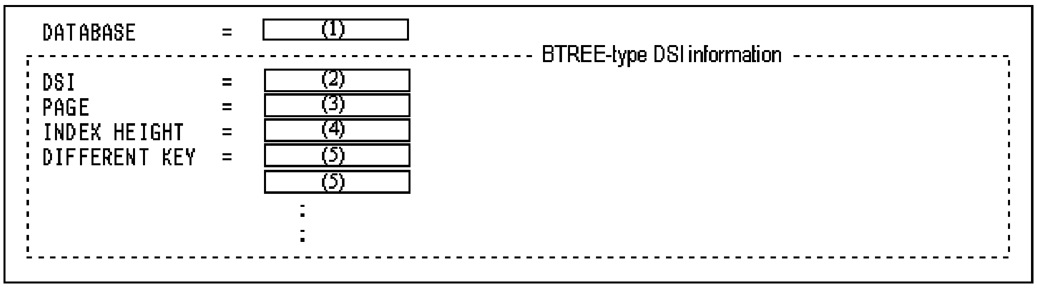
```
DATABASE      =  (1)
---------------------------------- RANDOM-type DSI information ------------
DSI           =  (2)
RECORD        =  (3)
PAGE          =  (4) . (5)
AVERAGE PAGE  =  (6)
MAX PAGE      =  (7)
```

(1) Database name
(2) DSI name
(3) Number of records stored in DSI
(4) Number of pages containing records in PRIME part
(5) Number of pages containing records in OVERFLOW part
(6) Average number of pages in OVERFLOW part for each packet classified by the RULE specification (default is determined by the system) in the DSD definition (For details, refer to Chapter 4, "Strage Structure.")
(7) Maximum number of pages in OVERFLOW part for each packet classified by the RULE specification (default is determined by the system) in the DSD definition (For details, refer to Chatper 4, "Strage Structure.")

Example 3:

Output example of optimization information defined for each index DSI (data structure: BTREE)

```
PRINT STATISTICS FOR DSI PRODUCT_IXDSI FILE /rdb2/statistics/DB01/ PRODUCT_IXDSI
```



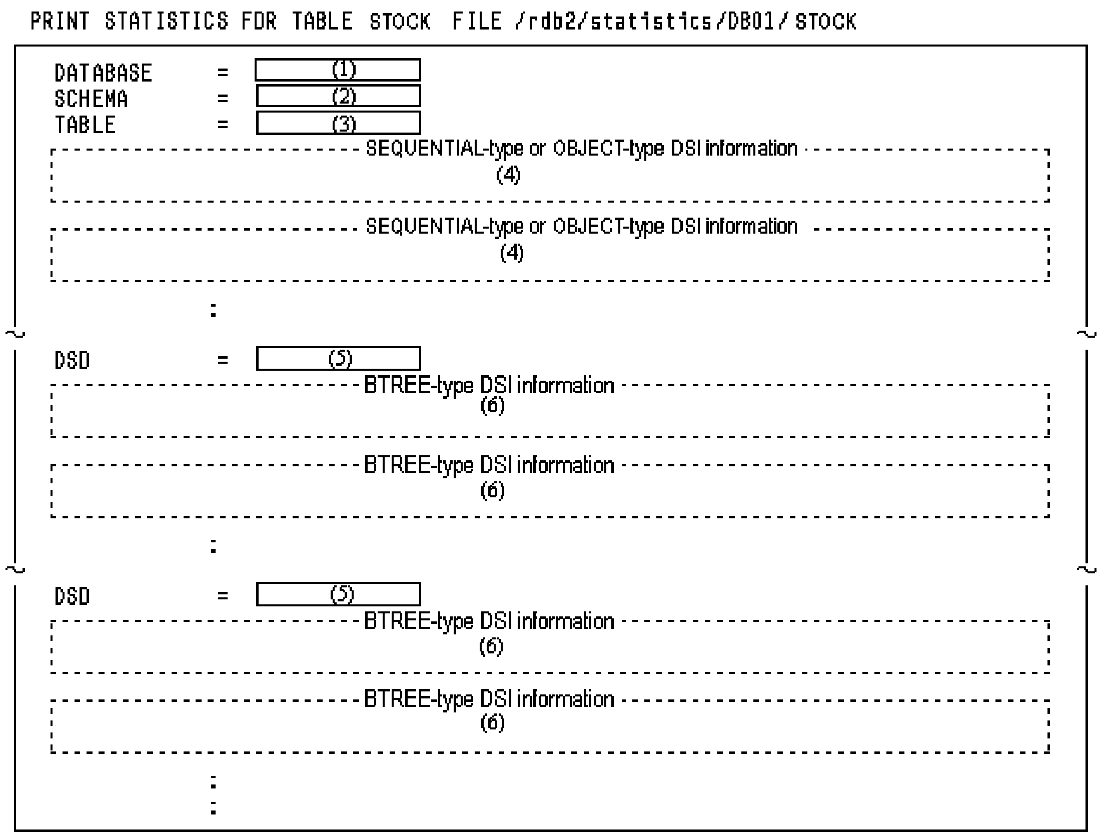
```
DATABASE       =  (1)
---------------------------------- BTREE-type DSI information ------------
DSI            =  (2)
PAGE           =  (3)
INDEX HEIGHT   =  (4)
DIFFERENT KEY  =  (5)
                  (5)
                   :
                   :
```

(1) Database name
(2) DSI name
(3) Number of pages containing records in BASE part
(4) Height of index part (For details, refer to Chapter 4, "Storage Structure.")
(5) Number of different key values of columns making up index

Example: Suppose that the index consists of three columns, "company code," "department code," and "section code." In this example, information about different key values is output as follows:
- Number of different key values for company code
- Number of different key values for combinations of company code with department code
- Number of different key values for combinations of company, department, and section codes

Example 4:

Output example of optimization information defined for each table (data structure: SEQUENTIAL or OBJECT)



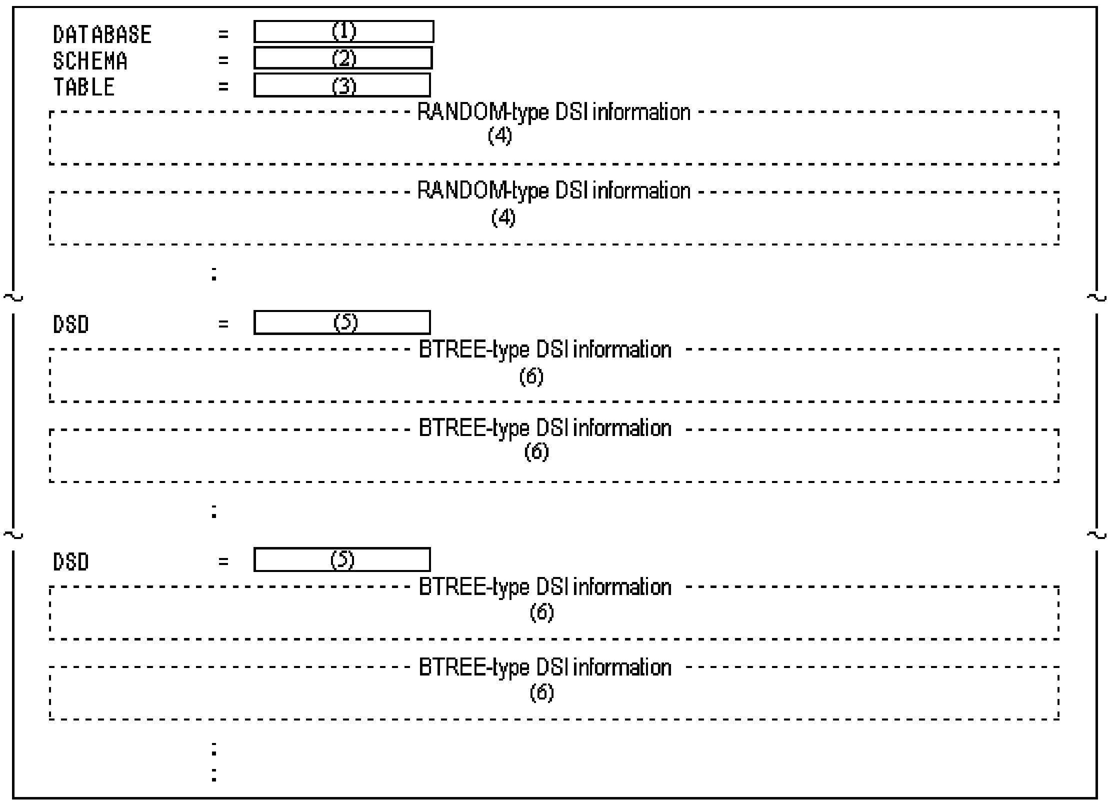
```
PRINT STATISTICS FOR TABLE STOCK  FILE /rdb2/statistics/DB01/STOCK

  DATABASE    =  [ (1) ]
  SCHEMA      =  [ (2) ]
  TABLE       =  [ (3) ]
  ------------- SEQUENTIAL-type or OBJECT-type DSI information -------------
                                  (4)
  ------------- SEQUENTIAL-type or OBJECT-type DSI information -------------
                                  (4)
              :
  DSD         =  [ (5) ]
  ------------------------ BTREE-type DSI information ----------------------
                                  (6)
  ------------------------ BTREE-type DSI information ----------------------
                                  (6)
              :
  DSD         =  [ (5) ]
  ------------------------ BTREE-type DSI information ----------------------
                                  (6)
  ------------------------ BTREE-type DSI information ----------------------
                                  (6)
              :
              :
```

(1) Database name
(2) Schema name
(3) Table name
(4) Optimization information of DSIs making up the table (output by the number of DSIs making up the table.)
(5) Table index DSO name (output by the number of table index DSOs together with the optimization information for index DSIs (6) below.)
(6) Optimization information for DSIs making up the index DSO (output by the number of DSIs making up the index DSO.)

Example 5:

Output example of optimization information defined for each table (data structure: RANDOM)

```
PRINT STATISTICS FOR TABLE STOCK  FILE /rdb2/statistics/DB01/STOCK
```



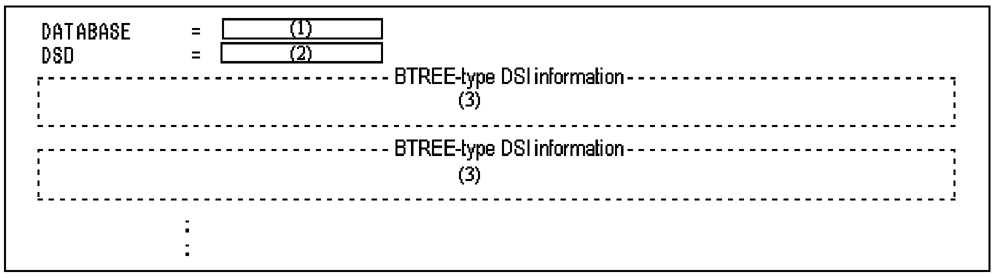
DATABASE = (1)
SCHEMA = (2)
TABLE = (3)
RANDOM-type DSI information (4)
RANDOM-type DSI information (4)
DSO = (5)
BTREE-type DSI information (6)
BTREE-type DSI information (6)
DSO = (5)
BTREE-type DSI information (6)
BTREE-type DSI information (6)

(1) Database name
(2) Schema name
(3) Table name
(4) Optimization information of DSIs making up the table (output by the number of DSIs making up the table.)
(5) Table index DSO name (output by the number of table index DSOs together with optimization information of index DSIs (6) below.)
(6) Optimization information of DSIs making up the index DSO (output by the number of DSIs making up the index DSO.)

Example 6:

Output example of optimization information defined for each index DSO

```
PRINT STATISTICS FOR INDEX PRODUCT_IXDSO FILE /rdb2/statistics/DB01/ PRODUCT_IXDSI
```



DATABASE = (1)
DSO = (2)
BTREE-type DSI information (3)
BTREE-type DSI information (3)

(1) Database name
(2) Index DSO name
(3) Optimization information of DSIs making up the index DSO (output by the number of DSIs making up the index DSO.)

2.13 Generating a Database

A database is generated by the entry of data in the base tables. Data is entered after the storage destination database spaces are created, and the definition of logical structures and storage structures for base tables and indexes is completed.

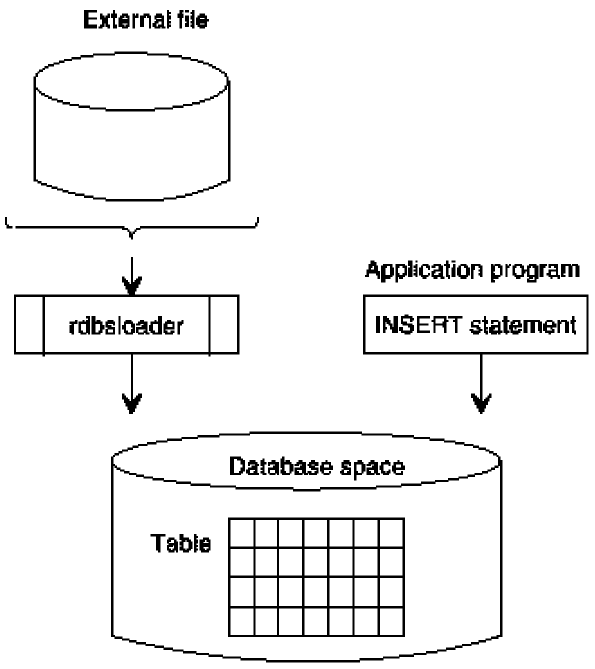
■Database generation methods

A database can be generated in the following two ways:

- By using a data file and the rdbsloader command (DSI initialization unnecessary)
- By creating an application program for database generation and using the INSERT statement (DSI initialization required)

Figure: Database generation methods provides an overview of the two database generation methods: using the rdbsloader command and using the INSERT statement of an application program.

[Figure: Database generation methods]



◆Generating a database by using the INSERT statement

An example of generating a database by using the INSERT statement follows.

For details on the INSERT statement, refer to the "SQL Beginner's Guide."

```
#include<stdio.h>

main(){
    EXEC SQL BEGIN DECLARE SECTION;
        char SQLSTATE[6];
        char SQLMSG[256];
        short hcustomer;
        short hprodno;
        long hprice;
        short horderqty;
    EXEC SQL END DECLARE SECTION;

    char PROC_NO;

    for(;;){
        printf("Enter data      ... 1 \n");
        printf("End data entry ... 2 \n");
        scanf("%c",&PROC_NO);

        switch(PROC_NO) {
            case '1':
              printf("Enter customer number.\n");
              scanf("%d,&hcustomer);
              printf("Enter customer number.\n");
              scanf("%d,&hprodno);
                      :
                      :
              EXEC SQL INSERT INTO STOCKS.ORDER(CUSTOMER,PRODNO,
                                                  PRICE,ORDERQTY)
                       VALUES(:hcustomer,:hprodno,:hprice,:horderqty);
              continue;
            case'2':
              break;
        }
     break;
    }
    EXEC SQL COMMIT WORK;
                       :
                       :
}
```

◆Generating a database by using the rdbsloader command

A sample for generating a database by using the rdbsloader command follows.

UNIX

```
rdbsloader -mi -i STOCK_DB.EAST_ORDER_DSI /home/rdb/kantou.data
```

Windows NT/2000/XP

```
rdbsloader -mi -i STOCK_DB.EAST_ORDER_DSI C:\DEFAULT\USERS\KANTOU.DATA
```

For information about the rdbsloader command, refer to "RDB Operations Guide."

2.14 Referencing Database Definition Information

After the database is defined, verify the database definition information. This section explains how to print database definition information.

The rdbprt command prints the database name list and definition information. The rdbddlex command with the PRINT STATISTICS statement or the rdbups command prints the optimization information. These informations can be used to perform database management tasks such as confirming the range of database usage.

For details on the PRINT STATISTICS statement, refer to the SQL Reference Guide. For details on the rdbups and rdbprt commands, refer to the RDB Operations Guide, or the man command (under UNIX) or the SymfoWARE/RDB online manual (under Windows NT).

This section explains the information printed by the rdbprt command, the command specification method, and the print format.

2.14.1 Information printed by the rdbprt command

The rdbprt command prints the following information:

- Database name list information
  Database name list information lists the names of all databases under the target SymfoWARE/RDB.
- User name list information
  User name list information is a list of the names of all users under the target SymfoWARE/RDB.
- Role name list information
  Role name list information is a list of the names of all roles under the target SymfoWARE/RDB.
- Definition information
  Definition information includes information about a database, schema, base table, view table, temporary table, trigger, routine, DSO, DSI, scope, database space, sequence, and user, and role definitions.
- User parameter information
  User parameter information includes user parameter information used by the target SymfoWARE/RDB.

To print database name list information, specify DB in the m option of the rdbprt command. To print definition information, specify DEF in the m option of the rdbprt command. The definition information print specifications are:

- DB
- SCHEMA
- TABLE
- TRIGGER
- ROUTINE
- DSO
- DSI
- SCOPE
- DBSPACE
- SEQUENCE
- USER
- ROLE

■Items printed in definition information

The information generated for each print specification type is given next.

◆DB specification

- Database name
- Database creator
- Database definition date and time
- Names of schemas belonging to the database
- Names of database spaces belonging to the database
- Log group name of log environment used for database space belonging to the database
- Names of scope belonging to the database

◆SCHEMA specification

- Schema name
- Name of database to which the schema belongs
- Schema creator
- Schema definition date and time

· Schema comment definition
· Privilege information (only when -p is specified)
· Names of tables belonging to the schema
· Table type (base, view, or temporary table)
· Names of routines belonging to a schema
· Types of routines belonging to a schema
· Names of triggers belonging to a schema
· Names of sequences belonging to a schema

◆TABLE specification

· Table, view table, or temporary table name
· Name of database to which the table belongs
· Name of schema to which the table belongs
· Type (base, view, or temporary table)
· Base, view, or temporary table creator
· Table definition date and time and update date and time
· Table comment definition
· Privilege information (only when -p is specified)
· Column information (column name, existence of NOT NULL specification, byte length, data type, DEFAULT value)
· Table record length
· Table constraint information (PRIMARY KEY information and UNIQUE information)
· PRIMARY KEY information (component column names)
· UNIQUE information (component column names)
· Table DSO name
· Index DSO name
· Routine name
· Routine type
· View information (reference view information, configuration view information, query expression, indication of whether WITH CHECK OPTION is specified, updatability)
· Trigger name
· Name of the database space to which a temporary table is allocated
· Number of users of the database space to which a temporary table is allocated

◆TRIGGER specification

· Trigger name
· Name of database to which the trigger belongs
· Name of schema to which the trigger belongs
· Trigger creator
· Trigger definition date and time
· Name of schema to which the trigger-target table belongs
· Name of trigger-target table
· Type of trigger-target table
· Trigger operation point
· Trigger event
· Name of column to be updated in the trigger-target table
· Correlation name of old and new values in the trigger-target table.
· Execution unit of triggered operation
· Execution condition of triggered operation
· Table name included in the SQL statement of triggered operation
· Name of schema to which the table included in the SQL statement of the triggered operation belongs
· Type of table included in the SQL statement of triggered operation
· SQL statement of triggered operation
· Routine name included in the SQL statement of a triggered operation
· Name of the schema to which the routine included in the SQL statement of a triggered operation belongs
· Type of routine included in the SQL statement of a triggered operation
· Sequence name included in the SQL statement of a triggered operation
· Name of the schema to which a sequence included in the SQL statement of a triggered operation belongs

◆Routine specification

· Routine name
· Name of the database to which a routine belongs
· Name of the schema to which a routine belongs
· Routine creator

· Routine definition date and time
· Routine comment definition
· Privilege information (only if -p is specified)
· Parameter information (parameter names, parameter types, modes)
· Related table information (schema name, table name, type)
· Name of a routine called by this routine, name of the schema to which the called routine belongs, and type of routine
· Name of a routine that calls this routine, name of the schema to which the called routine belongs, and type of routine
· Procedure routine definition statement
· Symbol name of the execution module that processes the function routine
· Path name of the execution module that processes the function routine

◆DSO specification

· DSO name
· Type (table DSO or index DSO)
· Name of database to which DSO belongs
· Related table information (schema name and table name)
· DSO creator
· DSO definition date and time
· Data structure type (SEQUENTIAL, RANDOM, OBJECT, or BTREE, page size for each part, information [column names] about columns forming the cluster key, existence of NOT UNIQUE specification, storage order assurance level, page reuse point, and contents of RULE specification)
· Base representation (BY ADDRESS or BY KEY)
· Split condition for subdividing into DSIs
· Index degeneration specification
· Information about columns making up the index (column name, data type)
· DSI information (DSI name)

◆DSI specification

· DSI name
· Type (table DSI or index DSI)
· Name of database to which DSI belongs
· Related table information (schema name and table name)
· Related DSO information (DSO name)
· Related table DSI name
· Related index DSI name
· DSI creator
· DSI definition date and time
· Split key value
· Page reuse point
· Index degeneration specification
· Database space allocation information (allocation part, database space name, allocation size)
· Alarm point and expansion specification information (expansion point, allocation size, allocation candidate database space name)
· Name of the scope that limits DSI

◆SCOPE specification

· Scope name
· Name of database to which the scope belongs
· User ID of the user who defined the scope
· Scope definition date and time
· Name of DSI limited by scope
· User ID of the user to which the scope is applied

◆DBSPACE specification

· Database space name
· Name of database to be allocated
· Raw device name (for UNIX)
· File name
· Device type
· Database space creator
· Database space definition date and time
· Log group name of log environment used for database space

- Privilege information (only if -p is specified)
  Note:
  If a DSI exists for which database space has been allocated, the following information is also printed:
    - Information on the allocated DSI (DSI name and type)
    - Information on the table related to the DSI (schema name and table name)
    - Information on the DSO related to the DSI (DSO name)
    - Allocation size

◆SEQUENCE specification

- Sequence name
- Name of the database to which a sequence belongs
- Name of the schema to which a sequence belongs
- Sequence creator
- Sequence definition date and time
- Sequence number increment interval
- Initial sequence number
- Maximum sequence number
- Minimum sequence number
- Whether to use sequence numbers cyclically
- Number of memory spaces allocated to sequence numbers and number of memory spaces retained with sequence numbers
- Whether to ensure a sequence between clusters

◆USER specification

- User name
- User type
- User management type
- Password status
- User creator
- User definition date and time
- Last date and time when the user information was altered (using the ALTER USER statement)
- Last date and time when the user was connected to a database
- Number of times the user failed to connect to a database
- Name and value of each parameter set for the user

◆ROLE specification

- Role name
- User who defined the role
- Role definition date and time
- Role update date and time
- Resource type
- Database name
- Schema name
- Table name (output if the resource type is TABLE)
- Procedure routine name (output if the resource type is PROCEDURE)
- Function routine name (output if the resource type is FUNCTION)
- Sequence name (output if the resource type is SEQUENCE)
- Trigger name (output if the resource type is TRIGGER)
- Schema name (output if the resource type is SCHEMA)
- Database space name (output if the resource type is DBSPACE)
- Privilege type
- Grantor name
- Grantee names
- Whether the user has the privilege to grant a role

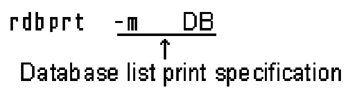
2.14.2 rdbprt command specification method

Figure: Sample database list print specification and Figure: Sample definition information print specification show sample specifications of the rdbprt command. For more information on specifying the rdbprt command, refer to the man command (under UNIX) or the SymfoWARE/RDB online manual (under Windows NT).

■Sample database list print specification

This example prints a list of all databases in a target SymfoWARE/RDB system.

[Figure: Sample database list print specification]

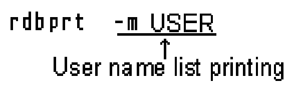


```
rdbprt  -m   DB
```

■Sample user name list print specification

This example specifies printing a list of names of all users under the target SymfoWARE/RDB.

[Figure: Sample user name list print specification]

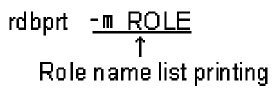


```
rdbprt  -m USER
```

■Sample role name list print specification

This example specifies printing a list of names of all roles under the target SymfoWARE/RDB.

[Figure: Sample role name list print specification]

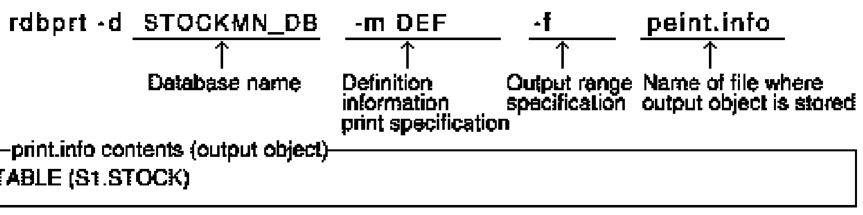


```
rdbprt  -m ROLE
```

■Sample definition information print specification

If TABLE is specified for the output object and the f option is specified in the rdbprt command, this example prints not only the table information but also the related DSO and DSI information.

[Figure: Sample definition information print specification]



```
rdbprt -d STOCKMN_DB  -m DEF   -f   peint.info
```

Database name
Definition information print specification
Output range specification
Name of file where output object is stored
print.info contents (output object)

```
TABLE (S1.STOCK)
```

■Sample user parameter information print specification

This example specifies printing a list of user parameter information defined in the target SymfoWARE/RDB.

[Figure: Sample user parameter information print specification]

```
  rdbprt  -m PARAM
               ↑
User parameter information
```

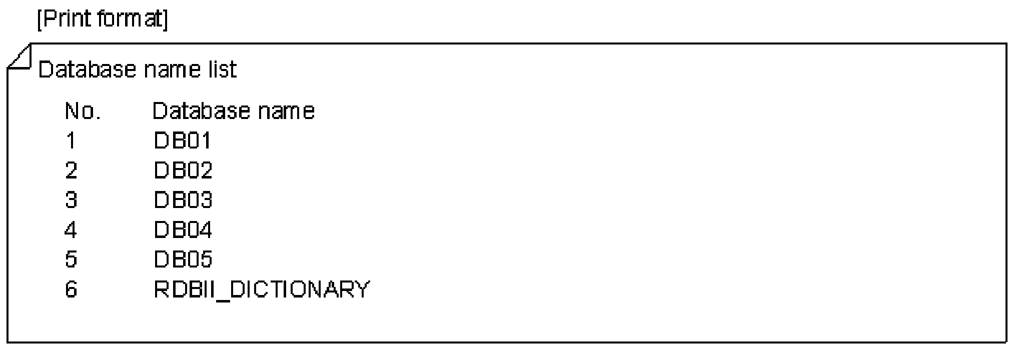
2.14.3 rdbprt command print format

Figure: Sample database list print results shows a sample printout for the rdbprt command specified in Figure: Sample database list print specification. Figure: Sample database output object specification and print results shows a sample printout for the rdbprt command specified in Figure: Sample definition information print specification. Figure: Definition information output format for a DB specification to Figure: Definition information output format for a SCOPE specification show the rdbprt command definition information output formats.

The following figure enable readers to see a print image. However, these figures are not complete figures.

■Sample database list print results

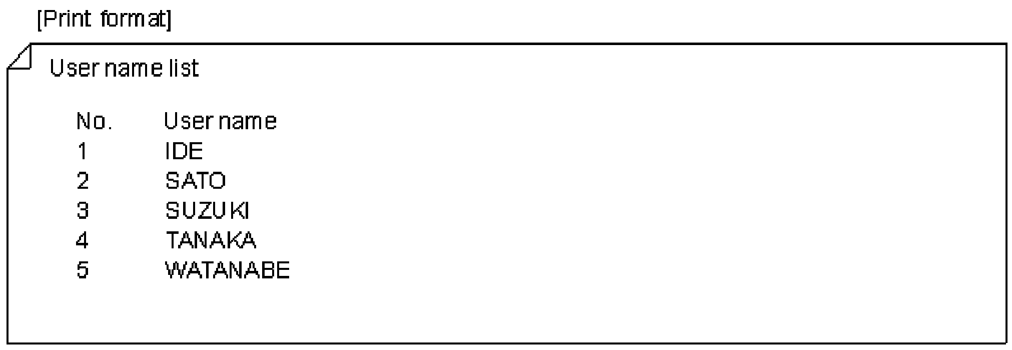
[Figure: Sample database list print results]



```
[Print format]
Database name list
  No.    Database name
  1      DB01
  2      DB02
  3      DB03
  4      DB04
  5      DB05
  6      RDBII_DICTIONARY
```

■Sample user name list print results

[Figure: Sample user name list print results]



```
[Print format]
 User name list
   No.    User name
   1      IDE
   2      SATO
   3      SUZUKI
   4      TANAKA
   5      WATANABE
```

■Sample role name list print results

[Figure: Sample role name list print results]

[Print format]

```
Role name list

  No.     Role name
  1       STOCKS_A2
  2       STOCKS_B1
  3       STOCKS_C3
```

■Sample user parameter information print results

[Figure: Sample user parameter information print results]

[Print format]

```
Parameter information
  INVALID_PASSWORD_TIME = 5
  INVALID_PASSWORD_WAIT_TIME = 4
  MIN_PASSWORD_SIZE = 6
  PASSWORD_CHANGE_TIME = 0
  PASSWORD_LIMIT_TIME = UNLIMITED
  USER_CONTROL = YES
```

Remarks: DEFAULT_ROLE is not printed in user parameter information.

It is printed in USER specification in definition information.

■Sample database output object specification and print results

[Figure: Sample database output object specification and print results]

[Output object specification]

```
TABLE (S1.STOCK)
```

[Print format]

```
NO. 1         Table name ...... STOCK

  Database name ...... STOCK_DB
  Schema name   ...... STOCKS
  Type          ...... BASE
  Creator       ...... SUZUKI
  Created date  ...... Mon Apr 1  17:06:25  2002
  Updated date  ...... Tue Apr 2  10:00:30  2002

  Comment
    'STOCK ITEMS, STOCK QUANTITIES, AND WAREHOUSES TABLE'

  Column information

    Column name         ...... ITMNO
    NOT NULL constraint ...... YES          Data length ...... 2(Byte)
    Data type           ...... SMALLINT
    Default             ......

    Comment
      'PRODUCT-NO'

    Column name         ...... PRODUCT


  Constraint information
  Primary key constraint information
    No.       Column name
    1         ITMNO

  Unique Constraint information
    No.1           Unique constraint
      No.          Column name
      1            ITMNO
      2            PRODUCT

  DSO information
    Base DSO name ...... STOCK_DSO
    Index DSO information
      No.          DSO name
      1            PRODUCT_IXDSO
      2            PRODUCT
```

■Definition information output format for a DB specification

[Figure: Definition information output format for a DB specification]

```
Database name ......  (1)

  Creator      ...... (2)
  Created date ...... (3)

  Schema information
  No.       Schema name
  1             (4)
  2             (4)
  3             (4)

  Database space information
  No.       Database space name    Log group name
  1             (5)                      (6)
  2             (5)                      (6)
  3             (5)                      (6)

  Scope information
  No.
  1             (7)
  2             (7)
  3             (7)
~                                                          ~
```

(1) base name

(2) Database creator

(3) Database name registration date and time

("day-of-week month day   hour:minutes:seconds calendar-year")

(4) Name of schema belonging to the database

(5) Name of database space belonging to the database

(6) Log group name of log environment used for database space belonging to the database

(7) Name of scope belonging to the database

■Definition information output format for a SCHEMA specification

[Figure: Definition information output format for a SCHEMA specification]

```
No. 1         Schema name ......         (1)

  Database name     ......         (2)
  Creator           ......         (3)
  Created date      ......         (4)

  Comment
    '         (5)        '
  Table information
    No.       Table name          Type
    1            (6)               (7)
    2            (6)               (7)
    3            (6)               (7)

  Routine information
    No.     Routine name    Routine type
    1            (8)               (9)
    2            (8)               (9)
    3            (8)               (9)

  Trigger information
    No.     Trigger name
    1            (10)
    2            (10)
    3            (10)
  Sequence information
    No.     Sequence name
    1             (11)
    2             (11)
    3             (11)
                   :
```

(1) Schema name

(2) Name of the database to which the schema belongs

(3) Schema creator

(4) Schema definition date and time

("day-of-week month day hours:minutes:seconds calendar-year")

Example: "Mon Apr 1 17:06:25 2002"

(5) Schema comment definition

(6) Name of a table belonging to the schema

(7) Table type

BASE	: Base table
VIEW	: View table
GLOBAL TEMPORARY	: Temporary table

(8) Name of a routine belonging to the schema

(9) Routine type

PROCEDURE	: Procedure routine
FUNCTION	: Function routine

(10) Name of a trigger belonging to the schema

(11) Name of a sequence belonging to the schema

■Definition information output form at for a TABLE specification

[Figure: Definition information output format for a TABLE specification]

```
No. 1         Table name ......       (1)

  Database name ......      (2)
  Schema name   ......      (3)
  Type          ......      (4)
  Creator       ......      (5)
  Created date  ......      (6)
  Updated date  ......      (7)                                               (*1)

  Comment
   '    (8)      '

  Column information

    Column name          ......       (9)
    NOT NULL constraint  ......      (10)      Data length ......  (11) (Byte)
    Data type            ......      (12)
    Default              ......      (13)

    Comment
     '   (8)      '

    Column name          ......       (9)
    NOT NULL constraint  ......      (10)      Data length ......  (11) (Byte)
    Data type            ......      (12)
    Default              ......      (13)

    Comment
     '   (14)     '

  Record length ......       (15)

  Constraint information                                                      (*1)
  Primary key constraint information                                          (*1)
    No.       Column name
    1           (16)
    2           (16)

  Unique Constraint information                                               (*1)
    No. 1        Unique constraint
      No.       Column name
      1           (17)
      2           (17)
    No. 2        Unique constraint
      No.       Column name
      1           (17)
      2           (17)

  Condition of view                                                           (*2)
                  (18)

  With check option  ......  (19)                                             (*2)

  Updatable view     ......  (20)                                             (*2)


  Preserve option    ......  (21)                                             (*4)

  View information
                 (22)                                                         (*3)

  DSO information                                                             (*5)
    Base DSO name ......      (23)
    Index DSO information
      No.         DSO name
      1             (24)
      2             (24)

  Routine information
    No.  1  Routine name ......    (25)
      Schema name  ......    (26)
      Routine type ......    (27)

  Trigger information
    No.  1  Trigger name ......    (28)
      Schema name  ......    (29)

  Triggered information
    No.  1  Trigger name ......    (30)
      Schema name  ......      (31)

  Used database space information                                            (*4)
    No.  Database space name       User number
    1           (32)                 (33)
    2           (32)                 (33)
```

*1 Output only if the specified table is a base table

*2 Output only if the specified table is a view table

*3 Output only if the specified table has a related view table

*4 Output only if the specified table is a temporary table

*5 Output only if the specified table is a base or temporary table

(1) Table name

(2) Name of the database to the which the table belongs

(3) Name of the schema to which the table belongs

(4) Table type

BASE : Base table

VIEW : View table

GLOBAL TEMPORARY: Temporary table

(5) Table creator

(6) Table definition date and time

("day-of-week month day hours:minutes:seconds calendar-year")

Example: "Mon Apr 1 17:06:25 2002"

(7) Last date and time when the table was altered (using ALTER TABLE)

("day-of-week month day hours:minutes:seconds calendar-year")

(8) Table comment definition

(9) Name of a column defined for the table

(10) Whether NOT NULL is specified in the column definition

YES: NOT NULL is specified.

NO: NOT NULL is not specified.

(11) Column data length (in bytes)

(12) Column data type

(13) Value specified in the DEFAULT clause

(14) Column comment definition

(15) Value obtained by adding 1 for each column containing NO in field (10)
to the total of values specified in field (11) for all columns

(16) Name of a column forming the unique constraint (PRIMARY KEY)

(17) Name of a column forming the unique constraint (UNIQUE)

(18) View definition query expression

(19) Whether WITH CHECK OPTION is specified in the view definition

YES: WITH CHECK OPTION is specified.

NO: WITH CHECK OPTION is not specified.

(20) View table updatability

YES: Updatable view

NO: Read-only view

(21) Temporary table valid range

YES: Valid within a session

NO: Valid within a transaction

(22) View information related to the table

Details are given in "View information output format."

(23) Table DSO name defined for the table

(24) DSO name of an index defined of the table

(25) Name of a routine that references the table

(26) Name of the schema to which the routine belongs

(27) Type of routine

PROCEDURE: Procedure routine

FUNCTION: Function routine

(28) Name of a trigger where a change to the table is specified as the trigger event

(29) Name of the schema to which the trigger belongs

(30) Name of a trigger for which the SQL statement of the triggered operation
contains the table

(31) Name of the schema to which the trigger belongs

(32) Name of a database space to which the temporary table is allocated

(33) Number of users of the database space to which the temporary table is allocated

[Figure: View information output format]

```
Referred to following view information

      No. 1         View name  ......         (1)              ┐
                                                               │
        Database name ......      (2)                          ├(*1)
        Schema name   ......      (3)                          │
        Creator       ......      (4)                          │
        Created date  ......      (5)                          ┘

Consist of following table/view information                    ┐
                                                               │
      No. 1         Table/view name  ......   (6)              │
                                                               │
        Database name ......      (7)                          │
        Schema name   ......      (8)                          │
        Type          ......      (9)                          │
        Creator       ......      (10)                         ├(*2)
        Created date  ......      (11)                         │
                                                               │
        Column information                                     │
                                                               │
          Column name          ......    (12)                  │
          NOT NULL constraint  ......    (13)                  │
          Data length          ......    (14)  (Byte)          │
          Data type            ......    (15)                  │
          Default              ......    (16)                  ┘
```

*1 Displayed only when a view that references the specified table exists

*2 Displayed only when the specified table is a view

(1) Name of view that references the specified table

(2) Name of database to which the view that references the specified table belongs

(3) Name of schema to which the view that references the specified table belongs

(4) Creator of view that references the specified table

(5) Definition date and time ("day-of-week month day hour:minutes:seconds calendar-year") of view that references the specified table

Example: "Mon Apr 1 17:06:25 2002"

(6) Name of table from which the view was derived

(7) Name of database to which the table from which the view was derived belongs

(8) Name of schema to which the table from which the view was derived belongs

(9) Type of table from which the view was derived

BASE: Base table

VIEW: View

GLOBAL TEMPORARY: TEMPORARY TABLE

(10) Creator of table from which the view was derived

(11) Definition date and time ("day-of-week month day hour:minutes:seconds calendar-year") of table from which the view was derived

Example: "Mon Apr 1 17:06:25 2002"

(12) Name of column forming the base table or view from which the view was derived

(13) Existence of NOT NULL specification for the column specified in field (12)

YES: NOT NULL specification exists.

NO: NOT NULL specification does not exist.

(14) Data length (units: bytes) for the column specified in field (12)

(15) Data type for the column specified in field (12)

(16) DEFAULT clause definition contents for the column specified in field (12)

■Definition information output format for a ROUTINE specification

[Figure: Definition information output format for a ROUTINE specification]

```
No. 1         Routine name ......     (1)

  Database name ......     (2)
  Schema name   ......     (3)
  Creator       ......     (4)
  Created date  ......     (5)
  Routine type  ......     (6)

  Comment
   '   (7)     '

  Parameter information                                        (*1)

   Parameter name ......   (8)
   Parameter type ......   (9)
   Parameter mode ......   (10)
              :
  Parameter information                                        (*2)

   Parameter mode ......   IN
   Parameter type ......   (11)
              :
   Parameter mode ......   RETURN
   Parameter type ......   (12)

  Using table information
    No.   1   Table name  ......    (13)
     Schema name ......     (14)
     Type        ......     (15)
  Called routine information
    No.   1   Routine name ......   (16)
     Schema name  ......    (17)
     Routine Type ......    (18)
  Calling routine information
    No.   1   Routine name ......   (19)
     Schema name  ......    (20)
     Routine Type ......    (21)

  Routine text information                                     (*1)
                (22)
                (22)
              :
  Routine Function information                                 (*2)
    Language      ......    C
    Symbol        ......   (23)
    Library       ......   (24)
```

*1 Output only if a parameter is specified for the procedure routine

*2 Output only if a parameter is specified for the function routine

(1) Routine name

(2) Name of the database to which the routine belongs

(3) Name of the schema to which the routine belongs

(4) Routine creator

(5) Routine definition date and time

("day-of-week month day hours:minutes:seconds calendar-year")

Example: "Mon Apr 1 17:06:25 2002"

(6) Routine type

PROCEDURE: Procedure routine

FUNCTION: Function routine

(7) Routine comment definition

(8) Name of a parameter of the procedure routine

(9) Data type of the parameter of the procedure routine

(10) Mode of the parameter of the procedure routine

IN: Input

OUT: Output

(11) Data type of an input parameter of the function routine

(12) Type of return data of the function routine

(13) Name of the table contained in the routine

(14) Name of the schema to which the table contained in the routine belongs

(15) Type of table contained in the routine

BASE: Base table

VIEW: View table

GLOBAL TEMPORARY: Temporary table

(16) Name of a routine called by routine

(17) Name of the schema to which routine (16) called by routine (1) belongs

(18) Type of routine (16) called by routine (1)

PROCEDURE: Procedure routine

FUNCTION: Function routine

(19) Name of a routine that calls routine (1)

(20) Name of the schema to which routine (19) that calls routine (1) belongs

(21) Type of the routine (19) that calls routine (1)

PROCEDURE: Procedure routine

FUNCTION: Function routine

(22) Definition statements for the procedure routine

(source text from "CREATE PROCEDURE" to "END label-name")

(23) Symbol name of the execution module that processes the function routine

(24) Path name of the execution module that processes the function routine

■Definition information output format for a DSO specification

[Figure: Definition information output format for a DSO specification]

```
No. 1         DSO name ......    (1)

  Usage type    ......    (2)
  Database name ......    (3)
  Schema name   ......    (4)
  Table name    ......    (5)
  Creator       ......    (6)
  Created date  ......    (7)

  Data structure information
    Type                ......   (8)
    Page size (  (9)  )......   (10) (K byte)
    Page size (  (9)  )......   (10) (K byte)
    Order               ......   (11)                                    (*1)
    Reuse page point    ......   (12) (%)                                (*1)
    Index degenerate    ......   (13)                                    (*2)

    Cluster key information                                              ┐
      Type      ......    (14)                                           │
      No.       Column name                                              ├(*1)
      1           (15)                                                   │
      2           (15)                                                   │
      3           (15)                                                   ┘

    Rule information                                                     ┐
                  (16)                                                   ┘(*1)

  Base expression ......  (17)                                           (*2)

  Divide condition                                                       ┐
                  (18)                                                   ┘(*1)
  Key column information                                                 ┐
    No.       Column name         Data type                              │
    1           (19)                (20)                                 ├(*2)
    2           (19)                (20)                                 │
    3           (19)                (20)                                 ┘

  DSI information
    No.         DSI name
    1             (21)
    2             (21)
    3             (21)
~                                                                        ~
```

*1 Displayed only for a table DSO

*2 Displayed only for an index DSO

(1) DSO name

(2) DSO type

BASE: Table DSO

INDEX: Index DSO

(3) Name of database to which the DSO belongs

(4) Name of schema to which the table having the defined DSO belongs

(5) Name of table for which the DSO is defined

(6) DSO creator

(7) DSO definition date and time

("day-of-week month day hour:minutes:seconds calendar-year")

Example: "Mon Apr 1 17:06:25 2002"

(8) Table or index data structure

(9) Allocation part for which the page size (field(10)) is displayed

(10) Page size of the allocation part (units: kilobytes)

(11) ORDER specification (0 or 1)

Note: Displayed only when the data structure is SEQUENTIAL

(12) Page reuse point (units: %)

Note: Displayed only when the data structure is SEQUENTIAL and
ORDER(1) is specified

(13) Index degeneration specification

YES: Degeneration specification exists.

NO: No degeneration specification exists.

(14) Cluster key type

NOT UNIQUE: Cluster key having a NOT UNIQUE specification

Note: Displayed only for a cluster key having a NOT UNIQUE specification

(15) Name of column forming the table cluster key

(16) Contents of RULE specification

Note: Displayed only when the data structure is RANDOM and
RULE is specified

(17) Base representation

BY KEY: Displayed when the data structure of the table to which
the DSO belongs is RANDOM

BY ADDRESS: Displayed when the data structure of the table to which
the DSO belongs is SEQUENTIAL or OBJECT

(18) Split condition

(19) Name of column forming the index

(20) Data type of column forming the index

(21) DSI name defined for the DSO

■Definition information output format for a DSI specification

[Figure: Definition information output format for a DSI specification]

```
No. 1        DSI name ......    (1)

  Usage type    ......    (2)
  Database name ......    (3)
  Schema name   ......    (4)
  Table name    ......    (5)
  DSO name      ......    (6)
  Creator       ......    (7)
  Created date  ......    (8)

  Base DSI information                                                  ┐
      No.         Base DSI name                                         ├(*1)
      1                (9)                                              ┘

  Related Index DSI information                                         ┐
      No.         Index DSI name                                        │
      1                (10)                                             │
      2                (10)                                             ├(*2)
                                                                        │
  Divide value                                                          │
    Using              (11)                                             │
                                                                        ┘
  Reuse page point   ...... (12)  (%)
  Index degenerate   ...... (13)                                         (*1)

  Allocation information
    No. 1         Allocation target ......   (14)
      Used database space information
      No.         Database space name  Allocate size     Status
      1                 (15)            (16)  (K Byte)     (17)
      2                 (15)            (16)  (K Byte)     (17)
      3                 (15)            (16)  (K Byte)     (17)

    No. 2         Allocation target ......   (14)
      Used database space information
      No.         Database space name  Allocate size     Status
      1                 (15)            (16)  (K Byte)     (17)
      2                 (15)            (16)  (K Byte)     (17)
      3                 (15)            (16)  (K Byte)     (17)

  Alarm information                                                     ┐
    Alarm point     ......    (18) (K Byte)                             │
    expand point    ......    (19) (K Byte)                             │
    Allocation size ......    (20) (K Byte)                             │
    No. 1         Allocation target ......   (21)                       ├(*3)
      Stand by database space information                               │
      No.         Database space name                                   │
      1                 (22)                                            │
      2                 (22)                                            │
      3                 (22)                                            ┘
  Scope information
    No.         Scope name
    1             (23)
    2             (23)
    3             (23)
```

*1 Displayed only for an index DSI

*2 Displayed only for a table DSI

*3 Displayed only when the rdbalmdsi command has been set

(1) DSI name

(2) DSI type

BASE: Table DSI

INDEX: Index DSI

(3) Name of database to which the DSI belongs

(4) Name of schema to which the table having the defined DSI belongs

(5) Name of table for which the DSI is defined

(6) Name of DSO to which the DSI is belongs

(7) DSI creator

(8) DSI definition date and time

("day-of-week month day hour:minutes:seconds calendar-year")

Example: "Mon Apr 1 17:06:25 2002"

(9) DSI name of table to which the DSI belongs

(10) DSI name of index related to the DSI

(11) Split values (displayed in value definition order)

(12) Page reuse point (units: %)

Note: Displayed only when ORDER(1) is specified in the table DSO definition statement.

(13) Index degeneration specification

YES: Degeneration specification exists.

NO: No degeneration specification exists.

(14) Allocation part for which database space information (fields (15) to (17)) is displayed

(15) Allocation target database space name

(16) Allocation size (units: kilobytes)

(17) Allocation point

INT: Allocated when DSI is defined and rdbgcdsi is executed

EXP: Allocated when size is automatically expanded

(18) Alarm point (units: kilobytes)

(19) Expansion point (units: kilobytes) for size expansion

(20) Allocation size (units: kilobytes) for size expansion

(21) Allocation element for which size is to be expanded

(22) Name of allocation database space for which size is to be expanded

(23) Name of scope that limits DSI

■Definition information output format for a DBSPACE specification

[Figure: Definition information output format for a DBSPACE specification]

```
No. 1         Database space name ......     (1)

  Database name   ......      (2)
  Raw device name ......      (3)
  File name       ......      (4)
  Device kind     ......      (5)
  Allocate  size  ......      (6)
  Creator         ......      (7)
  Created date    ......      (8)
  Log group       ......      (9)

  DSI information
  No. 1         DSI name ......      (10)
    Usage type    ......   (11)
    Schema name   ......   (12)
    Table name    ......   (13)
    DSO name      ......   (14)
    Allocate size ......   (15)  (K byte)

  No. 2         DSI name ......      (10)
    Usage type    ......   (11)
    Schema name   ......   (12)
    Table name    ......   (13)
    DSO name      ......   (14)
    Allocate size ......   (15) (K byte)

  No. 3         DSI name ......      (10)
    Usage type    ......   (11)
    Schema name   ......   (12)
    Table name    ......   (13)
    DSO name      ......   (14)
    Allocate size ......   (15) (K byte)
```

(1) Database space name

(2) Name of database to which the database space belongs

(3) Name of raw device where the database space is created

Displayed only when a raw device is used.

(4) Name of the file used to create the database space

Displayed only when a network or local file is used
(A network file can be used with SymfoWARE Server Enterprise Extended Edition, SymfoWARE Server Enterprise Edition, and SymfoWARE Server Standard Edition)

(5) Device type

RAWDEVICE: Raw device (displayed only on a UNIX system)

LOCAL FILE: Local file (displayed only in Windows NT/200/XP)

(6) Allocation size of a database space (in kilobytes)

(7) Database space creator

(8) Database space definition date and time

("day-of-week month day hours:minutes:seconds calendar-year")

Example: "Mon Apr 1 17:06:25 2002"

(9) Log group name of the log environment used by the database space

(10) DSI name defined in the database space

(11) DSI type

BASE: Table DSI

INDEX: Index DSI

(12) Name of schema to which the table having the defined DSI belongs

(13) Name of table for which the DSI is defined

(14) Name of DSO to which the DSI belongs

(15) Amount assigned by DSI (units: kilobytes)

■Definition information output format for a TRIGGER specification

[Figure: Definition information output format for a TRIGGER specification]

```
No. 1         Trigger name ......    (1)

  Database name ......     (2)
  Schema name   ......     (3)
  Creator       ......     (4)
  Created date  ......     (5)

  Subject table information
    Schema name ......     (6)
    Table name  ......     (7)
    Type        ......     (8)

  Trigger active time ......     (9)
  Trigger event       ......     (10)

  Trigger column information
    No.     Colum name
    1          (11)
    2          (11)
    3          (11)

  Old or new values alias information
    OLD ......    (12)
    NEW ......    (12)

  Trigger action information
    Operation ......    (13)

  Search condition information
            (14)

  Contained table information
    No.   1   Table name ......      (15)
      Schema name ......     (16)
      Type        ......     (17)

  Triggered SQL statemant information
            (18)

  Triggered routine information                                          (*1)
    No.   1   Routine name   ......  (19)
      Schema name ......     (20)
      Type        ......     (21)

  Triggered sequence information                                         (*2)
    No.   1   Sequence name   ...... (22)
      Schema name ......     (23)
```

*1 Output only if the CALL statement is specified as the SQL statement of the triggered operation

*2 Output only if the SQL statement of the triggered oeration contains a sequence

(1) Trigger name

(2) Name of database to which the trigger belongs

(3) Name of schema to which the trigger belongs

(4) Trigger creator

(5) Trigger definition date and time

("day-of-week month day hour:minute:second year"

Example: "Mon Dec 3 17:06:25 2001"

(6) Name of schema to which the trigger-target table belongs

(7) Name of trigger-target table

(8) Type of trigger-target table

BASE: Base table

(9) Trigger operation point

BEFORE: Executes a triggered operation before a trigger operation.

AFTER: Executes a triggered operation after a trigger operation.

(10) Trigger event

INSERT: Inserts a row.

DELETE: Deletes a row.

UPDATE: Updates a row.

(11) Name of column to be updated in the trigger-target table

(12) Correlation name of old and new values in the trigger-target table

OLD: Old value correlation name

NEW: New value correlation name

(13) Execution unit for triggered operation

ROW: By rows

(14) Execution condition of triggered operation

(15) Name of table contained in SQL statement of triggered operation

(16) Name of schema to which table contained in SQL statement of triggered operation belongs

(17) Type of table contained in SQL statement of triggered operation

BASE: Base table

VIEW: View table

GLOBAL TEMPORARY: Temporary table

(18) SQL statement of triggered operation

(19) Routine name contained in the SQL statement of the triggered operation

(20) Name of the schema to which the routine contained in the SQL statement of the triggered operation belongs

(21) Type of routine contained in the SQL statement of the triggered operation

PROCEDURE: Procedure routine

FUNCTION: Function routine

(22) Sequence name contained in the SQL statement of the triggered operation

(23) Name of the schema to which the sequence contained in the SQL statement of the triggered operation belongs

■Definition information output format for a SCOPE specification

[Figure: Definition information output format for a SCOPE specification]

```
No. 1         Scope name ......    (1)

  Database name ......     (2)
  Creator       ......     (3)
  Created date  ......     (4)

  DSI information
    No.     DSI name
    1           (5)
    2           (5)
    3           (5)

  Apply information
    No.     Applied auth-ID
    1           (6)
    2           (6)
    3           (6)
```

(1) Scope name

(2) Name of database to which the scope belongs

(3) User ID of user who defined the scope

(4) Scope definition date and time
("day-of-week month day hour:minute:second year")

(5) Name of DSI limited by scope

(6) User ID of user to which scope if applied

■Definition information output format of a SEQUENCE specification

[Figure: Definition information output format of a SEQUENCE specification]

```
No. 1          Sequence name ......        (1)

  Database name ......       (2)
  Schema name   ......       (3)
  Creator       ......       (4)
  Created date  ......       (5)
  Increment     ......       (6)
  Start number  ......       (7)
  Max Value     ......       (8)
  Min Value     ......       (9)
  Cycle         ......       (10)
  Cache         ......       (11) , (12)
  Order         ......       (13)
```

(1) Sequence name

(2) Name of the database to which the sequence belongs

(3) Name of the schema to which the sequence belongs

(4) Sequence creator

(5) Sequence definition date and time

("day-of-week month day hours:minutes:seconds calendar-year")

Example: "Mon Apr 1 17:06:25 2002"

(6) Sequence number increment interval

(7) Initial sequence number

(8) Maximum sequence number

(9) Minimum sequence number

(10) Whether to use sequence numbers cyclically

YES: The CYCLE clause is specified.

NO: The CYCLE clause is not specified.

(11) Number of memory spaces allocated to sequence numbers

(12) Number of memory spaces retained with sequence numbers

(13) Whether to ensure the sequence between clusters

YES: The ORDER clause is specified.

NO: The ORDER clause is not specified.

■Definition information output format of a USER specification

[Figure: Definition information output format of a USER specification]

```
NO.1 User name      ..... (1)

   Type               ..... (2)

   Manage             ..... (3)
   Password status    ..... (4)                              (*1)
   Creator            ..... (5)
   Created date       ..... (6)
   Updated date       ..... (7)
   Last login date    ..... (8)
   Login failed time  ..... (9)
   Parameter information
      (10)
      (10)
        :
```

*1 Output only if the user is a database-specific user.

(1) User name

(2) User type

(3) User management type

DBMS: Manages by SymfoWARE/RDB (database-specific user)

OS: Managed by associating the user with an OS login name

(4) Password status

NORMAL: Normal status

LOCK: Disabled status

(5) User creator

(6) User definition date and time

("day-of-week month day hours: minutes:seconds calender-year")

Example: "Mon Apr 1 10:05:30 2002"

(7) Last date and time when the user information was altered

(using the ALTER USER statement)

("day-of-week month day hours:minutes:seconds calendar-year")

(8) Last date and time when the user was connected to a database

("day-of-week month day hours:minutes:seconds calendar-year")

(9) Number of times the user failed to connect to a database

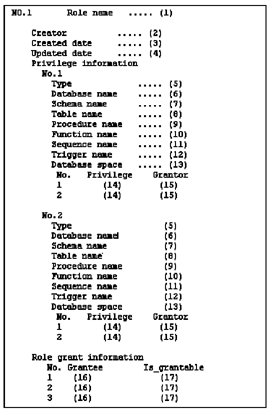
(10) Name and value of a user parameter specified for the user

(output only if a user parameter is specified)

For details, see "Sample user parameter information print results."

■Definition information output format of a ROLE specification

[Figure: Definition information output format of a ROLE specification]



```
NO.1       Role name   ..... (1)

    Creator          ..... (2)
    Created date     ..... (3)
    Updated date     ..... (4)
    Privilege information
      No.1
        Type            ..... (5)
        Database name   ..... (6)
        Schema name     ..... (7)
        Table name      ..... (8)
        Procedure name  ..... (9)
        Function name   ..... (10)
        Sequence name   ..... (11)
        Trigger name    ..... (12)
        Database space  ..... (13)
         No.   Privilege    Grantor
         1        (14)       (15)
         2        (14)       (15)

      No.2
        Type                  (5)
        Database named        (6)
        Schema name           (7)
        Table name            (8)
        Procedure name        (9)
        Function name         (10)
        Sequence name         (11)
        Trigger name          (12)
        Database space        (13)
         No.   Privilege    Grantor
         1        (14)       (15)
         2        (14)       (15)

    Role grant information
       No. Grantee        Is_grantable
       1    (16)              (17)
       2    (16)              (17)
       3    (16)              (17)
```

(1) Role name

(2) User who defined the role

(3) Role definition date and time

("day-of-week month day hours:minutes:seconds calendar-year")

Example: "Mon Apr 1 10:05:30 2002"

(4) Role update date and time

("day-of-week month day hours:minutes:seconds calendar-year")

(5) Resource type

TABLE: Table

PROCEDURE: Procedure routine

FUNCTION: Function routine

SEQUENCE: Sequence

TRIGGER: Trigger

SCHEMA: Schema

DBSPACE: Database space

(6) Database name

(7) Schema name (output if the resource type is TABLE, PROCEDURE, FUNCTION, SEQUENCE, TRIGGER, or SCHEMA)

(8) Table name (output if the resource type is TABLE)

(9) Procedure routine name (output if the resource type is PROCEDURE)

(10) Function routine name (output if the resource type is FUNCTION)

(11) Sequence name (output if the resource type is SEQUENCE)

(12) Trigger name (output if the resource type is TRIGGER)

(13) Database space name (output if the resource type is DBSPACE)

(14) Privilege type

When the resource type is TABLE

SELECT: SELECT privilege

INSERT: INSERT privilege

UPDATE: UPDATE privilege

DELETE: DELETE privilege

TRIGGER: TRIGGER privilege

ALTER: ALTER privilege

INDEX: INDEX privilege

DROP: DROP privilege

When the resource type is PROCEDUER

EXECUTE: EXECUTE privilege

DROP: DROP privilege

When the resource type is FUNCTION

EXECUTE: EXECUTE privilege

DROP: DROP privilege

When the resource type is SEQUENCE

SELECT: SELECT privilege

DROP: DROP privilege

When the resource type is TRIGGER

DROP: DROP privilege

When the resource type is SCHEMA

CREATE: CREATE privilege

DROP: DROP privilege

When the resource type is DBSPACE

ALLOCATE: ALLOCATE privilege

(15) Grantor name

(16) Grantee names

(17) Whether the user has the privilege to grant a role

YES: The user has the grant privilege.

NO: The user does not have the grant privilege.

2.14.4 Printing privilege information

To print privilege information, specify -p in the rdbprt command. Privilege information is printed only for the following specifications:

- SCHEMA
- TABLE
- ROUTINE
- DBSPACE
- TRIGGER
- SEQUENCE

Figure: Sample privilege information printout shows a sample command specification and a sample printout (SCHEMA specification). For more information on how to specify the rdbprt command, refer to the man command (under UNIX) or the SymfoWARE/RDB online manual (under Windows NT).

[Figure: Sample privilege information printout]

```
rdbprt -d STOCKMN_DB -m DEF -f -p print.info
```

```
No. 1         Schema name ......     (1)

  Database name ......     (2)
  Creator       ......     (3)
  Created date  ......     (4)

  Privilege information
    Privilege ...... CREATE
    No.   1   Grantee ......     (5)
      Grantor   ......     (6)
      Grantable ......     (7)
~                                                                          ~
  Table information
    No.       Table name          Type
    1             (8)              (9)
    2             (8)              (9)
    3             (8)              (9)

  Routine information
    No.     Routine name        Type
    1         (10)              (11)
    2         (10)              (11)
    3         (10)              (11)
~                                                                          ~
```

(1) Schema name

(2) Name of database to which the schema belongs

(3) Creator of schema

(4) Schema definition date and time

("day-of-week month day hours:minutes:seconds calendar-year")

Example: "Mon Apr 1 17:06:25 2002"

(5) Privilege grantee (An asterisk (*) is displayed if the privilege was granted to all authorization identifiers. The privilege grantee is PUBLIC.)

(6) Privilege grantor

(7) Whether the privilege was granted with the "WITH GRANT OPTION" specification

YES: "WITH GRANT OPTION" specified

NO: "WITH GRANT OPTION" not specified

(8) Name of table belonging to the schema

(9) Table type

BASE: Base table

VIEW: View

GLOBAL TEMPORARY: Temporary table

(10) Name of routine belonging to the schema

(11) Routine type

PROCEDURE: Procedure routine

FUNCTION: Function routine

Chapter 3 Database Definition Alteration and Deletion

A database can be used after it has been created. To use the database, create an application program. For information about how to use an application program to process a database, refer to the RDB User's Guide: Application Programs Development.

After a database has been created, the user may need to add data items to the designed database. Alternatively, data items may become unnecessary.

This chapter explains how to alter and delete a database definition. The explanations are given in the following order:

3.1 Altering a Database Definition

3.2 Deleting a Database

3.1 Altering a Database Definition

The user alters a database definition by performing the following operations.

See

For more information about the SQL statements described in this chapter, refer to the following manuals:

- SQL Reference Guide
- SQL Beginner's Guide

■Altering the logical structure definition:

- Adding a schema definition (CREATE SCHEMA statement).
- Deleting a schema definition (DROP SCHEMA statement).
- Adding a sequence definition (CREATE SEQUENCE statement)
- Deleting a sequence definition (DROP SEQUENCE statement)
- Adding a table definition (CREATE TABLE statement).
- Deleting a table definition (DROP TABLE statement).
- Altering a table definition (ALTER TABLE statement).
- Adding a view definition (CREATE VIEW statement).
- Deleting a view definition (DROP VIEW statement).
- Adding a trigger definition (CREATE TRIGGER statement).
- Deleting a trigger definition (DROP TRIGGER statement).
- Adding a procedure routine definition (CREATE PROCEDURE statement).
- Deleting a procedure routine definition (DROP PROCEDURE statement).
- Adding a function routine definition (CREATE FUNCTION statement)
- Deleting a function routine definition (DROP FUNCTION statement)
- Altering a comment definition.
- Swapping a table (SWAP TABLE statement).

■Altering the storage structure definition:

- Adding a table data structure organization (DSO) definition (CREATE DSO statement).
- Deleting a table DSO definition (DROP DSO statement).
- Adding a table data structure instance (DSI) definition (CREATE DSI statement).
- Deleting a table DSI definition (DROP DSI statement).
- Adding an index DSO definition (CREATE DSO statement).
- Deleting an index DSO definition (DROP DSO statement).
- Altering the split value of a DSI definition (ALTER DSI statement).
- Addinging a scope definition (CREATE SCOPE statement).
- Applying a scope definition (APPLY SCOPE statement).
- Releasing a scope definition (RELEASE SCOPE statement).
- Deleting a scope definition (DROP SCOPE statement).

■Defining optimization information for added definition

If a table and index is added, define the optimization information for them.

- Defining optimization information (SET STATISTICS statement)

■Altering privileges

- Adding privilege information (GRANT statement)
- Deleting privilege information (REVOKE statement)
- Adding a role definition (CREATE ROLE statement)
    - Adding a privilege to a role (GRANT statement)
    - Granting the role privileges to a user (GRANT statement)
- Altering role privilege information (GRANT statement)
- Deleting a role definition (DROP ROLE statement)
- Removing a role privilege (REVOKE statement)
    - Deleting a privilege for a table from a role
    - Removing the role privileges from a user

Note that when a database definition is deleted or altered, the definition information is deleted or altered. Moreover, the data itself (table or index) is deleted or altered at the same time. If the deletion or alteration of a database definition is specified by mistake, important data may also be deleted. Always use care when deleting or altering a database definition.

The following database is used in the examples in this section:

Database:
    STOCKMN_DB
Schema:
    STOCKS
Sequence:
    SEQUENCE1
Tables:
    STOCK and ORDER
View:
    MASS_STOCK
Table DSOs:
    STOCK_DSO and ORDER_DSO
Table DSIs:
    STOCK_DSI and WEST_ORDER_DSI
Index DSO:
    PRODUCT_IXDSO
Index DSI:
    PRODUCT_IXDSI

Because of dependencies between definitions, the user must follow an alteration sequence that conforms to those dependencies when altering a database definition. For information about basic alteration sequences, see Appendix B "Sequential Relationships Among Definition Changes."

3.1.1 Altering a logical structure defining

■Adding a schema definition (CREATE SCHEMA statement)

A new schema definition can be added to an existing database definition. To add a schema definition, specify the CREATE SCHEMA statement in a similar manner as when defining a schema. For details about how to specify the CREATE SCHEMA statement, see 2.6 "Defining a Logical Structure."

Example:

Adding a schema belonging to STOCKMN_DB.

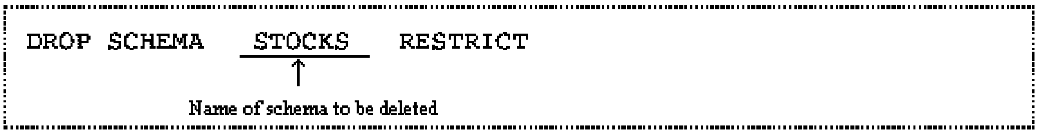
```
CREATE  SCHEMA  STOCKS
                  ↑
              Schema name
```

■Deleting a schema definition (DROP SCHEMA statement)

To delete a schema definition, use the DROP SCHEMA statement. A specification example follows.

Example:

Delete a schema belonging to STOCKMN_DB.



```
DROP SCHEMA STOCKS RESTRICT
```

When an attempt is made to delete a schema definition, if any of the following definitions subordinate to that schema exist, the schema cannot be deleted. The user must delete these definitions before deleting the schema definition:

- Sequence
- Table
- View
- Temporary table
- Trigger
- Index
- Storage structure
  - Table DSO
  - Index DSO
  - Table DSI
  - Index DSI
- Procedure
- Function routine

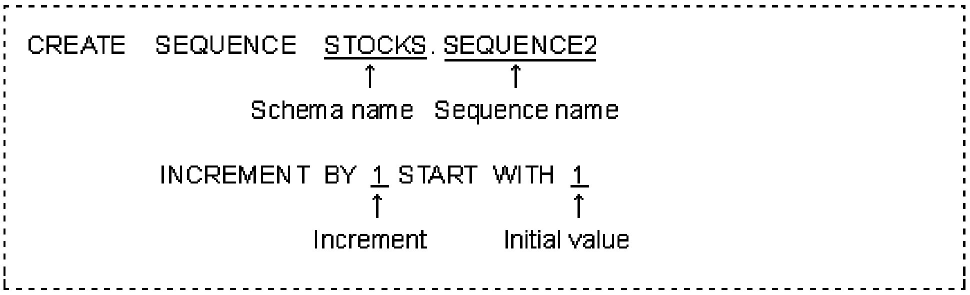
However, if the user specifies CASCADE, then even if the preceding definitions exist, all related definitions are unconditionally deleted.

■Adding a sequence definition (CREATE SEQUENCE statement)

To add a sequence definition to a schema, use the CREATE SEQUENCE statement. A specification example follows. For details on how to specify the CREATE SEQUENCE statement, see 2.6 "Defining a Logical Structure."

Example:

Adds a definition of SEQUENCE2 to schema STOCKS



```
CREATE SEQUENCE STOCKS.SEQUENCE2
        INCREMENT BY 1 START WITH 1
```

■Deleting a sequence definition (DROP SEQUENCE statement)

To delete a sequence definition, use the DROP SEQUENCE statement. If a base table, view table, procedure routine, function routine, or trigger references a sequence, the sequence definition cannot be deleted. Delete the definition of the base table, view table, procedure routine, function routine, or trigger that references the sequence before deleting the sequence definition. A specification example follows.

Example:

Deletes SEQUENCE1.

```
DROP  SEQUENCE  STOCKS. SEQUENCE1
                  ↑         ↑
              Schema name  Sequence name
```

■Adding a table definition (CREATE TABLE statement)

To add a table definition to a schema, use the CREATE TABLE statement. A specification example follows. For details about how to specify the CREATE TABLE statement, see 2.6 "Defining a Logical Structure."

Example:

Add a definition of the PRODUCT table to the schema named STOCKS.

```
CREATE TABLE
        STOCKS.PRODUCT  (NO SMALLINT NOT NULL, NAME CHAR(25), ...)
          ↑       ↑
    Schema name Table name
            ↑
  Table name specification of base table to be added
```

■Deleting a table definition (DROP TABLE statement)

To delete a table definition, use the DROP TABLE statement. If a view table, procedure routine, function routine, or trigger references a base table, the table definition cannot be deleted. Delete the definition of the view table, procedure routine, function routine, or trigger that references the base table before deleting the table definition. To delete the view, procedure routine, function routine, or trigger definition that references a base table together with the table definition, specify CASCADE. Similarly, the CASCADE specification is required to delete any storage structure definition of a base table together with the table definition.

When a table definition is deleted by the DROP TABLE statement, the base table database data is also deleted at the same time. If another table name is specified by mistake in the table definition, the definition information will be lost. Moreover, the important data will be lost as well. Be especially careful when using the DROP TABLE statement.

An example of deleting a table definition follows.

Example:

Delete the STOCK table definition information and STOCK table.

```
DROP TABLE         STOCKS.STOCK        CASCADE
                     ↑      ↑             ↑
               Schema name Table name  CASCADE specification
                        ↑
       Table name specification of base table to be deleted
```

■Altering a table definition (ALTER TABLE statement)

To alter a table definition, use the ALTER TABLE statement. The ALTER TABLE statement can be used to make the following changes:

· Add a column definition.
· Delete a column definition.

◆Adding a column definition

To add a column to a base table, specify addition of a column definition in the ALTER TABLE statement. Only one column can be added at the end of the existing columns per table definition alter statement. For the SEQUENTIAL structure, data corresponding to one row in a table may exceed one page after a BLOB-type column is added.

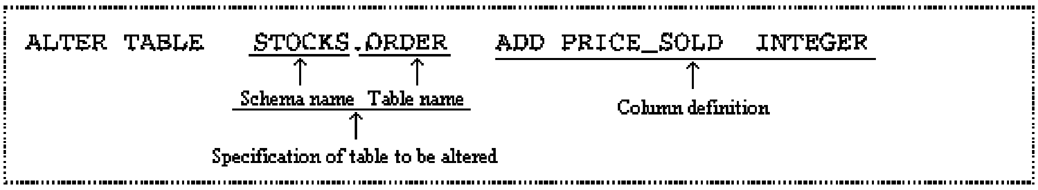
Only NOT NULL can be specified as a constraint for the column to be added. If NOT NULL is specified, the DEFAULT clause must be specified in the column definition.

DEFAULT values are set for existing data.

A specification example for adding a column definition follows.

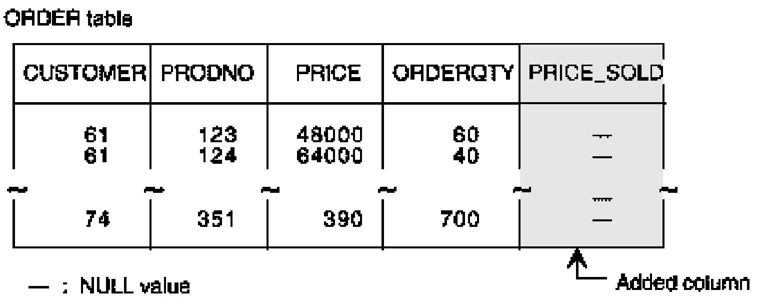
Example:

Add a PRICE_SOLD column to the ORDER table. Figure: Adding a column to a table shows the result.



```
ALTER TABLE   STOCKS.ORDER   ADD PRICE_SOLD  INTEGER
```

[Figure: Adding a column to a table]



ORDER table

CUSTOMER	PRODNO	PRICE	ORDERQTY	PRICE_SOLD
61	123	48000	60	—
61	124	64000	40	—
~	~	~	~	~
74	351	390	700	—

Added column

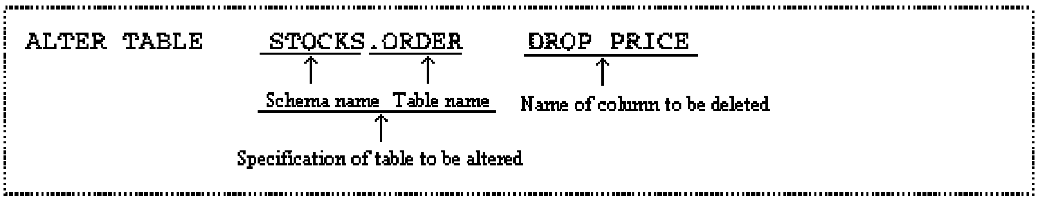
— : NULL value

◆Deleting a column definition

To delete a column from a base table, specify the deletion of a column definition in the ALTER TABLE statement. Only one column can be deleted by one table alteration statement. A column cannot be deleted if it is referenced by a view definition. If an index DSO definition for the column exists, the column cannot be deleted. The index DSO definition must be deleted first. However, if a unique constraint for the column exists, the column cannot be deleted even if the index DSO definition is deleted. The number of columns in a table cannot be reduced to zero by deleting a column definition. To delete all columns, use the DROP TABLE definition. A specification example for deleting a column definition follows.

Example:

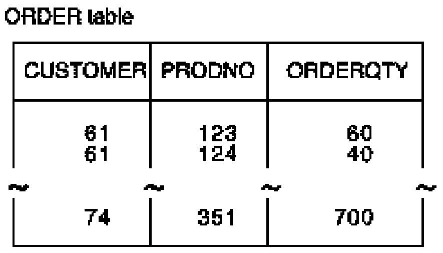
Delete the PRICE column from the ORDER table. Figure: Deleting a column from a table shows the result.



```
ALTER TABLE   STOCKS.ORDER   DROP PRICE
```

Schema name
Table name
Name of column to be deleted
Specification of table to be altered

[Figure: Deleting a column from a table]



ORDER table

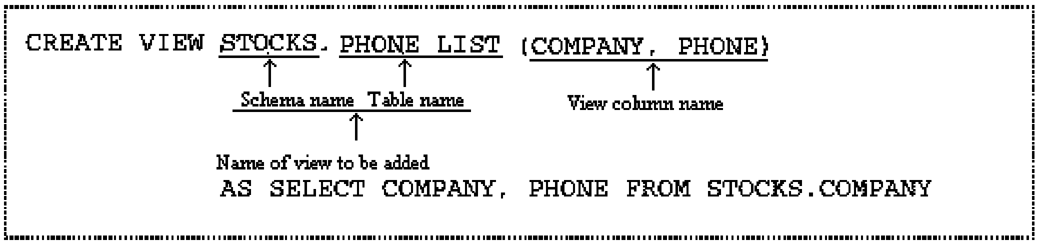
CUSTOMER	PRODNO	ORDERQTY
61	123	60
61	124	40
~	~	~
74	351	700

■Adding a view definition (CREATE VIEW statement)

To add a view definition to a previously defined schema, use a CREATE VIEW statement. A view definition cannot be altered. To change a view definition, first delete the view definition and then add a new view definition. An example of adding a view definition follows. For details about how to specify the CREATE VIEW statement, see "2.6 Defining a Local Structure."

Example:

Add a PHONE_LIST view to the schema named STOCKS.



```
CREATE VIEW STOCKS.PHONE_LIST (COMPANY, PHONE)
            AS SELECT COMPANY, PHONE FROM STOCKS.COMPANY
```

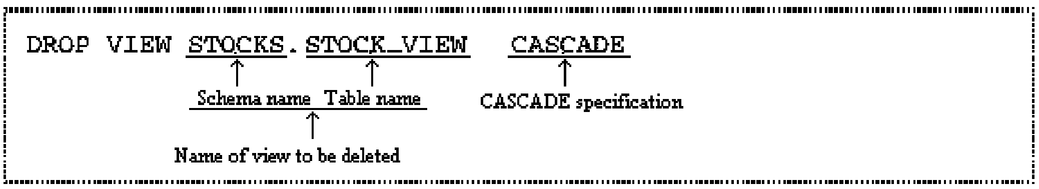
■Deleting a view definition (DROP VIEW statement)

To delete a view definition from a previously defined schema, use the DROP VIEW statement. If a view table to be deleted is referenced in another view table, a procedure routine, a function routine, or a trigger definition, the view definition cannot be deleted. Delete the view table, procedure routine, function routine, or trigger definition that references the view table to be deleted before deleting the view definition. To delete the view table, procedure routine, function routine, or trigger definition that references a view table together with the view table definition, specify CASCADE.

An example of deleting a view definition follows.

Example:

Delete the definition of the view named STOCK_VIEW.



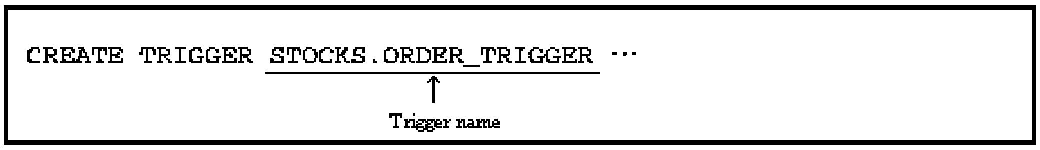
```
DROP VIEW STOCKS.STOCK_VIEW   CASCADE
```

■Adding a trigger definition (CREATE TRIGGER statement)

To add a trigger definition, use the CREATE TRIGGER statement. For more information on how to specify the CREATE TRIGGER statement, see 2.6 "Defining a Logical Structure."

Example:

Define the trigger ORDER_TRIGGER.



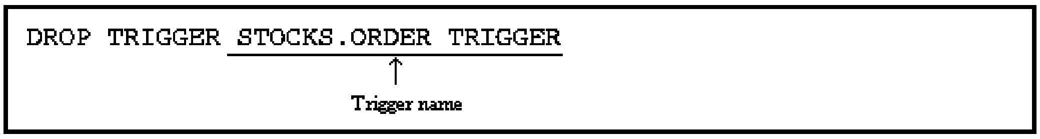
```
CREATE TRIGGER STOCKS.ORDER_TRIGGER …
```

■Deleting a trigger definition (DROP TRIGGER statement)

To delete a trigger definition, use the DROP TRIGGER statement.

Example:

Delete trigger ORDER_TRIGGER.



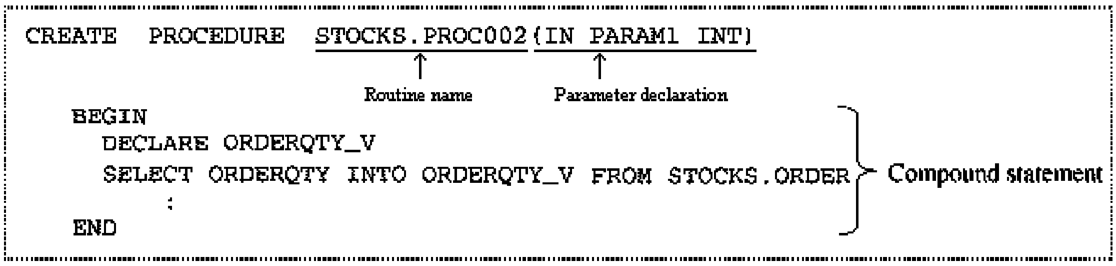
```
DROP TRIGGER STOCKS.ORDER TRIGGER
```

■Adding a procedure routine definition (CREATE PROCEDURE statement)

To add a procedure routine, use the CREATE PROCEDURE statement. For details about how to specify the CREATE PROCEDURE statement, see 2.6 "Defining a Logical Structure."

Example:

Add PROC002 to the STOCKS schema.



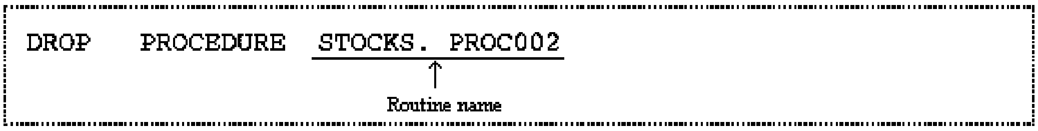
```
CREATE  PROCEDURE  STOCKS.PROC002(IN PARAM1 INT)

   BEGIN
     DECLARE ORDERQTY_V
     SELECT ORDERQTY INTO ORDERQTY_V FROM STOCKS.ORDER
         :
   END
```

■Deleting a procedure routine definition (DROP PROCEDURE statement)

To delete a procedure routine, use the DROP PROCEDURE statement. If the schema has another procedure routine that calls the procedure routine to be deleted, the procedure routine cannot be deleted. To delete all related procedure routines, specify CASCADE. For details about how to specify the DROP PROCEDURE statement, see 2.6 "Defining a Logical Structure."

Example:

Delete PROC002 from the STOCKS schema.



```
DROP   PROCEDURE  STOCKS. PROC002
```

■Adding a function routine definition (CREATE FUNCTION statement)

To add a function routine, use a function routine definition statement. For details on how to specify the function routine definition statement, see 2.6 "Defining a Logical Structure."

Example:

Defines function routine USER002 to schema STOCKS.

```
CREATE   FUNCTION   STOCKS.USER002 ( IN INTEGER, IN INTEGER )
                       ↑        ↑              ↑
                Schema name  Routine name   Parameter declaration

           RETURNS INTEGER   LANGUAGE C
                      ↑
                 Return data type

           NAME 'ABCDEFG'  LIBRARY '/usr/local/lib/libuser1.so'
                    ↑                          ↑
                Symbol name                  Library
```

■Deleting a function routine definition (DROP FUNCTION statement)

To delete a function routine, use a function routine deletion statement. If a function routine to be deleted is specified in an SQL statement related to a procedure routine or trigger, the function routine cannot be deleted. However, specifying CASCADE deletes all related definitions.

Example:

Deletes USER001 from schema STOCKS.

```
DROP   FUNCTION   STOCKS. USER001
                     ↑        ↑
               Schema name  Routine name
```

■Changing a comment definition

Comment definitions for tables and columns can be changed with the ALTER TABLE statement. An example follows.

Example:

Change the table comment definition.

```
ALTER TABLE   STOCKS.STOCK

              COMMENT 'PRODUCT NAME, STOCK QUANTITY, STORAGE WAREHOUSE, AND SALES PRICE TABLE'
                                                  ↑
                                    Comment definition to be changed
```

Example:

Change the column comment definition.

```
ALTER TABLE   STOCKS.STOCK ...

              MODIFY ITMNO COMMENT 'PRODUCT CODE NUMBER'
                                 ↑
                    Comment definition to be changed
```

■Swapping a table (SWAP TABLE statement)

The table name exchange in table swapping exchanges the table's relationship with DSO. Use the SWAP TABLE statement for table swapping. This produces the following effects:

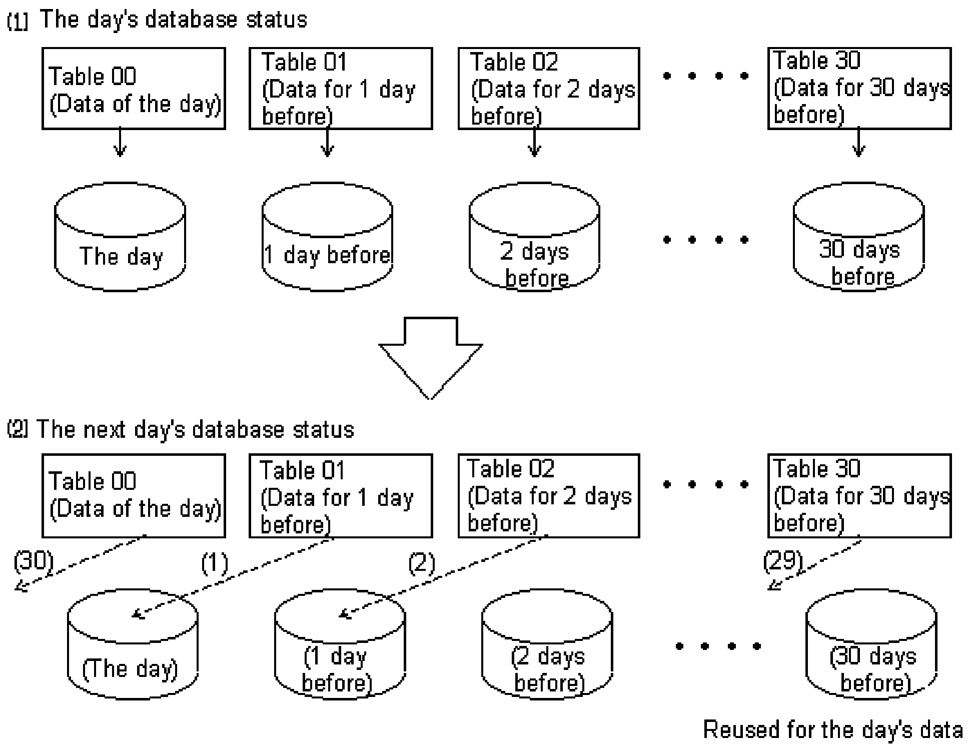
- Data can be transfered in a short time.
- An application program need not be changed.
- Previous views and routines can be used without changing their definitions.

An example of table swapping follows.

Example:

This example shows table swapping when the data for each day is shifted to the day before in a system managing data by days. For a new day, data is initialized.

```
SWAP TABLE  Table 00  Table 01  (1)
SWAP TABLE  Table 00  Table 02  (2)
                :
                :
                :
SWAP TABLE  Table 00  Table 29  (29)
SWAP TABLE  Table 00  Table 30  (30)
```


[1] The day's database status
Table 00 (Data of the day)
Table 01 (Data for 1 day before)
Table 02 (Data for 2 days before)
Table 30 (Data for 30 days before)
The day
1 day before
2 days before
30 days before
[2] The next day's database status
Table 00 (Data of the day)
Table 01 (Data for 1 day before)
Table 02 (Data for 2 days before)
Table 30 (Data for 30 days before)
(30)
(1)
(2)
(29)
(The day)
(1 day before)
(2 days before)
(30 days before)
Reused for the day's data

3.1.2 Changing a definition of a storage structure

■Adding a table DSO definition (CREATE DSO statement)

To add a table DSO definition after adding the table definition, use the CREATE DSO statement. An example of adding a table DSO definition follows. For details about how to specify the CREATE DSO statement, see 2.7 "Defining a Storage Structure."

Example:

Add STOCK_DSO as the DSO of the STOCK table.

```
CREATE DSO  STOCK_DSO FROM STOCKS.STOCK
                ↑                ↑
   Name of DSO to be added     Table name
            TYPE SEQUENTIAL  (PAGESIZE(4),  ORDER(0))
                                  ↑
                            Data structure
```

■Deleting a table DSO definition (DROP DSO statement)

To delete a table DSO definition from a base table, use the DROP DSO statement. If an index DSO or a table DSI has been defined for the DSO to be deleted, the relevant DSO definition cannot be deleted. All related index DSO definitions or table DSI definitions must be deleted in advance.

However, if the user specifies CASCADE, all DSIs related to the DSO to be deleted are deleted.

An example of deleting a table DSO definition follows.

Example:

Delete all table DSOs belonging to STOCKMN_DB.

```
DROP DSO   STOCK_DSO
               ↑
   Name of DSO to be deleted
DROP DSO   ORDER_DSO
               ↑
   Name of DSO to be deleted
```

■Adding an index DSO definition (CREATE DSO statement)

To add an index DSO definition for a base table for which the table DSO has already been defined, use the CREATE DSO statement. An example of adding an index DSO definition follows. For details about how to specify the CREATE DSO statement, see 2.7 "Defining a Storage Structure."

Example:

Add STOCKNO_IXDSO as the index DSO for the WHCODE column of the STOCK table.

```
CREATE DSO  STOCKNO_IXDSO  INDEX ON   STOCKS.STOCK   (WHCODE)
                 ↑                          ↑            ↑
   Name of index DSO to be added  Table name qualified by schema name  Column name
            TYPE BTREE (PAGESIZE1(16), PAGESIZE2(1))
```

■Deleting an index DSO definition (DROP DSO statement)

To delete an index DSO definition from a base table, use the DROP DSO statement.

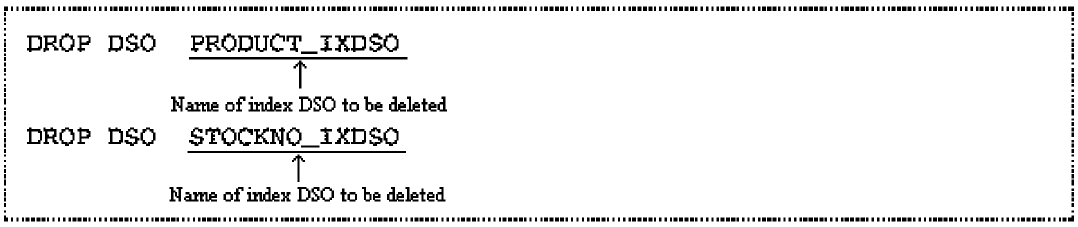
If an index DSI has been defined, the relevant index DSO definition cannot be deleted. All related index DSI definitions must be deleted in advance.

However, when CASCADE is specified, if the DSO to be deleted is for a base table, all related index DSOs and related DSIs are deleted.

An example of deleting an index DSO definition follows.

Example:

Delete all index DSOs belonging to STOCKMN_DB.



```
DROP DSO   PRODUCT_IXDSO
DROP DSO   STOCKNO_IXDSO
```

Name of index DSO to be deleted
Name of index DSO to be deleted

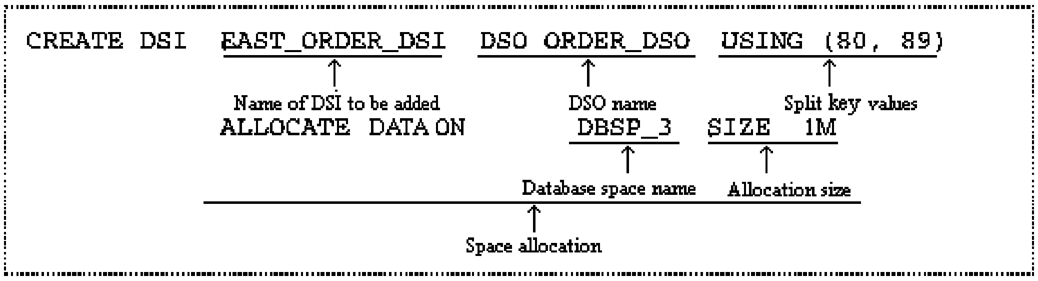
■Adding a table DSI definition (CREATE DSI statement)

To add a table DSI definition for a base table for which the DSO has been defined, use the CREATE DSI statement. Specify each item of the CREATE DSI statement in a similar manner as when defining the storage structure.

An example of adding a table DSI definition follows. For details about how to specify the CREATE DSI statement, see 2.7 "Defining a Storage Structure".

Example:

Add EAST_ORDER_DSI to STOCKMN_DB. EAST_ORDER_DSI is subordinate to ORDER_DSO and is defined on the database space named DBSP_3.



```
CREATE DSI  EAST_ORDER_DSI  DSO ORDER_DSO  USING (80, 89)
            ALLOCATE  DATA ON   DBSP_3  SIZE  1M
```

■Deleting a table DSI definition (DROP DSI statement)

To delete a DSI definition from a base table, use the DROP DSI statement. If the DSI to be deleted is a table DSI, and an index DSI related to it exists, the relevant DSI definition cannot be deleted. All DSI definitions of related index DSIs must be deleted in advance.

However, when CASCADE is specified, if the DSI to be deleted is a table DSI, all related index DSIs are deleted.

An example of deleting a DSI definition follows.

Example:

Delete all DSIs belonging to STOCKMN_DB.

```
DROP DSI  PRODUCT_IXDSI
                ↑
          Name of DSI to be deleted

DROP DSI  STOCK_DSI
              ↑
        Name of DSI to be deleted

DROP DSI  WEST_ORDER_DSI
                ↑
           Name of DSI to be deleted

DROP DSI  EAST_ORDER_DSI
                ↑
          Name of DSI to be deleted
```

■Adding an index DSI definition (CREATE DSI statement)

To add an index DSI definition, use the CREATE DSI statement.

```
CREATE  DSI  STOCKNO_IX DSI  INDEX DSO  STOCKNO_IX DSO
                   ↑                         ↑
             Index DSI name            Index DSO name

          BASE STOCK_DSI
                 ↑
          Table DSI name

          ALLOCATE INDEX ON DBSP_4 SIZE 1M,     ← Space allocation
                   BASE  ON DBSP_4 SIZE 100M    ← Space allocation
```

■Changing a split key value of a DSI definition (ALTER DSI statement)

To change a split key value of a table DSI definition, use the ALTER DSI statement. If the table storage structure is SEQUENTIAL or RANDOM, the split key value can be changed. An example of changing a split key value follows.

◆Integrating DSIs

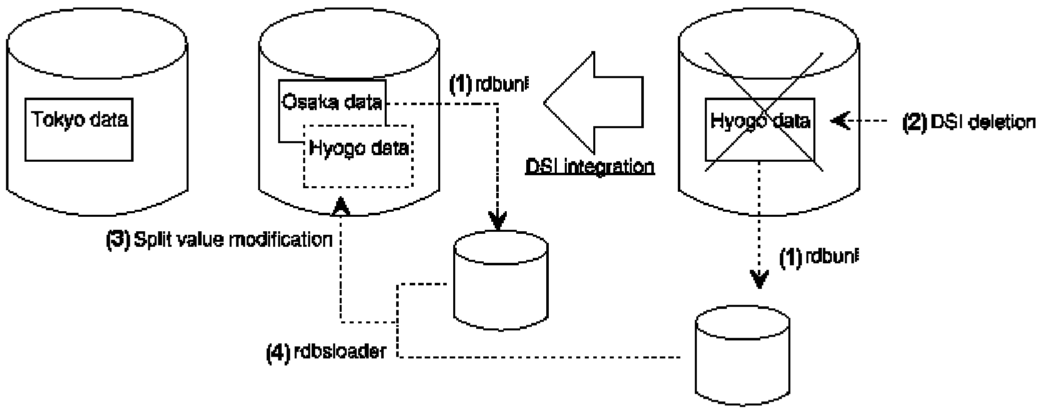
An example of integrating Hyogo data into Osaka data follows.

<Before modification>

```
CREATE DSO DSO01    FROM SCHO1. TABLE01
           TYPE SEQUENTIAL(PAGESIZE(4), ORDER(0))
           WHERE(LOCATION) = (?)

CREATE DSI TOKYO_DSI DSO DSO01
                     USING('TOKYO') ALLOCATE DATA ON ...
CREATE DSI OSAKA_DSI DSO DSO01
                     USING('OSAKA') ALLOCATE DATA ON ...
CREATE DSI HYOGO_DSI DSO DSO01
                     USING('HYOGO') ALLOCATE DATA ON ...
          :
```

<DSI modification example>

```
DROP   DSI  HYOGO_DSI

ALTER  DSI  OSAKA_DSI  ALTER    USING('OSAKA'), ('HYOGO')
                                          ↑
                                 Split value to be changed
```



(1) The Osaka data and Hyogo data is output to a file.-----------------(rdbunl command)

(2) The Hyogo data DSI is deleted. --(rdbddlex command)

(3) The split value is changed by ALTER DSI.---------------------------(rdbddlex command)

(4) The data output to the file ((1)) is loaded at the same time. ------(rdbsloader command)

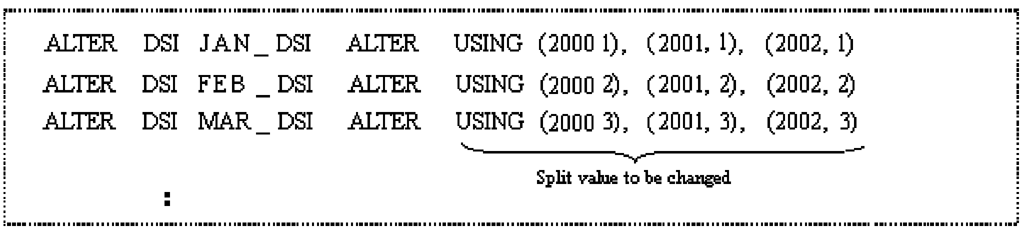
◆Changing the DSI storage range

An example follows in which the number of years of storage is changed for sales data having a three-year storage period.

<Before modification>

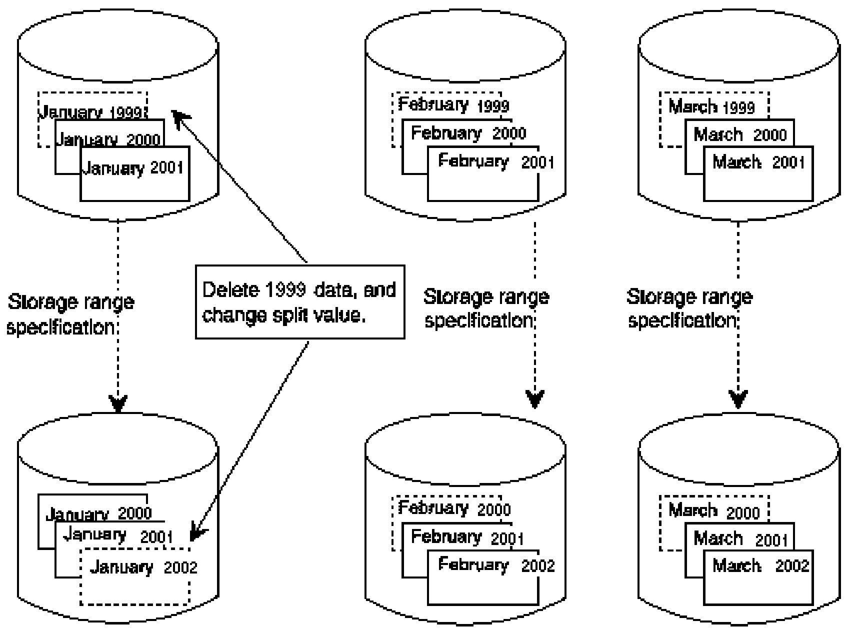
```
CREATE DSO SALES_DSO  FROM  STOCKS.SALES
           TYPE SEQUENTIAL (PAGESIZE(4), ORDER(0))
           WHERE (YEAR, MONTH) = (?, ?)

CREATE DSI JAN_DSI DSO SALES_DSO
                   USING (1999, 1),(2000, 1),(2001, 1) ...
CREATE DSI FEB_DSI DSO SALES_DSO
                   USING (1999, 2),(2000, 2),(2001, 2) ...
CREATE DSI MAR_DSI DSO SALES_DSO
                   USING (1999, 3),(2000, 3),(2001, 3) ...
       :
```

<Example of DSI modification specification>



```
ALTER  DSI  JAN_DSI  ALTER  USING (2000 1), (2001, 1), (2002, 1)
ALTER  DSI  FEB_DSI  ALTER  USING (2000 2), (2001, 2), (2002, 2)
ALTER  DSI  MAR_DSI  ALTER  USING (2000 3), (2001, 3), (2002, 3)
           :
```

Split value to be changed

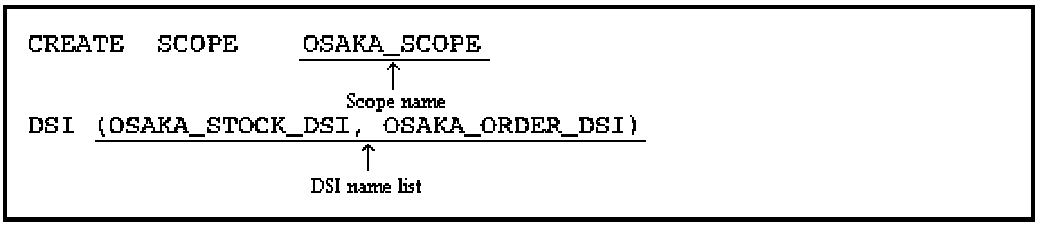


■Adding a scope definition (CREATE SCOPE statement)

To add a scope definition, use the CREATE SCOPE statement. An example of applying a scope definition follows. For more information on how to specify the CREATE SCOPE statement, see 2.7.7 "Scope definition."

Example:

Add scope OSAKA_SCOPE.



```
CREATE  SCOPE   OSAKA_SCOPE
DSI (OSAKA_STOCK_DSI, OSAKA_ORDER_DSI)
```

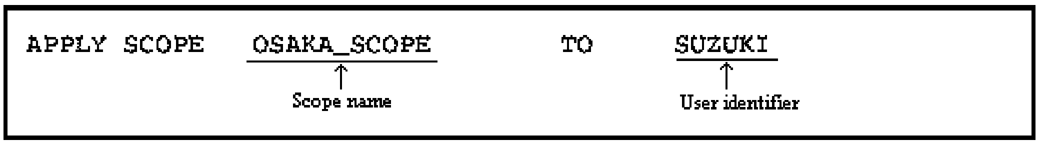
Scope name
DSI name list

■Applying a scope definition (APPLY SCOPE statement)

To apply an added scope definition, use the APPLY SCOPE statement. An example of applying a scope definition follows. For more information on how to specify the APPLY SCOPE statement, see 2.8.1 "Scope definition application."

Example:

Apply scope OSAKA_SCOPE to user SUZUKI.



```
APPLY SCOPE  OSAKA_SCOPE    TO    SUZUKI
```

■Releasing a scope definition (RELEASE SCOPE statement)

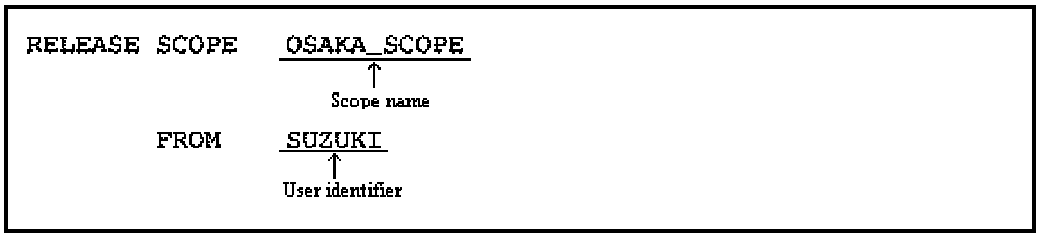
To release a scope definition, use the RELEASE SCOPE statement. This statement releases a scope definition that was applied to a user by the APPLY SCOPE statement.

The statements "RELEASE SCOPE" and "APPLY SCOPE" must be executed by the same user.

An example of using the RELEASE SCOPE statement follows.

Example:

Release the scope "OSAKA_SCOPE" that is currently applied to user "SUZUKI."



```
RELEASE SCOPE  OSAKA_SCOPE

        FROM   SUZUKI
```

■Deleting a user scope (DROP SCOPE statement)

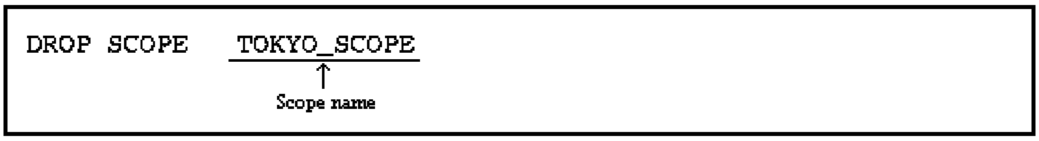
To delete a scope definition, use the DROP SCOPE statement. This statement deletes a scope definition that was applied to a user by the APPLY SCOPE statement.

The statements "DROP SCOPE" and "CREATE SCOPE" must be executed by the same user.

An example of using the DROP SCOPE statement follows.

Example:

Delete scope TOKYO_SCOPE.



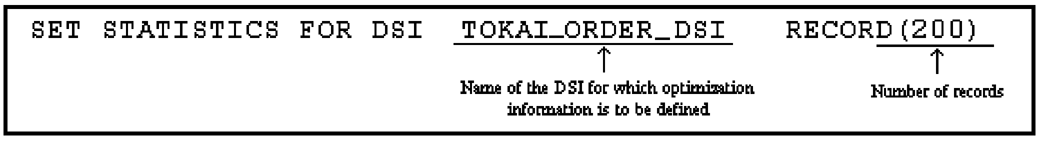
```
DROP SCOPE  TOKYO_SCOPE
```

3.1.3 Defining optimization information for added definitions (SET STATISTICS statement)

If a table and index are added, use the SET STATISTSICS statement to define optimization information for them. An example of the SET STATISTICS statement follows.

Example:

Define optimization information about an added TOKAI_ORDER _DSI.



```
SET STATISTICS FOR DSI  TOKAI_ORDER_DSI   RECORD(200)
```

Name of the DSI for which optimization information is to be defined
Number of records

■Precaution when altering a database definition

After a database definition is altered, use the rdbprdic command to confirm the RDB dictionary utilization rate and estimate the expansion point. This precaution ensures sufficient RDB dictionary space.

For information about estimating the RDB dictionary expansion point, refer to "RDB Operations Guide."

3.1.4 Altering privilege information

■Adding privilege information (GRANT statement)

To add privilege information, use the GRANT statement.

Example:

Adds the deletion privilege.

```
GRANT   DELETE    ON   STOCKS. STOCKTABLE  TO  SATO, SUZUKI, TANAKA
          ↑                    ↑                       ↑
       Privilege          Object name          Privilege grantees
```

■Deleting privilege information (REVOKE statement)

To delete privilege information, use the REVOKE statement.

Example:

Deletes deletion privilege.

```
REVOKE   DELETE   ON   STOCKS. STOCKTABLE   FROM  SATO, SUZUKI, TANAKA
            ↑                    ↑                        ↑
         Privilege          Object name           Privilege grantees
```

If a privilege is deleted with CASCADE specified, the base table, view table, temporary table, procedure routine, and trigger defined by the grantees using the privilege are deleted. The following example assumes that SUZUKI created the PHONE view table using the COMPANY table.

```
CREATE VIEW   STOCKS. PHONE (COMPANY_NAME, TELEPHONE NUMBER )
              AS SELECT   COMPANY_NAME, TELEPHONE NUMBER   FROM   STOCKS.COMPANY
```

When the SELECT privilege granted to SUZUKI for the COMPANY table is removed, the PHONE view table created by SUZUKI is deleted.

```
REVOKE   SELECT   ON   STOCKS. COMPANY   FROM   SUZUKI   CASCADE
```

■Adding a role definition (CREATE ROLE statement)

To add a role definition, use the CREATE ROLE statement. To specify a privilege to be granted for a table in a role and grant this role privilege to a user, use the GRANT statement.

To add a role, follow the procedure given below.

1. Define a role using the CREATE ROLE statement.
2. Specify the privileges to be granted in the role by using the GRANT statement.
3. Grant the role privileges to a user by using the GRANT statement.

Example:

Adds role STOCKS_A2.

```
CREATE ROLE STOCKS_A2 ;
```

Specify the privileges to be granted in role STOCKS_A2.

```
GRANT  SELECT  ON  STOCKS. STOCKTABLE  TO  ROLE  STOCKS_A2 ;
GRANT  SELECT, INSERT, UPDATE  ON  STOCKS. ORDER
          TO ROLE STOCKS_A2 ;
GRANT  SELECT, UPDATE, INSERT, DELETE  ON  STOCKS. COMPANY
          TO ROLE STOCKS_A2 ;
```

Grant the privileges of role STOCKS_A2 to users.

```
GRANT  STOCKS_A2  TO  SUZUKI, TANAKA, SATO ;
```

■Altering role privilege information (GRANT statement)

To alter the role privileges for a table, use the GRANT statement.

Example:

Adds privileges for the STOCK table to role STOCKS_A2.

```
GRANT  INSERT, UPDATE  ON  STOCKS, STOCKTABLE  TO  ROLE  STOCKS_A2 ;
```

■Deleting a role definition (DROP ROLE statement)

To delete a role definition, use the DROP ROLE statement. When a role is deleted, the privileges to be granted as defined with the GRANT statement in the role are also deleted, and the role privileges granted to users are removed.

Example:

Deletes role STOCKS_A2.

```
DROP  ROLE  STOCKS_A2 ;
```

■Deleting/removing role privileges (REVOKE statement)

To delete a privilege from a role or remove the role privileges from a user, use the REVOKE statement.

Example 1:

Deletes the SELECT privilege for the STOCK table from role STOCKS_A2.

```
REVOKE  SELECT  ON  STOCKS, STOCKTABLE  FROM  ROLE  STOCKS_A2 ;
```

Example 2:

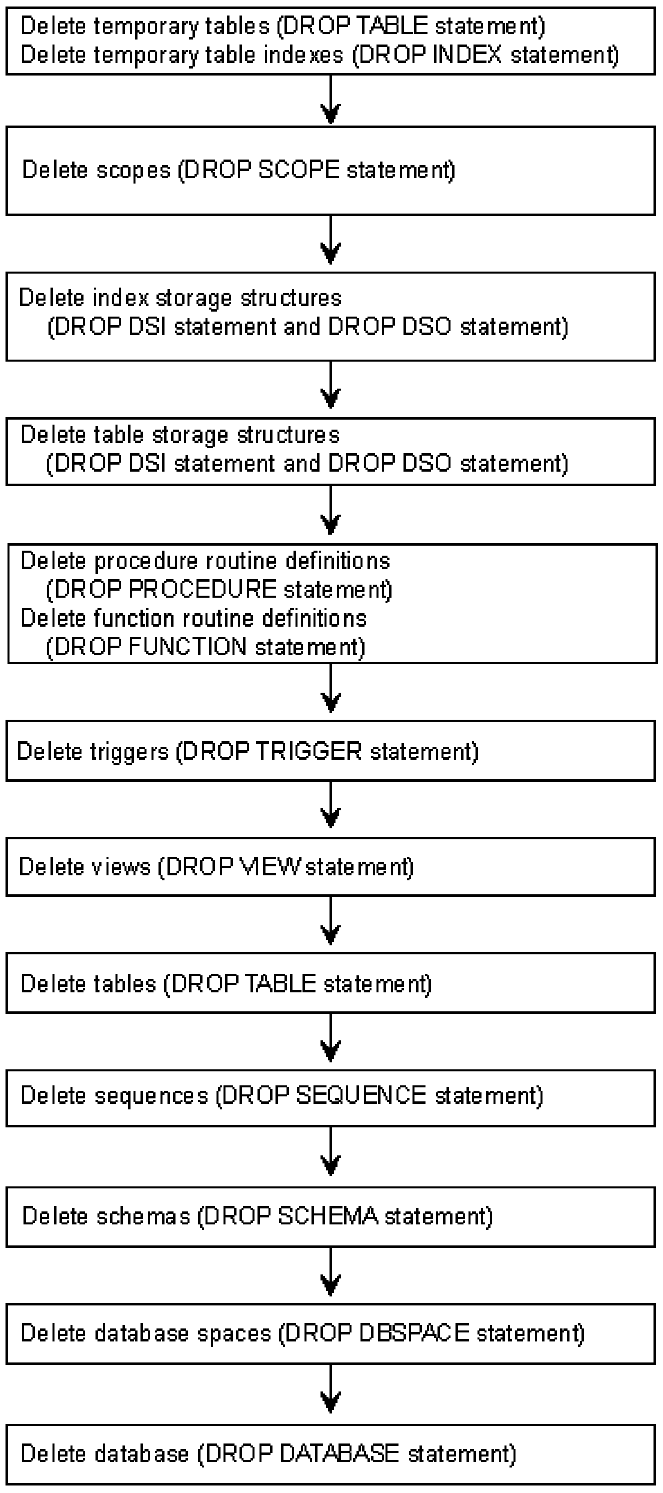
Removes the role privileges from user TANAKA.

```
REVOKE STOCKS_A2 FROM TANAKA ;
```

3.2 Deleting a Database

The user deletes a database definition by performing a sequence of operations. Figure: Database deletion procedure shows this procedure.

[Figure: Database deletion procedure]


Delete temporary tables (DROP TABLE statement)
Delete temporary table indexes (DROP INDEX statement)
Delete scopes (DROP SCOPE statement)
Delete index storage structures (DROP DSI statement and DROP DSO statement)
Delete table storage structures (DROP DSI statement and DROP DSO statement)
Delete procedure routine definitions (DROP PROCEDURE statement)
Delete function routine definitions (DROP FUNCTION statement)
Delete triggers (DROP TRIGGER statement)
Delete views (DROP VIEW statement)
Delete tables (DROP TABLE statement)
Delete sequences (DROP SEQUENCE statement)
Delete schemas (DROP SCHEMA statement)
Delete database spaces (DROP DBSPACE statement)
Delete database (DROP DATABASE statement)

The following database is used in the examples in this section:

Database:
    STOCKMN_DB
Schema:
    STOCKS
Sequence:
    SEQUENCE1

Tables:
  STOCK, ORDER, and STOCK_TEMPORARY TABLE
Table DSO:
  ORDER_DSO
Table DSIs:
  STOCK_DSI and WEST_ORDER_DSI
Index:
  STOCK_TEMPORARY_IX
Index DSO:
  PRODUCT_IXDSO
Index DSI:
  PRODUCT_IXDSI
Database spaces:
  DBSPACE_1, DBSPACE_2
Procedure routine:
  PROC002
Function routine:
  USER001
Trigger:
  ORDER_TRIGGER
Scope:
  TOKYO_SCOPE

■Deleting temporary tables

To delete a temporary table, use the DROP TABLE statement.

Example:

Deletes temporary table STOCK_TEMPORARY.

```
DROP  TABLE   STOCKS. STOCK_TEMPORARY TABLE  CASCADE
                      ---------------------
                                ↑
                            Index name
```

■Deleting temporary table indexes

To delete the index of a temporary table, use the DROP INDEX statement.

Example:

Deletes temporary table index STOCK_TEMPORARY_IX.

```
DROP  INDEX   STOCKS. STOCK_TEMPORARY TABLE .
                      STOCK_TEMPORARY TABLE  I X
                      --------------------------
                                ↑
                            Index name
```

■Deleting scopes

To delete a scope, use the DROP SCOPE statement.

Example:

Delete scope TOKYO_SCOPE.

```
DROP SCOPE    TOKYO_SCOPE
                   ↑
               Scope name
```

■Deleting index storage structures

To delete index storage structures, use the DROP DSI and DROP DSO statements.

Example:

Delete the index DSI and index DSO belonging the STOCKMN_DB.

```
DROP DSI  PRODUCT_IXDSI
               ↑
            DSI name
DROP DSO  PRODUCT_IXDSO
               ↑
            DSO name
```

When the DSI specified in the DROP DSI statement is deleted, all data stored in the database space allocated to that DSI becomes invalid.

When the index storage structure is defined by an index definition, use the following method to delete it.

Example:

Delete the index belonging to STOCKMN_DB.

```
DROP INDEX STOCKS.STOCK. IDX1
                          ↑
                      Index name
```

The index specified in the DROP INDEX statement is deleted. At this time, the index DSI and index DSO are deleted.

■Deleting table storage structures

To delete table storage structures, use the DROP DSI and DROP DSO statements.

Example:

Delete the table DSIs and table DSO belonging to STOCKMN_DB.

```
DROP DSI  WEST_ORDER_DSI
                ↑
             DSI name
DROP DSI  STOCK_DSI
              ↑
           DSI name
DROP DSO  ORDER_DSO
              ↑
           DSO name
```

The table DSIs specified in the DROP DSI statements and the table DSO specified in the DROP DSO statement are deleted.

■Deleting procedure routines

To delete a procedure routine, use the DROP PROCEDURE statement. When a procedure routine is deleted, the privilege information for the procedure is also deleted.

Example:

Delete PROC002 from the STOCKS schema.

```
DROP   PROCEDURE  STOCKS.PROC002
                        ↑
                   Routine name
```

■Deleting function routine definitions

To delete a function routine, use the DROP FUNCTION statement. When a function routine is deleted, privilege information of the function routine is also deleted.

Example:

Deletes USER001 from schema STOCKS.

```
DROP FUNCTION STOCKS. USER001
                         ↑
                    Routine name
```

■Deleting triggers

To delete a trigger, use the DROP TRIGGER statement.

Example:

Delete the trigger belonging to the STOCKMN_DB.

```
DROP    TRIGGER    STOCKS.ORDER_TRIGGER
                          ↑
                     Trigger name
```

■Deleting views

To delete a view, use the DROP VIEW statement.

Example:

Delete the view belonging to STOCKMN_DB.

```
DROP VIEW STOCKS.STOCK_VIEW CASCADE
                 ↑
          Table name of view
```

The view specified in the DROP VIEW statement is deleted.

■Deleting tables

To delete a table, use the DROP TABLE statement. When a table is deleted, the privilege information for the table is also deleted.

Example:

Delete the tables belonging to STOCKMN_DB.

```
DROP TABLE STOCKS.STOCK RESTRICT
                  ↑
               Table name
DROP TABLE STOCKS.ORDER RESTRICT
                  ↑
               Table name
```

The tables specified in the DROP TABLE statements are deleted.

■Deleting sequences

To delete a sequence, use the DROP SEQUENCE statement. When a sequence is deleted, privilege information of the sequence is also deleted.

Example:

Deletes a sequence belonging to the stock management database.

```
DROP SEQUENCE STOCKS. SEQUENCE1
                          ↑
                    Sequence name
```

■Deleting schemas

To delete a schema, use the DROP SCHEMA statement. When a schema is deleted, the privilege information for the schema is also deleted.

Example:

Delete the schema belonging to STOCKMN_DB.

```
DROP SCHEMA STOCKS RESTRICT
               ↑
           Schema name
```

The schema specified in the DROP SCHEMA statement is deleted.

■Deleting database spaces

To delete a database space, use the DROP DBSPACE statement. When a database space is deleted, the privilege information for the database space is also deleted.

Example:

Delete the database spaces belonging to STOCKMN_DB.

```
DROP DBSPACE   DBSPACE_1
                   ↑
               Database space name
DROP DBSPACE   DBSPACE_2
                   ↑
               Database space name
```

■Deleting a database

To delete the database name, use the DROP DATABASE statement.

Example:

Delete STOCKMN_DB.

```
DROP DATABASE  STOCKNM_DB
                    ↑
               Database name
```

Chapter 4 Storage Structure

From the application program viewpoint, database data is represented in table format. The application program manipulates data as if it were manipulating rows and columns of a table by using structured query language (SQL) statements.

The structure for storing data represented in table format on physical pages is called the storage structure. The storage structure cannot be directly seen from the application program. Regardless of the storage structure used, from the application program viewpoint, the table rows and columns appear as if they are being manipulated according to SQL statements.

However, since the physical data is stored according to the storage structure, the storage structure is an important factor in determining processing efficiency.

Ignoring the interrelationships between transactions, such as exclusion, and focusing on storage structure, one can see the following elements affecting the data processing efficiency of an application program:

- Addition of an index
  Add an index for a table.
- Allocation of database space
  Carefully consider the amount of data to be processed and area usage patterns. Allocate database space for each component of the storage structure.
- Association with the shared buffer pool
  Select a page size and shared buffer pool appropriate to the data processing.

This chapter explains the features of storage structures. The explanation covers the following topics:

4.1 Features of Table Storage Structures

Specify the storage structure according to a data structure organization (DSO) definition. The three types of storage structures are SEQUENTIAL, RANDOM, OBJECT, and BTREE. The SEQUENTIAL, RANDOM, and OBJECT structures are used as storage structures for tables. The BTREE structure is used as the storage structure for indexes.

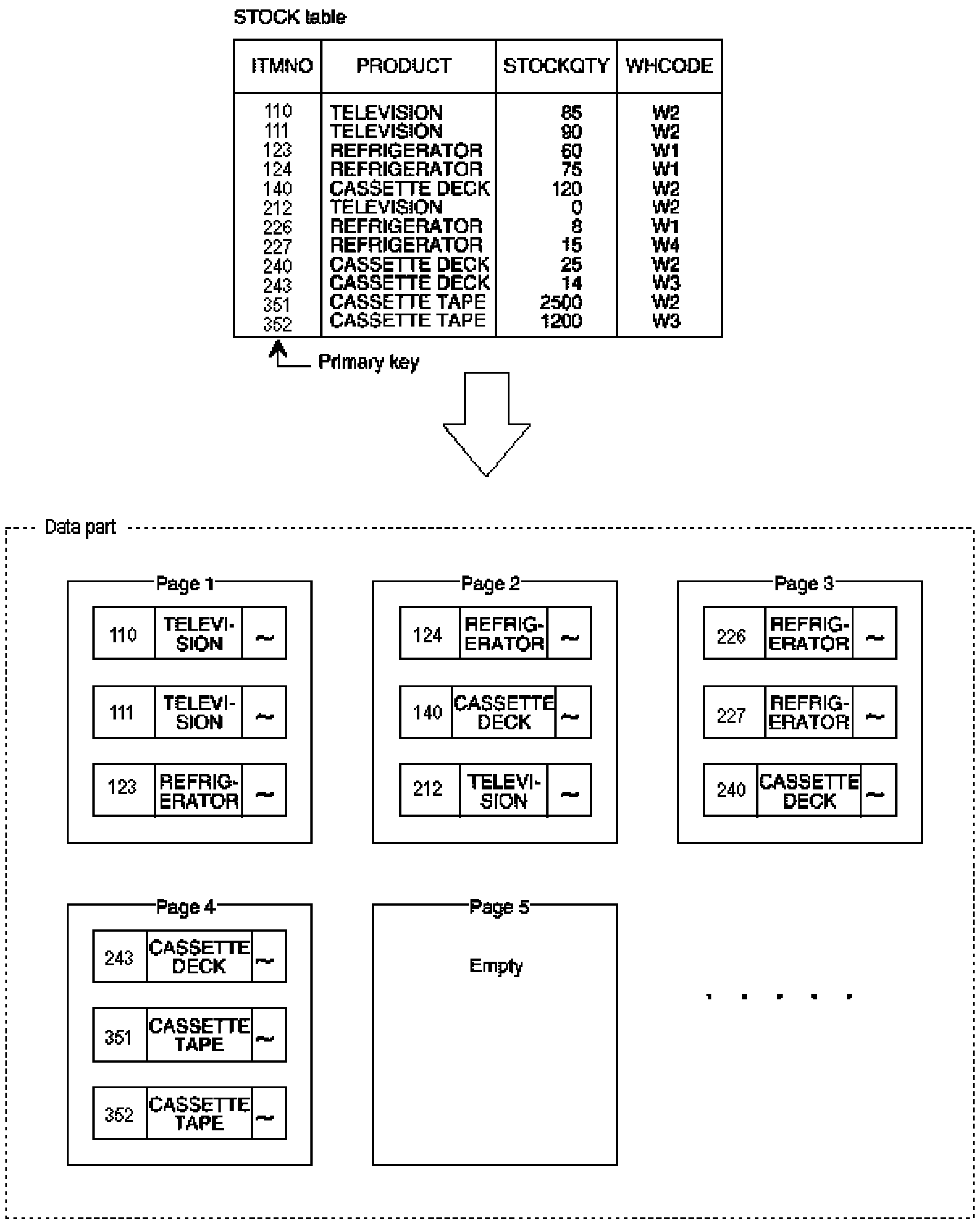
This section explains the features of the table storage structures and the data processing appropriate to those structures.

4.1.1 SEQUENTIAL structure

Data is stored in a SEQUENTIAL structure in the order that the data is inserted.

Figure: Overview of SEQUENTIAL structure shows an overview of the SEQUENTIAL structure, using the STOCK table as an example.

[Figure: Overview of SEQUENTIAL structure]



STOCK table

ITMNO	PRODUCT	STOCKQTY	WHCODE
110	TELEVISION	85	W2
111	TELEVISION	90	W2
123	REFRIGERATOR	60	W1
124	REFRIGERATOR	75	W1
140	CASSETTE DECK	120	W2
212	TELEVISION	0	W2
226	REFRIGERATOR	8	W1
227	REFRIGERATOR	15	W4
240	CASSETTE DECK	25	W2
243	CASSETTE DECK	14	W3
351	CASSETTE TAPE	2500	W2
352	CASSETTE TAPE	1200	W3

■SEQUENTIAL structure features for data processing patterns

The factor having the greatest effect on data processing efficiency is the I-O frequency. The SEQUENTIAL structure has the following features:

- All data is referenced for data processing unless an index exists. Thus, the I-O frequency depends greatly on the data volume. When data manipulations specify a column, the user must add an index corresponding to that column.

For information about items to carefully consider when adding an index, see 4.2.1 "BTREE structure."

■SEQUENTIAL structure page size specification

In a SEQUENTIAL structure, the page size is specified by the PAGESIZE option of the DSO definition.

When specifying the page size, carefully consider the following point:

- If the table does not contain a BLOB-type column, a row of data in the table must fit within in one page.
- If a table including a BLOB-type column is defined or a BLOB-type column is added during the change of a table definition, a row of data in the table may exceed one page. However, the total size of the data of the columns other than the BLOB-type column must not exceed one page.

■When an index must be added to a SEQUENTIAL structure

The user must add an index for a column of the corresponding table that has a unique constraint. If no index is added for such a column, the table cannot be accessed.

4.1.2 RANDOM structure

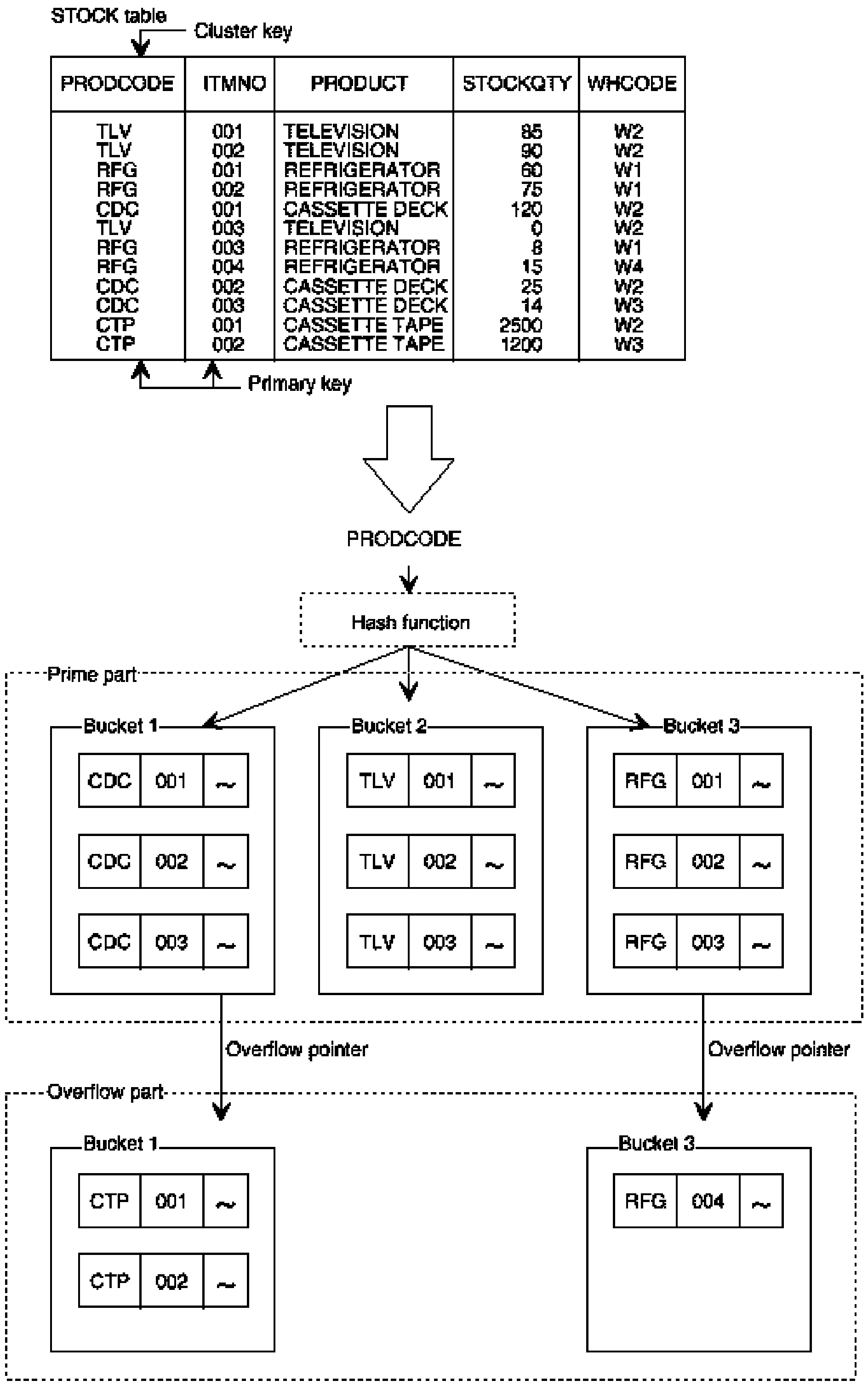
In a RANDOM structure, collections of storage pages (called buckets) are calculated from the values of the group of columns defined as a key for the data. The data is stored in pages within those buckets. If the data cannot fit in a bucket, the SymfoWARE/RDB system automatically creates an overflow part bucket and stores the data in that bucket. The collection of pages that belong to the original bucket are called the prime part for the overflow part.

A group of columns that determines the page for storing data is called a cluster key. A cluster key is determined by the CLUSTER option of the table DSO definition. If the CLUSTER option is omitted, the cluster key becomes the primary key of the corresponding table definition. In a RANDOM structure, data having an equal cluster key is stored in the same packet.

The hash function is used in calculations to determine a bucket from the cluster key value. From the hash function, SymfoWARE/RDB automatically determines a bucket for storing data. If RULE is specified in the storage option of the table DSO definition statement, the data storing bucket is determined from the result of calculating the formula specified at RULE.

Figure: Overview of RANDOM structure shows an overview of the RANDOM structure, using the STOCK table as an example.

[Figure: Overview of RANDOM structure]



PRODCODE	ITMNO	PRODUCT	STOCKQTY	WHCODE
TLV	001	TELEVISION	85	W2
TLV	002	TELEVISION	90	W2
RFG	001	REFRIGERATOR	60	W1
RFG	002	REFRIGERATOR	75	W1
CDC	001	CASSETTE DECK	120	W2
TLV	003	TELEVISION	0	W2
RFG	003	REFRIGERATOR	8	W1
RFG	004	REFRIGERATOR	15	W4
CDC	002	CASSETTE DECK	25	W2
CDC	003	CASSETTE DECK	14	W3
CTP	001	CASSETTE TAPE	2500	W2
CTP	002	CASSETTE TAPE	1200	W3

■RANDOM structure features for data processing patterns

The factor having the greatest effect on data processing efficiency is the I-O frequency. The RANDOM structure has the following features:

- If all values of the cluster key are specified in conditional expressions and a unique constraint is set as the cluster key in data processing, the I/O frequency hardly depends on the data volume.
- In other cases, since all data is referenced, the I-O frequency depends greatly on the data volume. In these

cases, the entire cluster key is not specified in the data processing.

■RANDOM structure page size specification

In a RANDOM structure, a prime part bucket is associated with a single page. An overflow part bucket is also independently associated with a single page. The size of each of these pages can be independently specified. The prime part page size is specified by the PAGESIZE1 option of the DSO definition. The overflow part page size is specified by the PAGESIZE2 option.

When specifying the page size, carefully consider the following points:

- Data corresponding to a single row of the table must fit in one page.
- When setting the prime part page size, carefully consider the average I-O frequency (which depends on the number of data entries within the page).

■When an index must be added to a RANDOM structure

The user must add an index for a column of the corresponding table that has a unique constraint but is not specified as the cluster key. If no index is added for such a column, the table cannot be accessed.

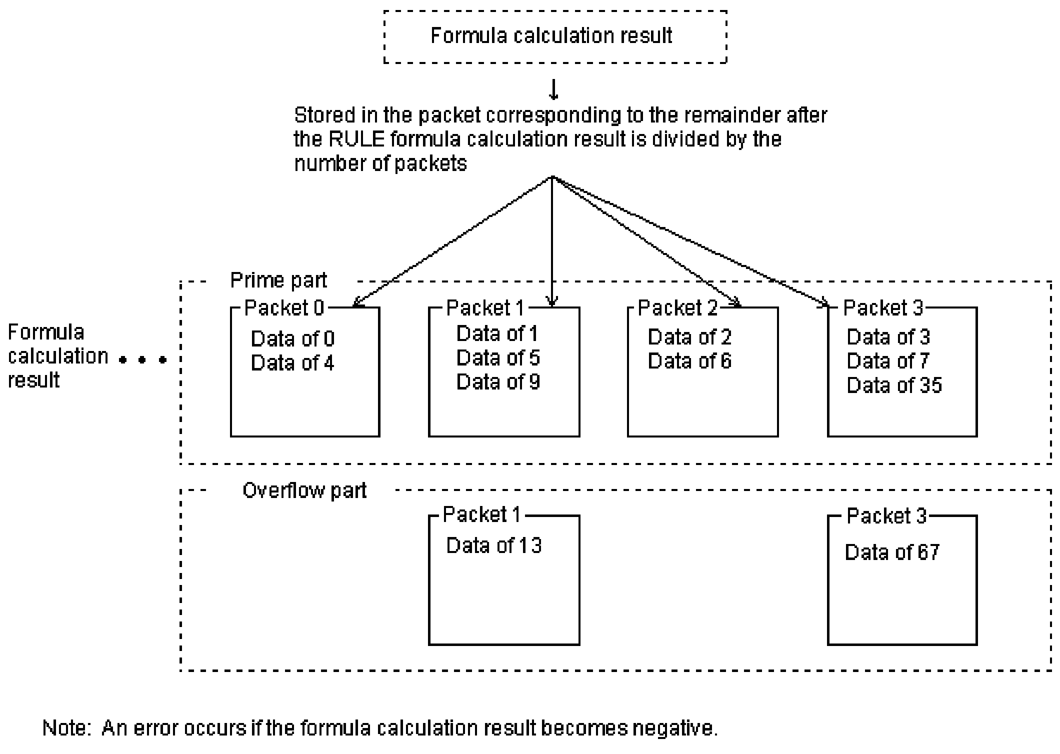
■RANDOM structure data storage position specification

A rule of determining a data storage packet can be specified by setting RULE to the storage option of the table DSO definition statement. When specifying RULE, note the following:

- A RULE formula resulting in a negative value cannot be specified.
- A RULE formula should be designed by considering the cluster key value to be stored in the table and the database I/O count and storage efficiency from the point of view of work.

A data-storing packet is determined from the remainder of dividing the RULE formula calculation result by the number of packets. Figure: Overview of data storage method shows an outline of the data storage method.

[Figure: Overview of data storage method]



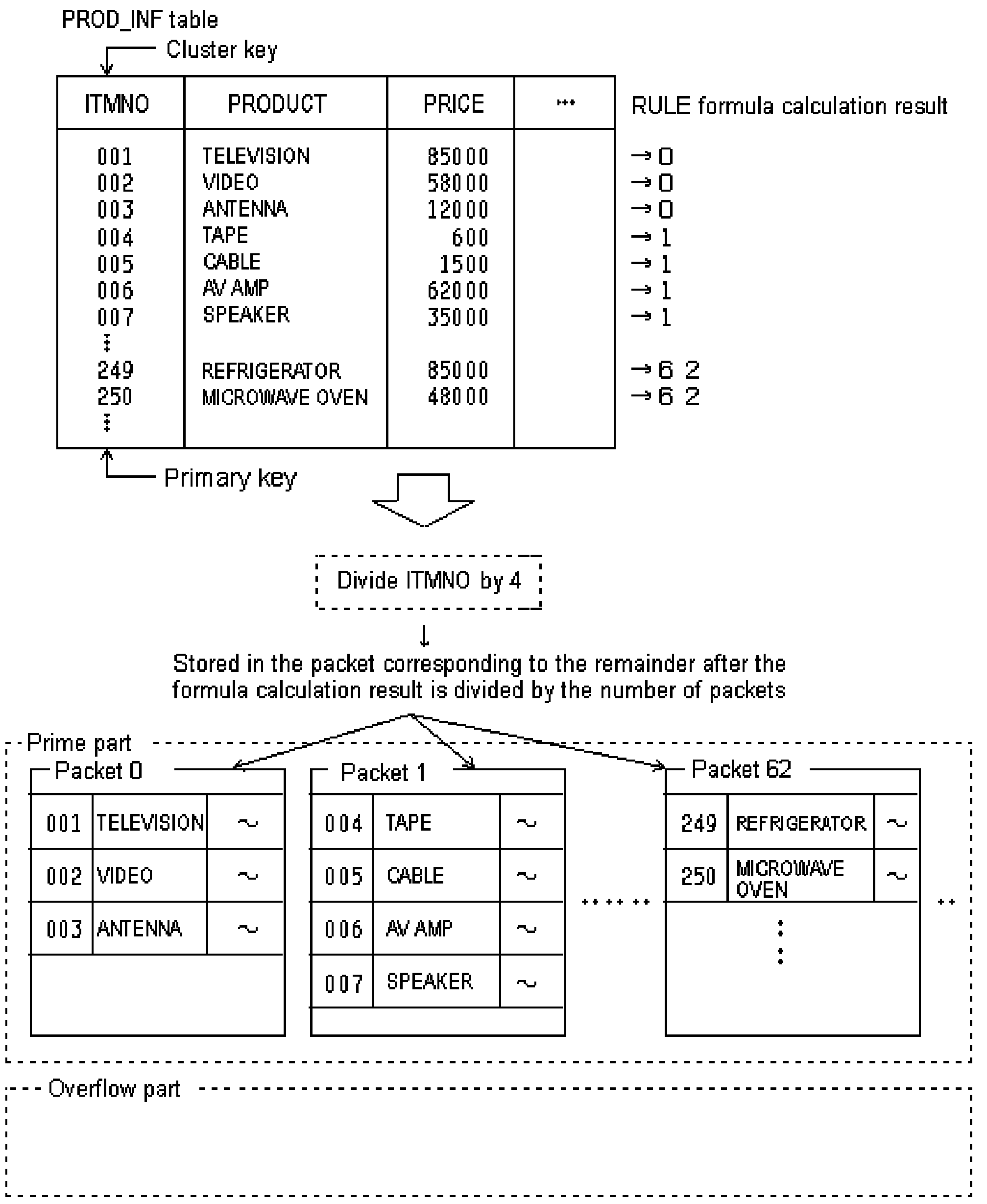
Formula calculation result
Stored in the packet corresponding to the remainder after the RULE formula calculation result is divided by the number of packets
Prime part
Formula calculation result • • •
Packet 0
Data of 0
Data of 4
Packet 1
Data of 1
Data of 5
Data of 9
Packet 2
Data of 2
Data of 6
Packet 3
Data of 3
Data of 7
Data of 35
Overflow part
Packet 1
Data of 13
Packet 3
Data of 67
Note: An error occurs if the formula calculation result becomes negative.

The example below shows the table DSO definition when RULE is specified.

Example:

When cluster keys are generated in ascending order, this definition enhances the storage efficiency and prevents data storage in the overflow part. The record size of the PROD_INF table is set to 200 bytes. By considering the page size (PAGESIZE1) of the prime part, design the RULE formula to store four data items on each page.

```
CREATE DSD  PROD_INF_DSO  FRDM  COMP_A.PROD_INF
 TYPE RANDDM ( PAGESIZE1(1). PAGESIZE2(1).CLUSTER (ITMNO),
              RULE(ITMNO/4))
```



Note: A sufficient number of packets should be prepared by considering the data counts in the table.
The number of packets can be determined from the prime part allocation size specified to the subject of allocation in the table DSI definition and the prime part page size.
The formula below is used to calculate the number of packets. (One is subtracted because SymfoWARE/RDB uses one page for page management.)

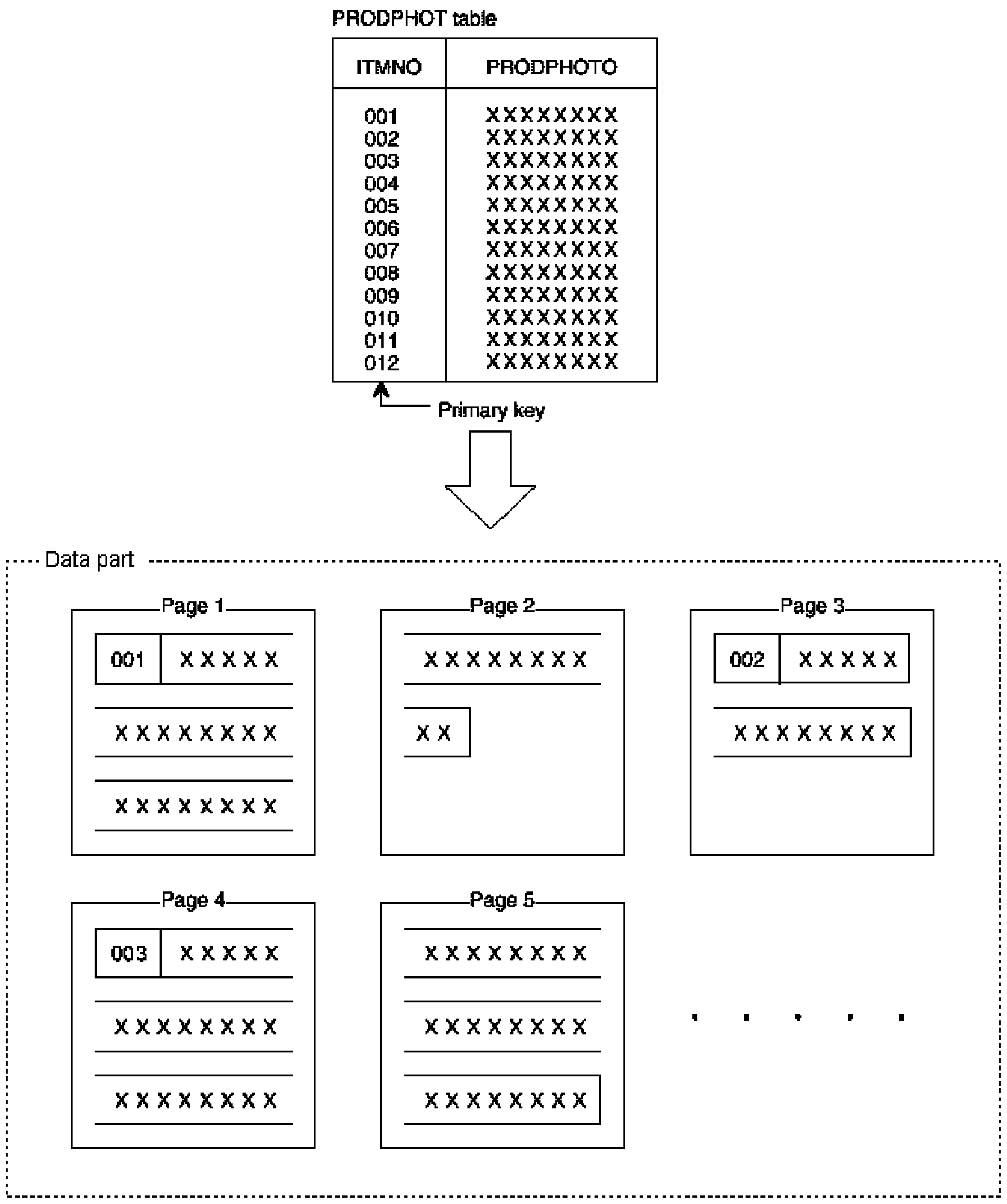
$$\text{Number of packets} = \frac{\text{Prime part allocation size}}{\text{Prime part page size}} - 1$$

4.1.3 OBJECT structure

An OBJECT structure stores BLOB-type data such as photograph in the order that the data is inserted.

Figure: Overview of OBJECT structure shows an overview of the OBJECT structure, using the PRODPHOT table as an example.

[Figure: Overview of OBJECT structure]



■OBJECT structure page size specification

In an OBJECT structure, the page size is specified by the PAGESIZE option of the DSO definition. The specified page size must be 32.

4.2 Features of the Index Storage Structure

The index storage structure is the BTREE structure only. This section explains the features of the index storage structure and the data processing appropriate to that structure.

4.2.1 BTREE structure

BTREE structure is the storage structure used for indexes. Internally, the BTREE structure consists of a tree structure index part and a data part. The index part, which consists of groups of correspondence information values for index keys and base table data, manages the pages where data is stored. The portion consisting of pages in which data is stored is called the data part for the index part.

If ADDRESS is specified as the base expression for the DSO definition, the table record storage address is used as information corresponding to the base table data. If KEY is specified, the cluster key is used.

If data cannot fit in a given page of the data part, the SymfoWARE/RDB system automatically creates a new page. The SymfoWARE/RDB system then divides the data between the new page and the original page for storage (called page splitting). Index part page splitting may be performed together with data part page splitting.

If the index key is not a unique key, multiple base table data correspondence information values may exist for a single index key value. This multiplicity (number of base table data correspondence information values), which also depends on the index key values, is not fixed. Therefore, the BTREE structure groups index keys and base table data correspondence information for management. Ascending order is guaranteed for these key groups. Also, to improve storage efficiency, the system performs front compression for the index key portion.

Figure: Overview of BTREE structure shows an overview of the BTREE structure, using the STOCK table as an example. In this example, an index is assumed to be defined for the WHCODE column, and the table storage structure is assumed to be a SEQUENTIAL structure.

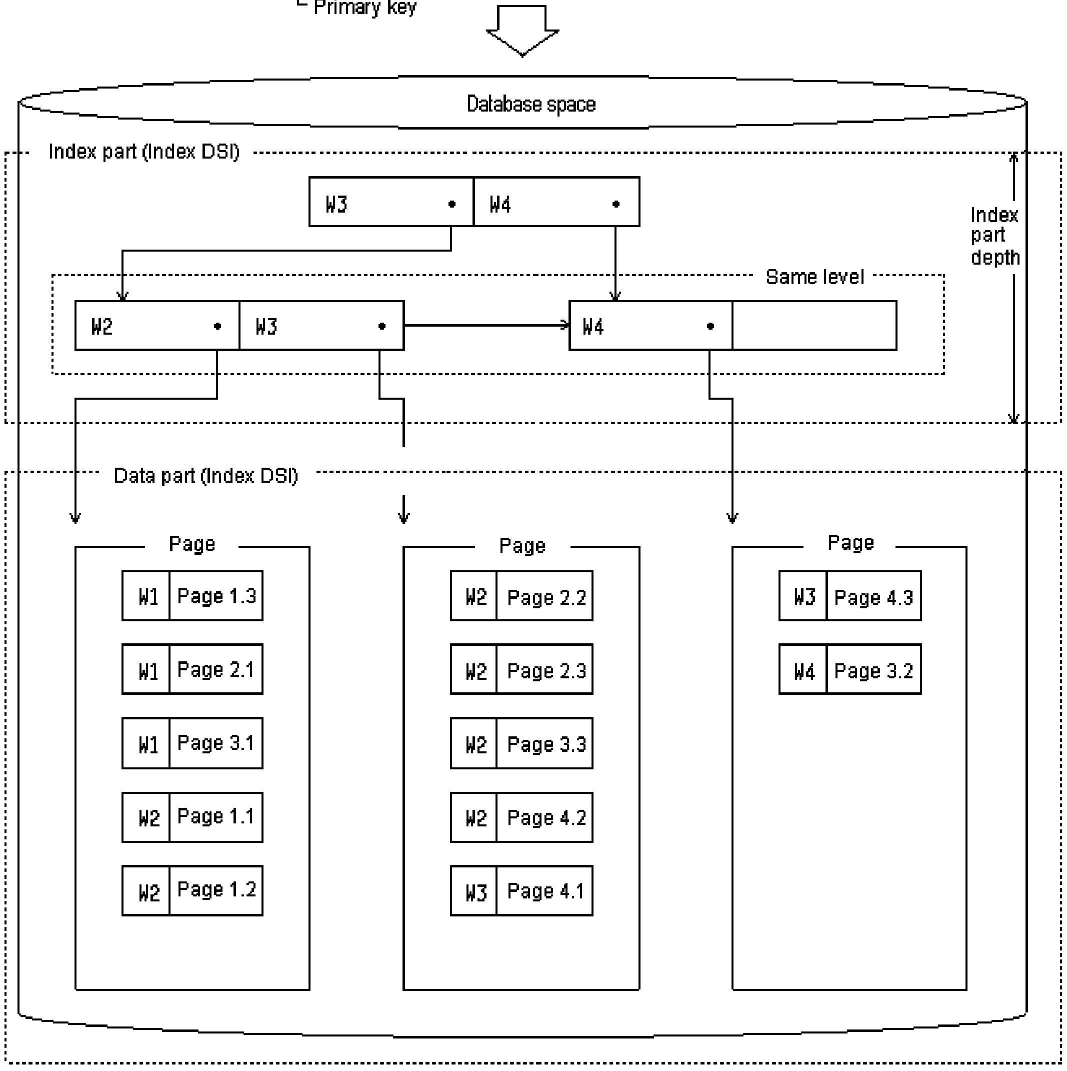
[Figure: Overview of BTREE structure]

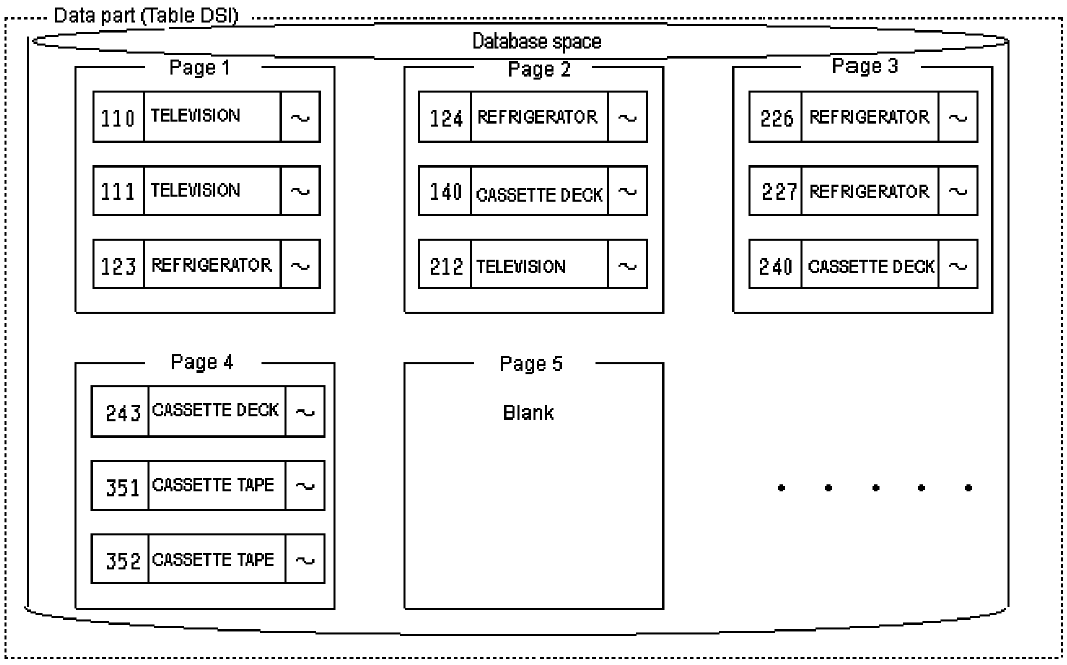
Index key ↓

STOCK table

ITMNO	PRODUCT	STOCKQTY	WHCODE
110	TELEVISION	85	W2
111	TELEBISION	90	W2
123	REFRIGERATOR	60	W1
124	REFRIGERATOR	75	W1
140	CASSETTE DECK	120	W2
212	TELEVISION	0	W2
226	REFRIGERATOR	8	W1
227	REFRIGERATOR	15	W4
240	CASSETTE DECK	25	W2
243	CASSETTE DECK	14	W3
351	CASSETTE TAPE	2500	W2
352	CASSETTE TAPE	1200	W3

↑ Primary key





■BTREE structure features for data processing patterns

The factor having the greatest effect on data processing efficiency is the I-O frequency. The BTREE structure has the following features:

- In the following cases, the I-O frequency depends on the depth of the index part. The range of index key values is specified, or only the values of the leftmost column of an index key consisting of a group of columns is specified in the data processing.
- Otherwise, the relevant index is not used.

■BTREE structure page size specification

In a BTREE structure, the page size can be independently specified for the index part and the data part. The data part page size is specified by the PAGESIZE1 option of the DSO definition. The index part page size is specified by the PAGESIZE2 option.

When specifying the page size, carefully consider the following points:

- The group of correspondence information values for index keys and base table data must fit as two records. These records must be in a single page of the index part and data part.
- If the amount of data that fits in the index part and data part is small, the page utilization rate (data storage rate) decreases. To increase the page utilization rate, specify a page size having a surplus area.
- To fit a maximum number of data entries in a single page of the data part, carefully consider and set the page splitting points. Frequent page splitting decreases the processing efficiency.

■Considerations when adding indexes

The user can add multiple indexes to a single base table. By adding indexes, the user increases the processing efficiency for searches. However, in the following cases, since processing for upgrading all related indexes is performed, overall processing efficiency may decrease:

- Application program processing updates values of columns for which indexes have been added or base table data correspondence information.
- Application program processing primarily performs insertions and deletions.

4.3 Allocating Space

This section gives details on the following topics related to the allocation of database space for DSIs:

- Formats for associating storage structures and database spaces
- Considerations when allocating space

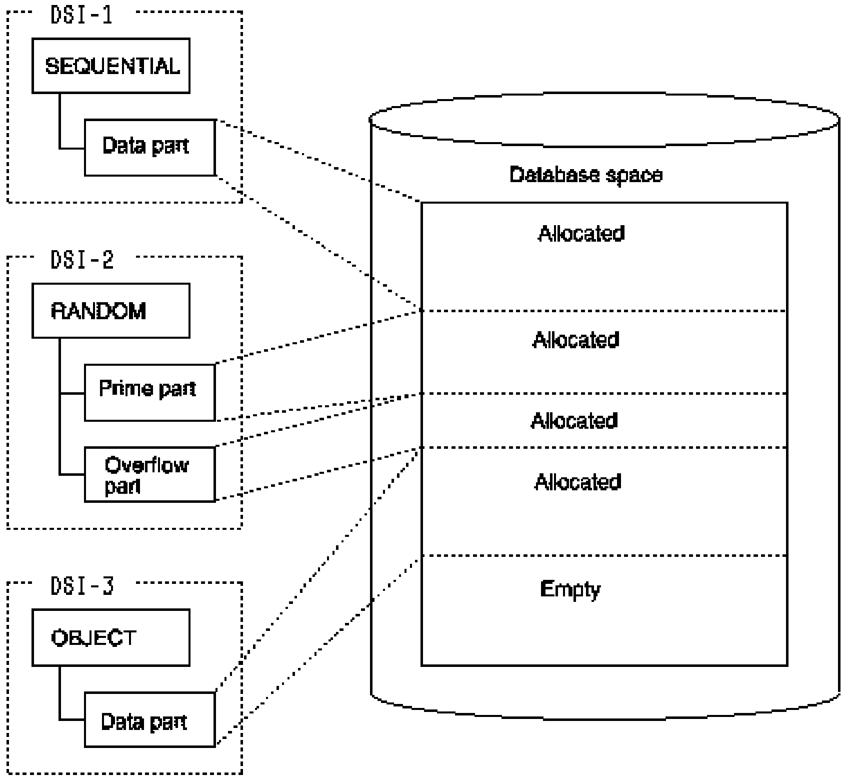
4.3.1 Formats for associating storage structures and database spaces

Each type of storage structure consists of several components. When database space is allocated to storage structures, each of these components is associated with a database space. The user can use one of the following formats to make these associations:

■Format 1

This format associates multiple components with the same database space, as shown in Figure: Database space sharing among components. This format reduces the number of database spaces to be used. However, multiple DSIs can be associated with the database space. When the DSIs are used at the same time, accesses to the database space can be concentrated.

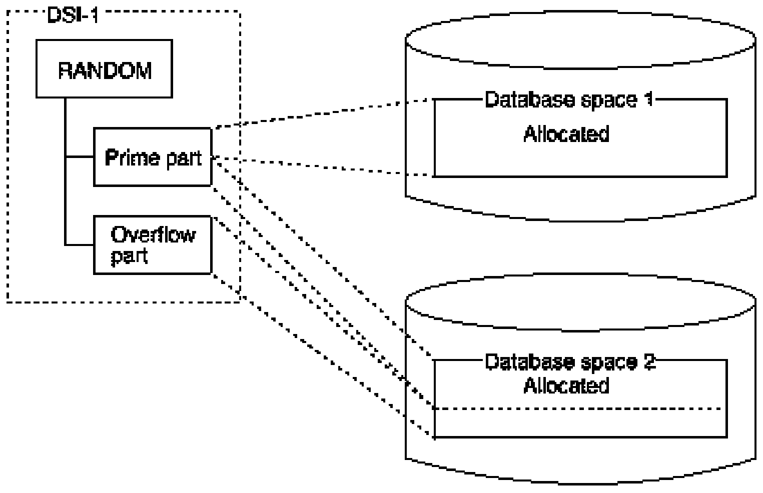
[Figure: Database space sharing among components]



■Format 2

This format associates one component with the multi-database space, as shown in Figure: Multi-database space components. This format enables a massive database to be built. However, the number of database spaces to be used increases.

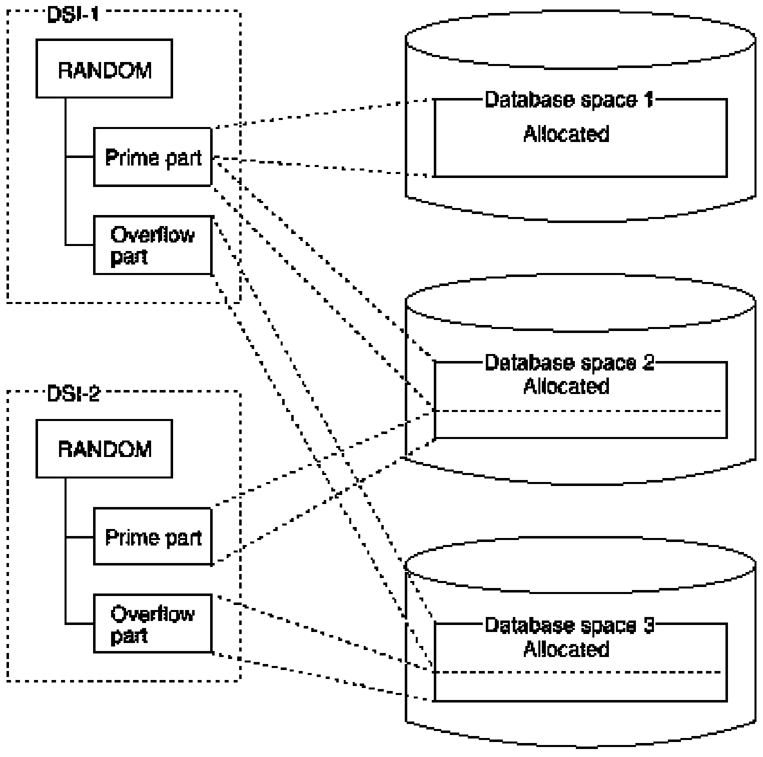
[Figure: Multi-database space components]



■Format 3

This format combines the two previous formats, as shown in Figure: Mixed database space configuration. In this case, infrequently used overflow parts are concentrated in a single database space.

[Figure: Mixed database space configuration]



4.3.2 Considerations when allocating space

In a storage structure where one DSO is divided into multiple DSIs, the user can maintain individual DSIs in parallel by allocating database spaces on different hard disks for individual DSIs or DSI groups.

Also, if the number of hard disk units and the capacity permits, consider the following points. Take into account the transitions required for reaccessing storage structure components:

- For a RANDOM structure
  Allocate the prime and overflow parts in database spaces on separate hard disks.
- For a BTREE structure
  Allocate the index and data parts in database spaces on separate hard disks.

4.4 Estimating the Required Amount of Database Space

This section explains how to estimate the required amount of database space for each type of storage structure. Criteria are presented later for determining values used in the formulas, such as utilization rates. However, since these values differ depending on the data, ranges of values are given. The user should make space estimates with some surplus built in. (That is, multiply the estimates by a safety factor.)

■SEQUENTIAL structure

① Record length $\boxed{}$	fixed-length-part $\boxed{}$ + variable-length-data-control-part $\boxed{}$ + variable-length-part $\boxed{}$ + null-tag-part $\boxed{}$ + 26 Fixed length part : Sum of lengths of fixed length columns Variable length data control part : number-of-variable-length-columns × 4 + 2 Variable length part : Sum of lengths of variable length columns NULL tag part : Number of columns not having a NOT NULL specification in the relevant base table If no variable length columns exist, set zero for both the variable length data control part and the variable length part in the calculation.
② Storage area size in page $\boxed{}$	page-size $\boxed{}$ − 94
③ Decision of record exceeding one page	If ② storage area in page is equal to or greater than ① record length, use ④ and ⑤ to obtain the amount of space. If ② storage area in page is smaller than ① record length, use ⑥ and ⑦ to obtain the amount of space.
④ Number of records in a page $\boxed{}$	$\left\lfloor \dfrac{\text{② Storage area size in page } \boxed{}}{\text{① record-length } \boxed{}} \right\rfloor^{-}$ $\lfloor\ \rfloor^{-}$: Truncate fractional digits to the next lowest integer.
⑤ Amount of space $\boxed{}$	$\left\{ \left\lceil \dfrac{\text{total-number-of-records } \boxed{}}{\text{③ number-of-records-in-a-page } \boxed{}} \right\rceil^{+} + \text{number-of-cpus} + 1 \right\} \times \text{page-size } \boxed{} \times \text{safety-factor } \boxed{}$ $\lfloor\ \rfloor^{+}$: Round up fractional digits to the next highest integer. Specify a value of at least 1.3 as the safety factor when making this estimate. Change the calculated amount of space to make it an integral multiple of the page size. Note: If ORDER(0) is specified by the storage option of the DSO definition, any deleted area (area of deleted records) is not reused. Therefore, specify the number of inserted records (including deleted records) as the total number of records in the calculation.
⑥ Number of pages required $\boxed{}$	$\left\lfloor \dfrac{\text{① record-length } \boxed{}}{\text{② Storage area size in page} - 58 \ \boxed{}} \right\rfloor^{+}$ $\lfloor\ \rfloor^{+}$: Raise fractions to the next integer.
⑦ Amount of space $\boxed{}$	$\left\{ \text{⑥ Number of pages required } \boxed{} \times \text{Total number of records } \boxed{} + \text{number-of-cpus} + 1 \right\} \times \text{page-size } \boxed{} \times \text{safety-factor } \boxed{}$ Specify a value of at least 1.3 as the safety factor when making this estimate. Change the calculated amount of space to make it an integral multiple of the page size. Note : If ORDER(0) is specified by the storage option of the DSO definition, any deleted area (area of deleted records) is not reused. Therefore, specify the number of inserted records (including deleted records) as the total number of records in the calculation.

■RANDOM structure (when a unique constraint is set for the cluster key)

① Record length $\boxed{}$

$$\overset{\text{fixed-length-part}}{\boxed{}} + \overset{\text{variable-length-data-control-part}}{\boxed{}} + \overset{\text{variable-length-part}}{\boxed{}} + \overset{\text{null-tag-part}}{\boxed{}} + 26$$

Fixed length part	:	Sum of lengths of fixed length columns (excluding columns forming the cluster key)
Variable length data control part	:	number-of-variable-length-columns (excluding columns forming the cluster key) × 4 + 6
Variable length part	:	Sum of lengths of variable length columns and columns forming the cluster key plus 12.
NULL tag part	:	Number of columns not having a NOT NULL specification in the relevant base table

Even when there are no variable length columns, calculate the variable length data control part and the variable length part.

② Number of records in a page $\boxed{}$

$$\left\lfloor \frac{\left(\overset{\text{page-size}}{\boxed{}} - 94\right)}{\overset{\text{①record-length}}{\boxed{}}} \right\rfloor$$

$\lfloor\ \rfloor$: Truncate fractional digits to the next lowest integer.

Page size: Calculated separately for the prime and overflow parts

③ Amount of space for prime part $\boxed{}$

$$\left\{ \frac{\left\lfloor \left(\overset{\text{total-number-of-records}}{\boxed{}} / \overset{\text{②number-of-records-in-a-page}}{\boxed{}}\right) \right\rfloor^{+}}{\underset{\text{prime-part-utilization-rate}}{\boxed{}}} + 1 \right\} \times \overset{\text{prime-part-page-size}}{\boxed{}}$$

$\lfloor\ \rfloor^{+}$: Round up fractional digits to the next highest integer.

For information about the prime part utilization rate, see the next section entitled, "Utilization rate criteria."

Note: To decrease the chance of overflow, make the total amount of space allocated to the prime part adhere to the following condition:

Condition ⇨ The value of (prime-part-space-amount - prime-part-page-size)/prime-part-page-size is a power of 2.

④ Amount of space for overflow part $\boxed{}$

$$\left\{ \frac{\left\lfloor \left(\overset{\text{total-number-of-records}}{\boxed{}} \times \overset{\text{overflow-rate}}{\boxed{}} / \overset{\text{②number-of-records-in-a-page}}{\boxed{}}\right) \right\rfloor^{+}}{\underset{\text{overflow-part-utilization-rate}}{\boxed{}}} + 1 \right\} \times \overset{\text{overflow-part-page-size}}{\boxed{}} \times \overset{\text{safety-factor}}{\boxed{}}$$

$\lfloor\ \rfloor^{+}$: Round up fractional digits to the next highest integer.

For information about the overflow part utilization rate, see the next section entitled, "Utilization rate criteria." Specify a value of at least 1.3 as the safety factor when making this estimate multiple of the page size. Change the calculated amount of space to make it an integral multiple of the page size.

■RANDOM structure (when a unique constraint is not set for the cluster key)

①Record length $\boxed{}$	See the record length formula for a RANDOM structure (when a unique constraint is set for the cluster key).
②Number of records in a page $\boxed{}$	$\left\lfloor \dfrac{(\overset{\text{page-size}}{\boxed{}} - 94)}{\overset{\text{① record-length}}{\boxed{}}} \right\rfloor$ $\lfloor\ \rfloor$: Truncate fractional digits to the next lowest integer. Page size: Calculated only for the overflow part
③ Amount of space for prime part $\boxed{}$	$(\overset{\text{number-of-different-key-values-for-the-cluster-key}}{\boxed{}} + 1) \times \overset{\text{prime-part-page-size}}{\boxed{}}$
④ Amount of space for overflow part $\boxed{}$	$\left\{ \dfrac{(\overset{\text{total-number-of-records}}{\boxed{}} \times \overset{\text{overflow-rate}}{\boxed{}})}{\overset{\text{② Number-of-records-in-a-page}}{\boxed{}}} + (\overset{\text{number-of-different-key-values-for-the-cluster-key}}{\boxed{}} + 1) \right\} \times \overset{\text{overflow-part-page-size}}{\boxed{}} \times \overset{\text{safety-factor}}{\boxed{}}$ For information about the overflow rate, see the next section entitled, "Utilization rate criteria." Specify a value of at least 1.3 as the safety factor when making this estimate multiple of the page size. Change the calculated amount of space to make it an integral multiple of the size.

■OBJECT structure

① Logical record length $\boxed{\qquad}$	sum-of-lengths-of-columns $\boxed{\qquad\qquad}$
② Storage record length $\boxed{\qquad}$	page-size $\boxed{\qquad}$ - 144
③ Number of pages necessary for one logical record $\boxed{\qquad}$	$\left\lfloor \dfrac{\text{① logical-record-length } \boxed{\qquad}}{\text{② storage-record-length } \boxed{\qquad}} \right\rfloor^{+}$ $\lfloor\ \rfloor^{+}$: Round up fractional digits to the next highest integer.
④ Amount of space $\boxed{\qquad}$	(③ number-of-pages-necessary-for-one-logical-record $\boxed{\qquad\qquad}$ X total-number-of-records $\boxed{\qquad}$ + 1) X page-size $\boxed{\qquad}$ X safety-factor $\boxed{\qquad}$ Specify a value of at least 1.3 as the safety factor when making this estimate. Change the calculated amount of space to make it an integral multiple of the page size.

■BTREE structure data part

① Entry size $\boxed{}$	- When the storage structure of the corresponding table is a SEQUENTIAL structure: $\left\lfloor \overset{\text{sum-of-lengths-of-columns-forming-the-index-key}}{\boxed{}} \times \left(1 - \overset{\text{key-compression-rate}}{\boxed{}} \right) \right\rfloor^{+} + 20$ $\lfloor \quad \rfloor^{+}$: Round up fractional digits to the next highest integer. For information about the key compression rate, see the next section entitled, "Key compression rate criteria."
	- When the storage structure of the corresponding table is a RANDOM structure: $\left\lfloor \overset{\text{sum-of-lengths-of-columns-forming-the-index-key}}{\boxed{}} \times \left(1 - \overset{\text{key-compression-rate}}{\boxed{}} \right) \right\rfloor^{+} + \overset{\text{cluster-key-size}}{\boxed{}} + 20$ $\lfloor \quad \rfloor^{+}$: Round up fractional digits to the next highest integer. For information about the key compression rate, see the next section entitled, "Key compression rate criteria."
② Section size $\boxed{}$	$\overset{\text{① entry-size}}{\boxed{}} \times 10 + 2$
③Number of sections in a page	$\left\lfloor \dfrac{\left(\overset{\text{data-part-page-size}}{\boxed{}} - 110 \right)}{\overset{\text{② section-size}}{\boxed{}}} \right\rfloor$ $\lfloor \quad \rfloor$: Truncate fractional digits to the next lowest integer.

④Number of entries in a page $\boxed{}$	- When the③ number of sections in a page is greater than or equal to two: $\left\lfloor \frac{\overset{\text{② section size}}{\boxed{}}}{\overset{\text{① entry size}}{\boxed{}}} \right\rfloor \times \overset{\text{③number-of-sections-in-a-page}}{\boxed{}} \times \overset{\text{data-part-utilization-rate}}{\boxed{}}$ $\lfloor\ \ \rfloor$: Truncate fractional digits to the next lowest integer. For information about the data part utilization rate, see the next section entitled, "Utilization rate criteria."
	- When the③ number of sections in a page is less than two: $\left\lfloor \frac{(\overset{\text{data-part-page-size}}{\boxed{}} - 110) / 2}{\overset{\text{① entry-size}}{\boxed{}}} \right\rfloor \times 2 \times \overset{\text{data-part-utilization-rate}}{\boxed{}}$ $\lfloor\ \ \rfloor$: Truncate fractional digits to the next lowest integer. For information about the data part utilization rate, see the next section entitled, "Utilization rate criteria."
⑤ Number of data part pages	$\left\lfloor \frac{\overset{\text{number-of-table-records}}{\boxed{}}}{\overset{\text{④number-of-entrie-in-a-page}}{\boxed{}}} \right\rfloor^{+}$ $\lfloor\ \ \rfloor^{+}$: Round up fractional digits to the next highest integer. The number of data part pages obtained as the calculation result is also used in estimating the size of the index part.
⑧ Amount of data part space	$\left\{ \overset{\text{⑤ Number-of-data-part-pages}}{\boxed{}} + 1 \right\} \times \overset{\text{data-part-page-size}}{\boxed{}} \times \overset{\text{safety-factor}}{\boxed{}}$ Specify a value of at least 1.3 as the safety factor when making this estimate. Change the calculated amount of space to make it an integral multiple of the page size.

■BTREE structure index part

① Entry size $\boxed{}$	- When the storage structure of the corresponding table is a SEQUENTIAL structure: $$\left\lfloor \left(\underset{\text{sum-of-lengths-of-columns-forming-the-index-key}}{\boxed{}} + 12 \right) \times \left(1 - \underset{\text{key-compres-sion-rate}}{\boxed{}} \right) \right\rfloor^{+} + 10$$ $\lfloor\ \rfloor^{+}$: Round up fractional digits to the next highest integer. For information about the key compression rate, see the next section entitled, "Key compression rate criteria."
	- When the storage structure of the corresponding table is a RANDOM structure: $$\left\lfloor \left(\underset{\text{sum-of-lengths-of-columns-forming-the-index-key}}{\boxed{}} + \underset{\text{cluster-key-size}}{\boxed{}} + 12 \right) \times \left(1 - \underset{\text{key-compres-sion-rate}}{\boxed{}} \right) \right\rfloor^{+} + 10$$ $\lfloor\ \rfloor^{+}$: Round up fractional digits to the next highest integer. For information about the key compression rate, see the next section entitled, "Key compression rate criteria."
② Section size $\boxed{}$	$$\underset{\text{① entry-size}}{\boxed{}} \times 10 + 2$$
③Number of sections in a page	$$\left\lfloor \frac{\left(\underset{\text{index-part-page-size}}{\boxed{}} - 110 \right)}{\underset{\text{② section-size}}{\boxed{}}} \right\rfloor$$ $\lfloor\ \rfloor$: Truncate fractional digits to the next lowest integer.

④Number of entries in a page $\boxed{}$	- When the③ number of sections in a page is greater than or equal to two: $\left\lfloor \dfrac{\text{② section-size } \boxed{}}{\text{① entry-size } \boxed{}} \right\rfloor \times \text{③number-of-sections-in-a-page } \boxed{} \times \text{index-part-utilization-rate } \boxed{}$ $\lfloor\ \rfloor$: Truncate fractional digits to the next lowest integer. For information about the index part utilization rate, see the next section entitled, "Utilization rate criteria."
	- When the③ number of sections in a page is less than two: $\left\lfloor \dfrac{(\text{index-part-page-size } \boxed{} - 110)\ /\ 2}{\text{① entry-size } \boxed{}} \right\rfloor \times 2 \times \text{index-part-utilization-rate } \boxed{}$ $\lfloor\ \rfloor$: Truncate fractional digits to the next lowest integer. For information about the index part utilization rate, see the next section entitled, "Utilization rate criteria."
⑤ Index level $\boxed{}$	$(\text{④number-of-entries-in-a-page } \boxed{})^{L} \geqq \text{number-of-data-part-pages } \boxed{}$ Obtain the minimum L that satisfies this relationship. Example: number-of-entries-in-an-BTREE-index-part-page ≒ 20 number-of-data-part-pages ≒ 1,300 ----------------------------------- L=1 (number-of-entries-in-a-BTREE-index-part-page)^{1} = 20^{1} = 20 L=2 (number-of-entries-in-a-BTREE-index-part-page)^{2} = 20^{2} = 400 L=3 (number-of-entries-in-a-BTREE-index-part-page)^{3} = 20^{3} = 8,000 ⇨ L=3
⑥Number of index part pages $\boxed{}$	$\sum_{i=1}^{\text{⑤ index-level } \boxed{}} \left\lfloor \text{number-of-data-part-pages-}P_{(i)}\ /\ \text{④ number-of-entries-in-a-page } \boxed{} \right\rfloor^{+}$ Example: number-of-entries-in-an-BTREE-index-part-page ≒ 20 number-of-data-part-pages-P = 1,300 index-level ≒ 3 ----------------------------------- $P_{(1)}$ = 1,300 ÷ 20 = 65 $P_{(2)}$ = 65 ÷ 20 = 4 $P_{(3)}$ = 4 ÷ 20 = 1 number-of-index-part-pages (total) = 70 $\lfloor\ \rfloor^{+}$: Round up fractional digits to the next highest integer.
⑦ Amount of index part space	$\left\{ \text{⑥number-of-index-part-pages } \boxed{} + 1 \right\} \times \text{index-part-page-size } \boxed{} \times \text{safety-factor } \boxed{}$ Specify a value of at least 1.3 as the safety factor when making this estimate. Change the calculated amount of space to make it an integral multiple of the page size.

■Utilization rate criteria

The utilization rates (average utilization rate of each page) of the prime and overflow parts of a RANDOM structure vary depending on the following factors. These factors are the data key values and the order of data additions and deletions. Similarly, the utilization rates vary depending on these factors for the data and index parts of a BTREE structure and the overflow rate of a RANDOM structure. (The overflow rate of a RANDOM structure is the rate of overflow to the overflow part relative to the total number of records.) When estimating the amounts of space needed, use the following kinds of criteria for the various values in the formulas.

If the number of records in a page is small, the various utilization rates are lower, and the overflow rate is higher.

Prime part utilization rate	:	0.4 to 0.8
Overflow part utilization rate	:	0.2 to 0.5
Overflow rate	:	0.2 to 0.6 (when a unique constraint is set for the cluster key).
	:	0.9 to 1.0 (when a unique constraint is not set for the key).
BTREE data part utilization rate	:	0.5
BTREE index part utilization rate	:	0.5

■Key compression rate criteria

The key is compressed and stored in the BTREE structure data part and index part. Use the following kinds of criteria for the key compression rates in the formulas:

BTREE data part key compression rate	:	0.3
BTREE index part key compression rate	:	0.5

■Example of estimating required space for each storage structure

Example 1:

Estimate of the required amount of space for a SEQUENTIAL structure

Calculate the required amount of space for the ORDER table (total number of records: 30,000) having this kind of structure:

```
CREATE TABLE  STOCKS.ORDER (
                            CUSTOMER   SMALLINT   NOT NULL,
                            PRODNO     SMALLINT   NOT NULL,
                            PRICE      INTEGER,
                            ORDERQTY   SMALLINT
                           )
```

Storage structure:
    SEQUENTIAL structure
Page size:
    32 kilobytes
Safety factor:
    1.3

```
record-size                  =  (CUSTOMER + PRODNO + PRICE + ORDERQTY)
                                + (number-of-variable-length-columns × 4 + 2) +
                                length-of-variable-length-columns + number-of-null-tags + 26
                             ≒  ( 2 + 2 + 4 + 2) + 0 + 0 + 2 + 26 *1
                             =  38

number-of-records-in-a-page  =  (page-size - 94)/record-length
                             =  (32768 - 94) / 38
                             =  859.8421053...
                             ≒  859 *2

amount-of-space              =  (total-number-of-records/number-of-records-in-a-page + 1) × page-size
                                × safety-factor
                             =  (30000 / 859 + 1) × 32768 × 1.3
                             =  (35 + 1) × 32768 × 1.3 *3
                             =  1533542.4 (bytes)
                             ≒  1498 (kilobytes)
                             ⇒  1504 (kilobyte) *4
*1  Since no variable length columns exist, zero is set for all terms related to variable length in the
    calculation.
*2  Fractional digits are truncated.
*3  35 is obtained from rounding up (30000/859) to an integer.
*4  The amount of space is changed to be an integral multiple of the page size.
```

Example 2:

Estimate of the required amount of space for a RANDOM structure

Calculate the required amount of space for the ORDER table (total number of records: 30,000) having this kind of structure:

```
CREATE TABLE  STOCKS.ORDER(
                           CUSTOMER   SMALLINT   NOT NULL,
                           PRODNO     SMALLINT   NOT NULL,
                           PRICE      INTEGER,
                           ORDERQTY   SMALLINT,
                           PRIMARY KEY (CUSTOMER, PRODNO)
                          )
```

The storage structure is designed as follows. The cluster key is the PRIMARY KEY (a unique constraint is set). Thus, the formulas for a RANDOM structure (when a unique constraint is set for the cluster key) are used.

Storage structure:
    RANDOM structure with CUSTOMER and PRODNO as the cluster key
Page size:
    4 kilobytes for both the prime part and the overflow part
Prime part utilization rate:
    0.5
Overflow part utilization rate:
    0.2
Overflow rate:
    0.2
Safety factor:
    1.3

record-size	= (PRICE + ORDERQTY) + (number-of-variable-length-columns × 4 + 6)
	+ (CUSTOMER + PRODNO + 12) + number-of-null-tags + 26
	= (4 + 2) + (0 × 4 + 6) + (2 + 2 + 12) + 2 + 26
	= 56
number-of-records-in-a-page	= (page-size - 94)/record-length (*1)
	= (4096 - 94) / 56
	= 71.46428571...
	≒ 71 (*2)
amount-of-space-for-prime-part	= ((total-number-of-records/number-of-records-in-a-page)/ prime-part-utilization-rate + 1) × prime-part-page-size
	= ((30000 / 71) / 0.5 + 1) × 4096
	= (423 / 0.5 + 1) × 4096 (*3)
	= 3469312 (bytes)
	= 3388 (kilobytes)
	[number-of-pages = 3388/4 = 847
	$2^n \geqq 847$ --> n = 10
	reference-number-of-pages = 2^{10}+1=1025] (*4)
	= 1025 × 4k
	⇒ 4100 (kilobytes)
amount-of-space-for-overflow-part	= ((total-number-of-records × overflow-rate/number-of-records-in-a-page)/ overflow-part-utilization-rate + 1) × overflow-part-page-size × safety-factor
	= ((30000 × 0.2 / 71) / 0.2 + 1) × 4096 × 1.3
	= (85/0.2 + 1) × 4096 × 1.3 (*5)
	= 2268364.8 (bytes)
	≒ 2216 (kilobytes) (*6)

*1 Since the page size is equal for both the prime part and overflow part, the number of records in a page is also equal.
*2 Fractional digits are truncated.
*3 423 was rounded up from (30000/71)
*4 The number of pages becomes a power of two.
*5 85 was rounded up from (30000 × 0.2/71).
*6 The amount of space is changed to be an integral multiple of the page size.

Example 3:

An example of a space size estimate for an OBJECT structure.

Calculate the required space size for the PRODPHOT table (total number of records: 3000), which is the following structure:

```
CREATE TABLE STOCKS. PRODPHOT (
              ITMNO      SMALLINT      PRIMARY KEY  NOT NULL,
              PRODPHOTO BLOB( 500K ) NOT NULL)
```

Storage structure:
    OBJECT structure
Page size:
    32 kilobytes
Safety factor:
    1.3

```
logical-record-length          = ITMNO + PRODPHOTO
                               = 2 + 512000
                               = 512002

storage-record-length          = page-size - 144
                               = 32768 - 144
                               = 32624

number-of-pages-               = logical-record-length/storage-record-length
necessary-for-one-logical-     = 512002 / 32624
record                         = 15.7          Note: The fractional digit is rounded up.
                               ≒ 16

amount-of-space                = (number-of-pages-necessary-for-one-logical-record × total-
                                   number-of-records + 1) × page-size × safety-factor
                               = (16 × 3000 + 1) × 32768 × 1.3
                               = (48001) × 32768 × 1.3
                               = 1996841.6 (kilobytes)
                               ≒ 1950 (megabytes)
```

Example 4:

Estimate of the required amount of space for a BTREE structure

Calculate the required amount of space when an index is added for the ORDER table having the following kind of structure. PRICE and ORDERQTY is used as the secondary key for the index. The ORDER table has a SEQUENTIAL structure, with a total of 30,000 records. The table structure is as follows:

```
CREATE TABLE  STOCKS.ORDER(
                           CUSTOMER   SMALLINT   NOT NULL,
                           PRODNO     SMALLINT   NOT NULL,
                           PRICE      INTEGER,
                           ORDERQTY   SMALLINT,
                           PRIMARY KEY (CUSTOMER, PRODNO)
                          )
```

Storage structure:
    BTREE structure with PRICE and ORDERQTY as a secondary key
Page size (data part):
    16 kilobytes
Page size (index part):
    2 kilobytes
Utilization rate (data part):
    0.5
Utilization rate (index part):
    0.5
Compression rate (data part):
    0.3
Compression rate (index part):
    0.5

[Data part]

entry-size	= (PRICE + ORDERQTY) × (1 - key-compression-rate) + 20
	= (4 + 2) × (1 - 0.3) + 20
	≒ 25 (*1)
section-size	= entry-size × 10 + 2
	= 25 × 10 + 2
	= 252
number-of-sections-in-a-page	= (data-part-page-size - 110)/section-size
	= (16384 - 110) / 252
	≒ 64 (*2)
number-of-entries-in-a-page	= (section-size/entry-size) × number-of-sections-in-a-page × data-part-utilization-rate
	= (252/25) × 64 × 0.5
	≒ 10 × 64 × 0.5 (*3)
	= 320
number-of-data-part-pages	= number-of-table-records/number-of-entries-in-a-page
	= 30000 / 320
	= 93.75
	≒ 94 (*1)
amount-of-data-part-space	= (number-of-data-part-pages + 1) × data-part-page-size × safety-factor
	= (94 + 1) × 16384 × 1.3
	= 2023424 (bytes)
	= 1976 (kilobytes)
	⇒ 1984 (kilobytes) (*4)

*1 Fractional digits are rounded up.
*2 Fractional digits are truncated.
*3 (252/25) is truncated.
*4 The amount of space is changed to be an integral multiple of the page size.

[Index part]

entry-size = (PRICE + ORDERQTY + 12) × (1 - key-compression-rate) + 10
= (4 + 2 + 12) × (1 - 0.5) + 10
= 19

section-size = entry-size × 10 + 2
= 19 × 10 + 2
= 192

number-of-sections-in-a-page = (index-part-page-size - 110)/section-size
= (2048 - 110) / 192
$\fallingdotseq$ 10 (*1)

number-of-entries-in-a-page = (section-size/entry-size) × number-of-sections-in-a-page
× index-part-utilization-rate
= (192 / 19) × 10 × 0.5
$\fallingdotseq$ 10 × 10 × 0.5
= 50 (*2)

index level = L $\Rightarrow$ Value satisfies the following condition:
(number-of-entries-in-a-page)^{L} $\geq$ number-of-data-part-pages
$(50)^{L} \geq 94$
= 2

number-of-index-part-pages = $\sum_{i=1}^{\text{Index level}}$ (number-of-data-part-pages/number-of-entries-in-a-page)
= $\sum_{i=1}^{2}$ (94/50)
= (94 / 50) + (94 / 50 / 50)
$\fallingdotseq$ 2 + 1 (*3)
= 3

amount-of-index-part-space = (number-of-index-part-pages + 1) × index-part-page-size × safety-factor
= (3 + 1) × 2048 × 1.3
= 10649.6 (bytes)
$\fallingdotseq$ 11 (kilobytes)
$\Rightarrow$ 12 (kilobytes) (*4)

*1 Fractional digits are rounded up.
*2 (192/19) is truncated.
*3 Fractional digits are rounded up.
*4 The amount of space is changed to be an integral multiple of the page size.

■Estimating column length

Column data attribute		Length (bytes)	
Fixed length	CHARACTER(n)	n	
	NUMERIC(p,q)	j	(*1)
	DECIMAL(p,q)	j	(*1)
	SMALLINT	2	
	INTEGER	4	
	REAL	4	
	DOUBLE PRECISION	8	
	TIMESTAMP	7	
	DATE	4	
	TIME	3	
	INTERVAL YEAR(p)	m	(*2)
	[TO MONTH]	m+1	(*2)
	INTERVAL MONTH(p)	m	(*2)
	INTERVAL DAY(p)	m	(*2)
	[{ TO HOUR |	m+1	(*2)
	TO MINUTE |	m+2	(*2)
	TO SECOND }]	m+3	(*2)
	INTERVAL HOUR(p)	m	(*2)
	[{ TO MINUTE |	m+1	(*2)
	TO SECOND }]	m+2	(*2)
	INTERVAL MINUTE(p)	m	(*2)
	[TO SECOND]	m+1	(*2)
	INTERVAL SECOND(p)	m	(*2)
Variable length	CHARACTER VARYING(n)	a	(*3)
	BLOB(nk)	b × 1024 + 6	(*4)

*1 j depends on the multiple of p as follows:

Value of p (Precision)	j
1～ 2	1
3～ 4	2
5～ 6	3
7～ 9	4
10～11	5
12～14	6
15～16	7
17～18	8

*2 m depends on the multiple of p as follows:

Value of p	m
1 ～2	2
3 ～4	3
5 ～9	5

*3 a is the average number of characters. The maximum number of characters is n.
*4 b is the average data length (number of bytes). The maximum length is n kilobytes.

Appendix A Quantitative Restrictions

Table: Quantitative limitations shows quantitative restrictions on SymfoWARE/RDB.

[Table: Quantitative limitations]

Item			Quantitative restriction
Number of databases			Not restricted
Length of names	Alphanumeric characters	Database name	Up to 36 characters (*2)
		Schema name	Up to 36 characters (*1)(*2)
		Table name	Up to 36 characters (*1)(*2)
		Column name	Up to 36 characters (*2)
		Index name	Up to 36 characters (*1)(*2)
		DSO name	Up to 36 characters (*2)
		DSI name	Up to 36 characters (*2)
		Database space name	Up to 36 characters (*2)
		Cursor name	Up to 36 characters (*2)
		Correlation name	Up to 36 characters (*2)
		Routine name	Up to 36 characters (*2)
		Scope name	Up to 36 characters (*2)
		Trigger name	Up to 36 characters (*2)
		SQL statement identifier	Up to 36 characters
		SQL variable name	Up to 36 characters (*2)
		Statement label	Up to 36 characters (*2)
		Parameter name	Up to 36 characters (*2)
		Authorization identifier	Up to 36 characters
		Descriptor name	Up to 36 characters
		Connection name	Up to 36 characters
		SQL server name	Up to 36 characters
		Sequence name	Up to 36 characters
		Role name	Up to 36 characters

Item				Quantitative restriction
Schema elements	Number of base tables per schema			Unrestricted
	Number of columns per table			Maximum 32,000
	Table row length	SEQUENTIAL structure (including BLOB-type column)		Maximum 2 gigabytes
		SEQUENTIAL structure (not including BLOB-type column) or RANDOM structure		Maximum 32,000 bytes
		OBJECT structure		Maximum 2 gigabytes + 32,000 bytes
	Number of columns in unique constraint			Maximum 64 columns
	Data length in unique constraint			Maximum 1,000 bytes
	Data size of default value option			Maximum 3,000 bytes (*3)
	Capacity per table			Petabyte order
	Comment definition			256 bytes
	Length of character string for search condition in trigger definition statement			Maximum 15,000 bytes
	Length of character string in triggered SQL statement of the trigger definition statement			Maximum 15,000 bytes
	Length of character string in the view definition statement for a query specification			Maximum 30,000 bytes
Storage structure configuration elements	Base table	Number of DSO per schema		Unrestricted
		Number of DSI		Unrestricted
		Number of DSI per DSO		Unrestricted
		Data size per fixed length character string type for specifying split condition	CHARACTER	Maximum 254 characters
		Number of columns in key		Maximum 64 columns
		Cluster key length		Maximum 1,000 bytes
		Page length (BLOB of 31 kilobytes or less)		1, 2, 4, 8, 16, or 32 kilobytes
		Page length (BLOB of 32 kilobytes or more)		32 kilobytes

Item				**Quantitative restriction**
Storage structure configuration elements	Indexes	Number of DSO		Unrestricted
		Number of DSI		Unrestricted
		Number of DSI per DSO		Unrestricted
		Number of columns in key		Maximum 64 columns
		Length of key		Maximum 1,000 bytes
		Page size		1, 2, 4, 8, 16, or 32 kilobytes
Data types and attributes that can be handled (COBOL)	Character		Fixed length	32,000 characters
			Variable length	32,000 characters
	Numbers	External decimal representation		Maximum 18 columns
		Internal binary representation	2 byte	Maximum 4 columns
			4 byte	Maximum 9 columns
		Internal decimal representation		Maximum 18 columns
		Floating point representation	4 byte	Up to 6 columns in mantissa
			8 byte	7 or more columns in mantissa
Data manipulation statements	Number of value expressions specified in option list			Maximum 32,000
	Number of tables specified in FROM clause of table expression			Maximum 64
	Number of value expressions specified in GROUP BY clause			Maximum 32,000
	Number of value expressions specified in ORDER BY clause			Maximum 32,000
	Number of cursors processed simultaneously per one session			Unrestricted
	Simultaneous connections per one session			Maximum 16
	Size of one SQL statement character string		Static SQL	Unlimited.
			Dynamic SQL	Maximum 32 kilobytes

*1 Up to eight characters can be specified in a simplified storage structure definition. However, if DEFAULT_DSI_NAME=CODE is specified in the system operating environment file, up to 36 alphanumeric characters can be specified.

*2 A product used together with SymfoWARE may not be able to handle an identifier with a length of more than 18 characters. In this event, NAME_SIZE_CHECK=YES can be specified in the system operating environment file to perform a check so that no resource name can be defined with 19 characters or more. An error is returned if a defined resource name has 19 characters or more.

*3 If the default value option is specified with a character string definition, quotation marks in the data count as 2 bytes.

Appendix B Sequential Relationships among Definition Changes

Definition changes must follow an order determined by definition dependencies. Table: Basic sequential relationships among addition-type definition changes (1/2) shows basic sequential relationships among addition-type changes. Table: Basic sequential relationships among addition-type definition changes (2/2) shows basic sequential relationships among deletion-type changes.

[Table: Basic sequential relationships among addition-type definition changes (1/2)]

Definition change operations / Prerequisite operations	Add database space	Add schema definition	Add sequence definition	Add table definition	Alter table definition (add column definition)	Add view definition	Add trigger definition	Add procedure routine definition
Define database	1	1	1	1	1	1	1	1
Define database space								
Define schema			2	2	2	2	2	2
Define sequence				3		3	3	3
Define table					3	4 (*1)	4 (*1)	4 (*1)
Define view						5 (*1)	5 (*1)	5 (*1)
Define trigger								
Define procedure routine							6 (*1)	6 (*1)
Define function routine						3	3	3
Define privilege information								
Define table DSO								
Define table DSI								
Define index DSO								
Define index DSI								
Define scope								

Definition change operations / Prerequisite operations	Add function routine definition	Add privilege information definition	Add table DSO definition	Add table DSI definition	Add index DSO definition	Add index DSI definition	Add scope definition	Apply scope
Define database	1	1	1	1	1	1	1	1
Define database space				4		5		
Define schema	2	2	2	2	2	2		
Define sequence		3						
Define table		4 (*)	3	3	3	3		
Define view		5 (*)						
Define trigger		7 (*)						
Define procedure routine		6 (*)						
Define function routine		3						
Define privilege information								
Define table DSO				4	4	4		
Define table DSI						5		
Define index DSO						5		
Define index DSI								
Define scope								2

*1 When prerequisite resources are specified in the definition statement

Remarks:
The numbers indicate the sequence of operations. Among operations with the same sequence number, they can be performed in any sequence.

[Table: Basic sequential relationships among addition-type definition changes (2/2)]

Definition change operations / Prerequisite operations	Delete database name registration	Delete database space	Delete schema definition	Delete sequence definition	Delete table definition	Alter table definition (delete column definition) (*1)	Delete view definition	Delete trigger definition	Delete procedure routine definition
Delete database space	3								
Delete schema definition	6								
Delete sequence definition	5 (*2)		5 (*2)						
Delete table definition	4		4	4					
Delete view definition	3 (*2)		3 (*2)	3 (*2)	3 (*2)	3 (*2)	3 (*2)		
Delete trigger definition	1		1	1	1	1	1		1
Delete procedure routine definition	2 (*2)		2 (*2)	2 (*2)	2 (*2)	2 (*2)	2 (*2)		2 (*2)
Delete function routine definition	4 (*2)		4 (*2)						
Delete privilege information definition									
Delete table DSO definition	3		3		3	3			
Delete table DSI definition	2	2	2		2	2			
Delete index DSO definition	2		2		2	2			
Delete index DSI definition	1	1	1		1	1			
Delete scope definition	1								

Definition change operations / Prerequisite operations	Delete function routine definition	Delete privilege informa-tion defini-tion (*3)	Delete table DSO definition	Delete table DSI definition	Delete index DSO definition	Delete index DSI definition	Delete scope definition	Release scope
Delete database space								
Delete schema definition								
Delete sequence definition								
Delete table definition								
Delete view definition	3 (*2)	3 (*2)						
Delete trigger definition	1	1						
Delete procedure routine definition	2 (*2)	2 (*2)						
Delete function routine definition								
Delete privilege information definition								
Delete table DSO definition								
Delete table DSI definition			2					
Delete index DSO definition			2					
Delete index DSI definition			1	1	1			
Delete scope definition								

*1 When the target column is referenced in the definition statement
*2 When prerequisite resources are specified in the definition statement
*3 When a prerequisite privilege is to be deleted

Remarks:
The numbers indicate the sequence of operations. Among operations with the same sequence number, they can be performed in any sequence.

Appendix C Operating Environment File Parameters

This appendix lists parameters that can be specified in operating environment files and the files in which the parameters can be specified. Table: Operating environment file parameters shows the operating environment file parameters.

[Table: Operating environment file parameters]

Type	Execution parameter	Overview	Number of specifications	Can be omitted?	SY	SV	CL
Communica-tion	BUFFER_SIZE	Size of a buffer used for communication	One	Yes			○
	CLUSTER_SERVICE_NAME	Information necessary for executing an application program in hot standby mode	One	Yes			○
	COMMUNICATION_BUFFER	Size of a buffer used for local access on a server	One	Yes	○		
	DEFAULT_CONNECTION	Default server connection information	One	No, when DEFAULT is specified in the CONNECT statement			○
	MAX_CONNECT_SYS	Maximum number of local connections that can be established	One	Yes	○		
	MAX_CONNECT_TCP	Maximum number of connections that can be established via RDB2_TCP to one SymfoWARE/RDB system	One	Yes	○		
	RDB_KEEPALIVE	Whether to use the KEEPALIVE function for connection to SymfoWARE/RDB via RDB2_TCP	One	Yes	○		
	SERVER_ENV_FILE	Name of a server operating environment file to be used	Multiple specifications permitted	Yes			○
	SERVER_SPEC	Information for communication with a remote server	Multiple specifications permitted	No, for access to a remote database. Cannot be specified for local access.			○
	TRAN_SPEC	Transaction in the event of an SQL error	One	Yes			○
	WAIT_TIME	Communication wait time	One	Yes			○
Work area	DESC_NUM (*1)	Maximum number of columns available for an application program	One	Yes			○
	DESCRIPTOR_SPEC	SQL descriptor information for dynamic SQL	One	Yes			○
	MAX_SQL	Number of SQL statements that can be processed concurrently	One	Yes			○
	OPL_BUFFER_SIZE	Size of a buffer for storing an SQL statement execution procedure	One	Yes			○
	RESULT_BUFFER	Number of buffers and the size of each buffer for performing a batch FETCH	One	Yes			○
	SORT_MEM_SIZE	Size of memory used as a sort work area	One	Yes	○	○	○
	WORK_ALLOC_SPACESIZE	Size of a file used as a work table or sort work area	One	Yes		○	○
	WORK_MEM_SIZE	Size of memory used as a work table	One	Yes	○	○	○
	WORK_PATH	Path to a work table or sort work area	One	Yes	○	○	○

Type	Execution parameter	Overview	Number of specifications	Can be omitted?	SY	SV	CL
Character data processing	CAL_ERROR	Processing performed if an overflow occurs in assignment processing	One	Yes			○
	CHARACTER_ TRANSLATE	Whether to convert the character code on a client	One	Yes			○
	CHAR_SET	Character code used in an application program	One	Yes			○
	NCHAR_CODE	Japanese character code used in an application program	One	Yes			○
	RDA_CONV_CODE (*1)	Character code set	One	Yes			○
Table and index	DEFAULT_DSI_NAME	Method of generating the DSO name and DSI name of a table or index. SymfoWARE/RDB automatically generates the names if a table or index with a simplified storage structure definition is defined.	One	Yes	○		
	DEFAULT_DSI_TYPE	Selection of the storage structure of the DSO of a table. SymfoWARE/RDB automatically generates the DSO if a table with a simplified storage structure definition is defined.	One	Yes	○		○
	DEFAULT_INDEX_ SIZE	Initial size, expansion size, page length, and other items of the index data storage area when an index with a simplified storage structure definition is defined	One	Yes	○	○	○
	DEFAULT_OBJECT_ TABLE_SIZE	Initial size, expansion size, page length, and other items of the OBJECT structure table data storage area when a table with a simplified storage structure definition is defined	One	Yes	○	○	○
	DEFAULT_TABLE_ SIZE	Initial size, expansion size, page length, and other items of the table data storage area when a table with a simplified storage structure definition is defined	One	Yes	○	○	○
	DSI_EXPAND_POINT	Whether to activate DSI expansion	One	Yes			○
	INCLUSION_DSI	Limitations on DSIs used by an application program	One	Yes		○	○
	INDEX_PREFIX	Prefix of the DSO name and DSI name of an index. SymfoWARE/RDB automatically generates the prefix if an index with a simplified storage structure definition is defined.	One	Yes	○		
	TABLE_PREFIX	Prefix of the DSO name and DSI name of a table. SymfoWARE/RDB automatically generates the prefix if a table with a simplified storage structure definition is defined.	One	Yes	○		
	TEMPORARY_INDEX_ SIZE	Initial size, expansion size, and other items of the index data storage area when an index is defined for a temporary table	One	Yes	○	○	○
	TEMPORARY_TABLE_ SIZE	Initial size, expansion size, and other items of the table data storage area when a temporary table is defined	One	Yes	○	○	○

Type	Execution parameter	Overview	Number of specifications	Can be omitted?	SY	SV	CL
Exclusive control	DSO_LOCK	DSO lock unit and mode to be used	One	Yes		○	○
	ISOLATION_WAIT	Lock waiting mode	One	Yes			○
	R_LOCK	Use of rows as lock units	One	Yes	○		○
Debugging	COREFILE_PATH (*2)	Dump output destination if an application program error occurs	One	Yes			○
	DIV_TRACE_FILE	Whether to output individual trace information when an application program is running under multiple processes	One	Yes			○
	EXTERNAL_ PROCESS_CORE	Dump output destination if an error occurs in a process not running under SymfoWARE/RDB	One	Yes	○		
	MAX_EXTPROC_ CORE_NUM	Maximum number of dumps output if an error occurs in a process running not under SymfoWARE/RDB	One	Yes	○		
	ROUTINE_SNAP	Whether to output trace data for a stored procedure routine	One	Yes			○
	SQL_SNAP	Whether to use the SQL_SNAP function	One	Yes			○
Access plans and performance information (*3)	ACCESS_PLAN	Whether to obtain an access plan for each application program	One	Yes			○
	IGNORE_INDEX	Whether to select an access plan that uses no index	One	Yes	○	○	○
	JOIN_RULE	Join method	One	Yes	○	○	○
	SORT_HASHAREA_ SIZE	Size of the area for hashing and storing records by sort processing	One	Yes	○	○	○
	SQL_TRACE	Whether to obtain the SQL performance information of each application program	One	Yes			○
	SS_RATE	Selection rate of the retrieval range for each predicate	One	Yes	○	○	○
	TID_SORT	Whether to use access model TID sorting to retrieve an index and obtain table data	One	Yes	○	○	○
	TID_UNION	Whether to enable the access model for TID union merge	One	Yes	○	○	○
	USQL_LOCK	Lock mode of the section where a record to be updated by the UPDATE statement (search) or DELETE statement (search) is located	One	Yes	○	○	○
Process control	EXTERNAL _PROCESS_USER	Specification of the ID of the user who executes a process not running under SymfoWARE/RDB	One	Yes	○		
Messages	CONSOLE_MSG	Whether to output messages to the console	One	Yes	○		
	MSG_LANG (*2)	Language of messages to be displayed	One	Yes			○
	MSG_PRINT	Whether to display an error message at execution of an SQL statement	One	Yes			○

Type	Execution parameter	Overview	Number of specifications	Can be omitted?	SY	SV	CL
Recovery	RCV_MODE	Specification of the recovery level for an application program	One	Yes			○
Reserved word	SQL_LEVEL	Reserved word level for an application program	One	Yes			○
Parallel query	MAX_PARALLEL	Multiplicity when a database is searched in parallel	One	Yes	○	○	○
	PARALLEL_SCAN	Whether to search a database in parallel in application or connection units	One	Yes		○	○
Others	ALTER_CHECK (*4)	Whether to check if a change to table rows because of a table design change affects an application program	One	Yes			○
	ARC_FULL	Whether to return an error message if the archive log is full	One	Yes	○		○
	NAME_SIZE_ CHECK	Whether to check the size of a resource name	One	Yes	○		
	SIGNAL_INF (*4)	Whether to use signals in an application program	One	Yes			○

SY: Indicates whether the parameter can be specified in a system operating environment file.
SV: Indicates whether the parameter can be specified in a server operating environment file.
CL: Indicates whether the parameter can be specified in a client operating environment file.

*1 Can be specified only for RDA connection.
*2 Can be specified only when Windows is used.
*3 They are execution parameters related to access plans and performance information.
For more information, refer to the "SQLTOOL User's Guide."
*4 Can be specified only when UNIX is used.

Appendix D Environment Variables

At compilation, and link-editing, and execution of application programs, the user can specify tuning the operating environment with environment variables.

For the specification formats and meanings of the environment variables, refer to the "RDB User's Guide: Application Program Development."

■Environment variables specified at compilation and link-editing of application programs

· LANG (UNIX only)
· RDBDB
· SQLPC
· SQLPCOB
· INCDIR (UNIX only)
· INCLUDE (Windows NT/2000/XP only)

■Environment variables specified at execution of application programs

· LD_LIBRARY_PATH_64
· LD_LIBRARY_PATH
· LD_PRELOAD
· SHLIB_PATH
· RDBNAME
· SQLRTENV

In addition, there are environment variables related to the execution parameters of the operating environment file for application programs. The operating environment file is used to tune the operating environment of application programs; however, a part of environment can be specified with environment variables. The parameters specified in those environment variables are also valid in the rdbupt command.

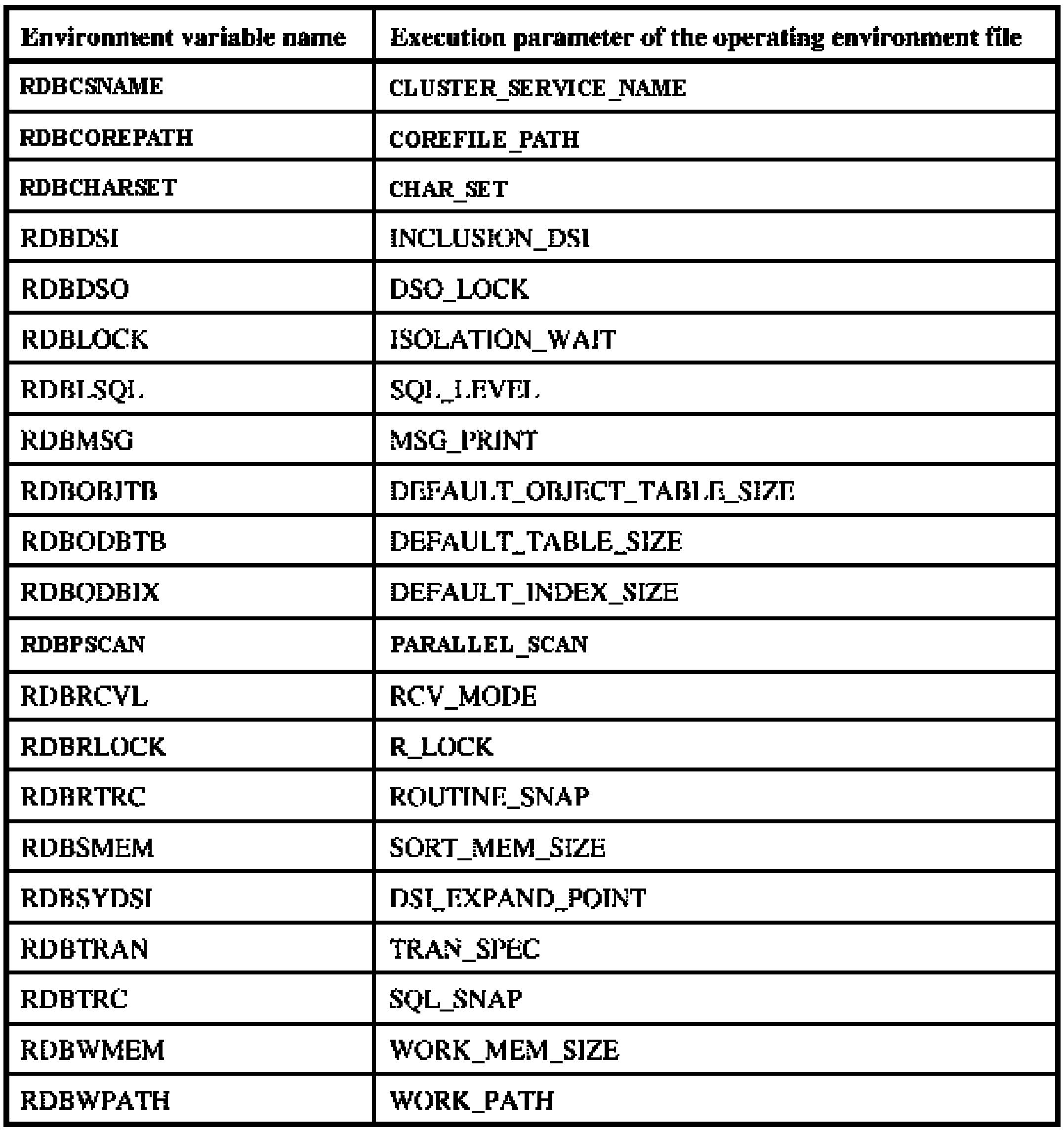
■Operating environment tuning priority

If a specification resulting from an environment variable duplicates a specification resulting from an operating environment file, the former specification takes precedence.

■Correspondence between environment variables and execution parameters of the operating environment file

Table: Environment variables specified upon execution of the application program and execution parameters of the operating environment file shows the correspondence between environment variables and execution parameters of the operating environment file.

[Table: Environment variables specified upon execution of the application program and execu tion parameters of the operating environment file]



Environment variable name	Execution parameter of the operating environment file
RDBCSNAME	CLUSTER_SERVICE_NAME
RDBCOREPATH	COREFILE_PATH
RDBCHARSET	CHAR_SET
RDBDSI	INCLUSION_DSI
RDBDSO	DSO_LOCK
RDBLOCK	ISOLATION_WAIT
RDBLSQL	SQL_LEVEL
RDBMSG	MSG_PRINT
RDBOBJTB	DEFAULT_OBJECT_TABLE_SIZE
RDBODBTB	DEFAULT_TABLE_SIZE
RDBODBIX	DEFAULT_INDEX_SIZE
RDBPSCAN	PARALLEL_SCAN
RDBRCVL	RCV_MODE
RDBRLOCK	R_LOCK
RDBRTRC	ROUTINE_SNAP
RDBSMEM	SORT_MEM_SIZE
RDBSYDSI	DSI_EXPAND_POINT
RDBTRAN	TRAN_SPEC
RDBTRC	SQL_SNAP
RDBWMEM	WORK_MEM_SIZE
RDBWPATH	WORK_PATH

Appendix E RDB Command Summary

This appendix gives an overview of RDB commands and functions. For more information about the syntax of the RDB commands, refer to the man command (under UNIX) or the SymfoWARE/RDB online manual (under Windows NT).

■RDB Command Summary

A list of the RDB commands is as follows:

[Table: RDB command list]

Command	Functional overview
rdbadjrcv	Recovers a database after an input-output fault occurs in a temporary log file.
rdbaldic	Performs additional allocation and capacity extension of RDB dictionary.
rdbalidx	Alters the index degradation specification.
rdbalmdsi	Defines alarm points and automatic extension for DSI.
rdbchksanity	Confirms SymfoWARE/RDB operation.
rdbcninf	Displays the connection and communication environment status.
rdbconbf	Registers mapping between DSI and shared memory pool.
rdbcrbf	Opens shared buffer having specified page size and number of pages.
rdbcrdic	Creates RDB dictionary.
rdbddlex	Creates database from database definition file.
rdbdisbf	Releases mapping between DSI and shared buffer pool.
rdbdmp	Saves data for each DSI into database in external file.
rdbdmpdic	Writes the data of the RDB dictionary to an external file.
rdbdrbf	Closes shared buffer of page size in specified shared buffer.
rdbdvinf	Displays information about the save medium.
rdbexdsi	Releases DSI from SQL processing or cancels exclusion.
rdbexecsql	Interactively tunes database performance.
rdbexspc	Connects or disconnects a SymfoWARE/RDB database space.
rdbfmt	Initializes DSI.
rdbgcdic	Reorganizes RDB dictionary.
rdbgcdsi	Reorganizes specified DSI.
rdbhsrsc	Registers, changes, deletes, and displays resources in hot-standby mode.
rdbhsuty	Displays the hot-standby declaration and pre-open information.
rdbinf	Displays operating information such as resource information, update inhibit information, and access inhibit information related to DSI, database space, or RDB dictionary.
rdbinh	Puts specified resource in access inhibit state.
rdblkinf	Displays information about the locked resource.
rdblog	Creates and initializes temporary log file and archive log file.
rdbpldic	Places table definition information into memory or deletes it from memory.
rdbpmt	Releases access inhibit status for specified resource.
rdbprdic	Outputs use status of RDB dictionary area to standard output.

Command	Functional overview
rdbprt	Outputs information about database definition or list of defined database names to standard output.
rdbprtbf	Displays list of shared buffer identifiers, shared buffer pool information, and information on mapping between shared buffer pool and DSI.
rdbprtmsg (*1)	Displays an explanation of the specified message number or the action to be taken at the occurrence of a SymfoWARE/RDB internal error.
rdbprxid	Displays a XID list of in doubt transactions under XA.
rdbps	Displays execution status of the application program or RDB command.
rdbrcv	Performs database resource recovery based on save data and data of database space.
rdbrcvdic	Performs recovery of RDB dictionary and RDB directory.
rdbrls	Releases recovery level switch or update inhibit, read-write inhibit, and other usage restrictions for specified resource.
rdbrtr	Registers recovery level switch or update inhibit, read-write inhibit, and other usage restrictions for specified resource.
rdbsaloader	Adds data to DSI from data of an external file.
rdbsar	Displays SymfoWARE/RDB performance information.
rdbscldir	Creates, displays, and deletes RDB directory file for a user log group.
rdbsetrp	Sets recovery point.
rdbsloader	Creates table DSI and all index DSI if indexes are defined from data in input file.
rdbspcinf	Outputs use status of areas of database space.
rdbstart	Activates SymfoWARE/RDB system.
rdbstop	Stops SymfoWARE/RDB system.
rdbsuloader	Adds, replaces, updates, and deletes DSI data from an external file.
rdbterm	Recovers a lost connection.
rdbudsi	Outputs use status of areas allocated to DSI.
rdbunl	Outputs table or table DSI data to file.
rdbups	Sets optimization information to use in SQL in RDB dictionary and outputs to standard output.
rdbupt	Adds, updates, or deletes table data in an input file.
rdbxarcv	Recovers a failed in doubt transaction under XA.
sqlpcob	Precompiles COBOL program with embedded SQL.
sqlpc	Precompiles C program with embedded SQL.
sqlcc (*2)	Compiles and link-edits a SQL-embedded C program.
sqlcobol (*2)	Compiles and link-edits a SQL-embedded COBOL program.

*1 Executable only under UNIX.
*2 Available under Windows NT; supplied as a command of a standard shell procedure under UNIX.

■Standard shell procedure

Table: Standard shell procedures lists the standard shell procedures used under UNIX.

[Table: Standard shell procedures]

Command	Function outline
sqlcc	Compiles and join-edits an SQL embedded C program.
sqlcobol	Compiles and join-edits an SQL embedded COBOL program.
sqlfcc	Compiles and join-edits an SQL embedded C program using the Fujitsu C compiler.

■Notes on using RDB commands

◆Handling of uppercase and lowercase letters

In RDB command specifications, lowercase letters are distinguished from uppercase letters. In SQL, an ordinary identifier (not enclosed in double quotes) is handled by converting lowercase letters to the corresponding uppercase letters. Therefore, when an identifier defined using lowercase letters in ordinary identifier format is specified in an RDB command, it must be specified using uppercase letters. In addition, lowercase letters in a delimited identifier (enclosed in double quotes) ate handled unchanged in SQL. Therefore, when an identifier defined using lowercase letters in delimited identifier format is specified in an RDB command, it must be specified using lowercase letters.

Example:

Example of specification using uppercase letters (specifying ordinary identifier as DSI name in SQL statement)

Table DSI definition

```
create dsi dsi001 dso dso001 allocate ...
```

RDB command

```
rdbfmt -mi -i DB01.DSI001
```

Example:

Example of specification using lowercase letters (specifying delimited identifier as DSI name in SQL statement)

Table DSI definition

```
create dsi "dsi001" dso dso001 allocate ...
```

RDB command

```
rdbfmt -mi -i DB01.dsi001
```

◆Handling characters that have special meaning in the shell

In UNIX system SQL, identifiers may contain #, ¥, and @. In the shell, these characters are treated as symbols having special meanings. # is the beginning of a comment in a shell script. ¥ is treated as an escape character or line continuation symbol. @ is used in the line deletion function. Therefore, when an identifier defined using these characters is specified in an RDB command, the following condition applies. The special meanings of the characters according to shell rules must be canceled. To cancel the special meaning of a character, enclose the character string in quotes ('), or specify an escape character (¥) just before the character.

Example:

Example of enclosing character string in quotes (') (specifying DSI name containing # in SQL statement)

Table DSI definition

```
create dsi dsi#001 dso dso001 allocate ...
```

RDB command

```
rdbfmt -mi -i DB01.'DSI#001'
```

Example:

Example of specifying escape character (¥) just before character string (specifying DSI name containing @ in SQL statement)

Table DSI definition

```
create dsi @DSI001 dso dso001 allocate ...
```

RDB command

```
rdbfmt -mi -i DB01.\@DSO001
```

Some characters that have special meanings can be changed by shell functions. The results must be considered in the user environment setup if those changes are being made.

◆Handling of reserved words in SQL

When a reserved word is used in an identifier in SQL, it must be specified in a delimited identifier (enclosed in double quotes). However, even if an identifier in an RDB command is an SQL reserved word, it can be specified without changing the format.

Example:

Example of specifying without changing format (specifying reserved word as DSI name in SQL statement)

Table DSI definition

```
create dsi "SELECT" dso dso001 allocate ...
```

RDB command

```
rdbfmt -mi -i DB01.SELECT
```

Appendix F Handling SymfoWARE/RDB Messages

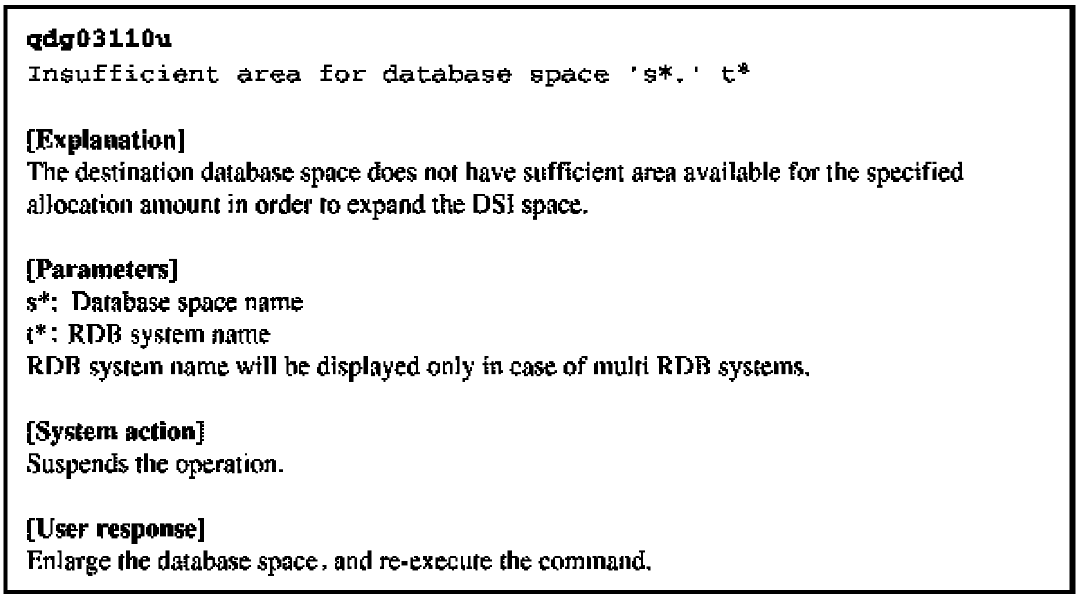
Refer to the online manual "SymfoWARE/RDB Message Reference" for the user handling of the following SymfoWARE/RDB output messages:

- Message issued when an RDB command is executed
- Message issued when an application program is compiled
- Message set in the message variable (SQLMSG) when an application program is compiled
- Message output when SymfoWARE/RDB terminates abnormally on detecting an internal inconsistency

■When executing an RDB command

Example 1:

Display a description of the qdg03110u message issued by executing an RDB command. (SymfoWARE Server Enterprise Edition)



qdg03110u

```
Insufficient area for database space 's*.' t*
```

[Explanation]
The destination database space does not have sufficient area available for the specified allocation amount in order to expand the DSI space.

[Parameters]
s*: Database space name
t*: RDB system name
RDB system name will be displayed only in case of multi RDB systems.

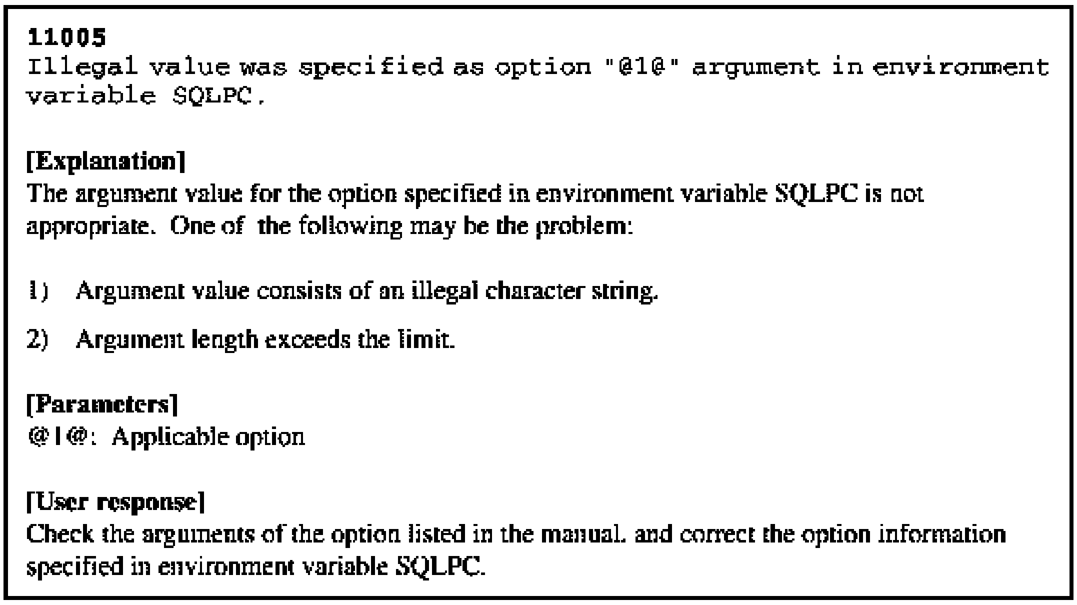
[System action]
Suspends the operation.

[User response]
Enlarge the database space, and re-execute the command.

■When compiling an application program (C language)

Example 2:

Display a description of message 11005 issued by the C precompiler.



```
11005
Illegal value was specified as option "@1@" argument in environment
variable SQLPC.
```

[Explanation]
The argument value for the option specified in environment variable SQLPC is not appropriate. One of the following may be the problem:

1) Argument value consists of an illegal character string.
2) Argument length exceeds the limit.

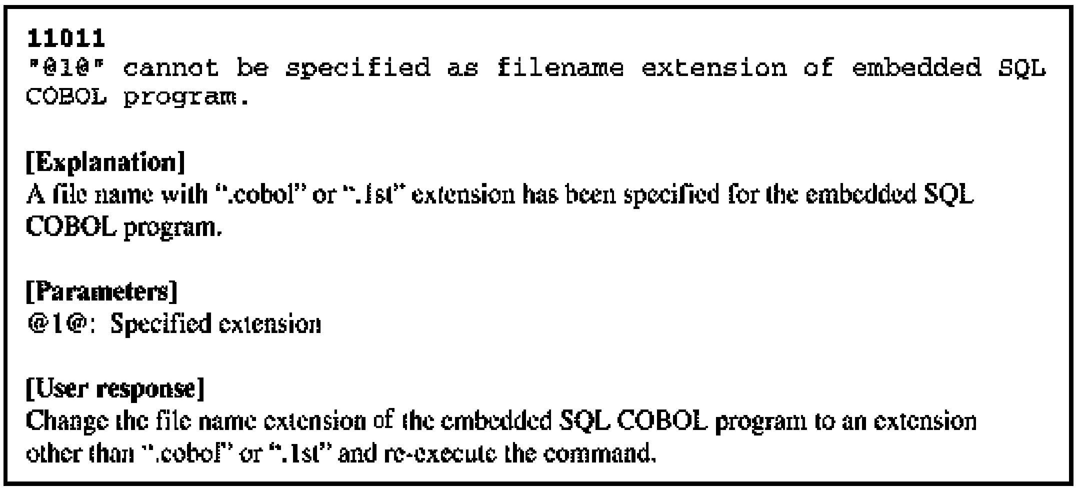
[Parameters]
@1@: Applicable option

[User response]
Check the arguments of the option listed in the manual. and correct the option information specified in environment variable SQLPC.

■When compiling an application program (COBOL)

Example 3:

Display a description of message 11011 issued by the COBOL precompiler.



```
11011
"@1@" cannot be specified as filename extension of embedded SQL
COBOL program.
```

[Explanation]
A file name with ".cobol" or ".lst" extension has been specified for the embedded SQL COBOL program.

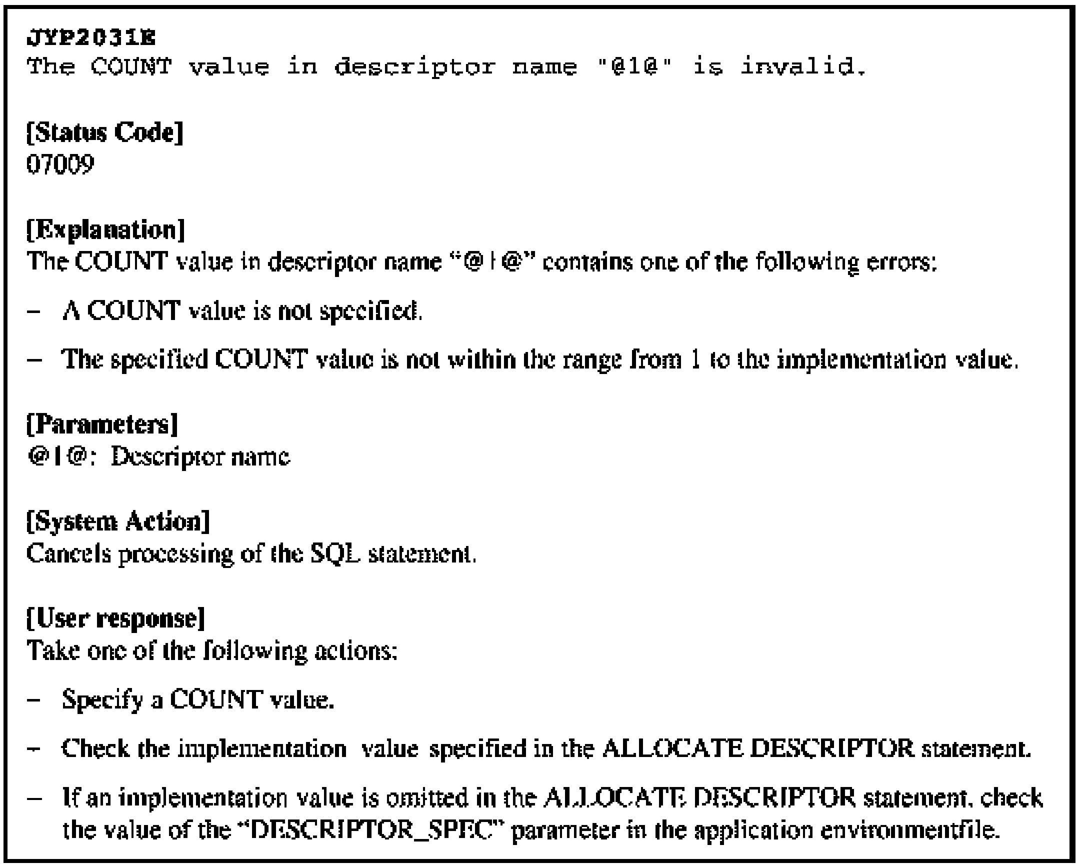
[Parameters]
@1@: Specified extension

[User response]
Change the file name extension of the embedded SQL COBOL program to an extension other than ".cobol" or ".lst" and re-execute the command.

■When executing an application program

Example 4:

Display a description of message JYP2031E set in the message variable (SQLMSG) when executing an application program.



```
JYP2031E
The COUNT value in descriptor name "@1@" is invalid.
```

[Status Code]
07009

[Explanation]
The COUNT value in descriptor name "@1@" contains one of the following errors:

- A COUNT value is not specified.
- The specified COUNT value is not within the range from 1 to the implementation value.

[Parameters]
@1@: Descriptor name

[System Action]
Cancels processing of the SQL statement.

[User response]
Take one of the following actions:

- Specify a COUNT value.
- Check the implementation value specified in the ALLOCATE DESCRIPTOR statement.
- If an implementation value is omitted in the ALLOCATE DESCRIPTOR statement, check the value of the "DESCRIPTOR_SPEC" parameter in the application environmentfile.

■When SymfoWARE/RDB has detected an internal inconsistency

Example 5:

Display a description of message qdg12695u issued when SymfoWARE/RDB terminates abnormally on detecting an internal inconsistency. (Reference code: 16.1.7.769)

```
qdg12695u: System error occurred. Reference code = 16.1.7.769
Internal_code = (idt_p:008b2394 sub_code:0 module:qdafcomi000
line:136) (SYSTEM NAME =TEST )
```

16.1.7.769

[UserResponse]

Activate SynfoWARE/RDB, and recover from the system failure.

[Investigation Report]

Core dump
  Storage location:
    The core dump is in the directory specified in the RDBCORE parameter of the RDB:configuration parameter file.
  Method of acquisition:
    Save the core dump in secondary storage by executing the ntbackup command, for example.
    Sample operation:
      Execute the ntbackup command, and save the core file by following the menu instructions.

Load module
  Storage location:
    The load module is in the sbin directory under the installation directory.
  Method of acquisition:
    Save the module in secondary storage by executing the ntbackup command, for example.
    Sample operation:
      Execute the ntbackup command, and save the rdb2base.exe file by following the menu instructions.

Library
  Storage location:
    The library is in the lib directory in the installation directory.
  Method of acquisition:
    When the dumpbin command included in VC++ is executed with the specification of rdb2base.exe as the import5 option parameter, save the files that are output to the secondary storage media.
    Sample operation:
    Execute the ntbackup command, and save the library files by following the menu instructions.

Console log
  Storage location:
    The console log is in the directory where the event logs were saved by the event viewer.
  Method of acquisition:
    To save the log to the secondary storage media, select "Save As" from the log menu of the event viewer.

Execution procedure log
  Storage location:
    Not defined
  Method of acquisition:
    Save the file containing the processing records and execution process records in secondary storage.

Appendix G Exclusive Control between Application Programs and RDB Commands

When an application program and RDB command operate simultaneously on the same database resource, the SymfoWARE/RDB system performs exclusive control. In this way, each process can be executed without contradiction.

Each function performs processing by applying exclusive control of the resource specified by the option or input data. The period for which the resource is exclusively controlled is as follows:

Application program:

    From the time the first SQL statement is issued from the application program until the application program terminates

RDB command:

    From the time the command is entered until the response message is generated

    Table: Locked resources and locked strength for each SymfoWARE/RDB function shows resources for which lock is performed by each function and the strength of the lock. Table: Relationships among lock strengths shows the relationships among locked strengths.

[Table: Locked resources and locked strength for each SymfoWARE/RDB function]

Function	Command		Locked resources													
			Logical						Storage					Physical	User authorization identifier	ROLE
			DB	SCH	SEQ	TBL	RTN	TRG	DSO	DSO (IX)	DSI	DSI (IX)	SCOP	DBS		
Define database	rdbddlex	CREATE DB	EX	-	-	-	-	-	-	-	-	-	-	-	-	-
Delete database	rdbddlex	DROP DB	EX	-	-	-	-	-	-	-	-	-	-	-	-	-
Define schema	rdbddlex	CREATE SCHEMA	SH	EX	-	-	-	-	-	-	-	-	-	-	-	-
Delete schema	rdbddlex	DROP SCHEMA (without CASCADE)	SH	EX	-	-	-	-	-	-	-	-	-	-	-	-
		DROP SCHEMA (with CASCADE)	SH	EX	-	EX	EX	EX	EX	EX	EX	EX	EX	SH	EX	-
Define sequence	rdbddlex	CREATE SEQUENCE	SH	SH	EX	-	-	-	-	-	-	-	-	-	-	-
Delete sequence	rdbddlex	DROP SEQUENCE	SH	SH	EX	-	-	-	-	-	-	-	-	-	-	-
Define table	rdbddlex	CREATE TABLE	SH	SH	-	EX	-	-	-	-	-	-	-	-	-	-
Delete table	rdbddlex	DROP TABLE (without CASCADE)	SH	SH	-	EX	-	-	-	-	-	-	-	-	-	-
		DROP TABLE (with CASCADE)	SH	SH	-	EX	EX	EX	EX	EX	EX	EX	EX	SH	EX	-
Define view	rdbddlex	CREATE VIEW	SH	SH	-	EX	-	-	-	-	-	-	-	-	-	-
Delete view	rdbddlex	DROP VIEW	SH	SH	-	EX	-	-	-	-	-	-	-	-	-	-
Define routine	rdbddlex	CREATE PROCEDURE	SH	SH	-	-	EX	-	-	-	-	-	-	-	-	-
		CREATE FUNCTION	SH	SH	-	-	EX	-	-	-	-	-	-	-	-	-
Delete routine	rdbddlex	DROP PROCEDURE	SH	SH	-	-	EX	-	-	-	-	-	-	-	-	-
		DROP FUNCTION	SH	SH	-	-	EX	-	-	-	-	-	-	-	-	-
Define trigger	rdbddlex	CREATE TRIGGER	SH	SH	-	EX/SH (*1)	-	EX	-	-	-	-	-	-	-	-
Delete trigger	rdbddlex	DROP TRIGGER	SH	SH	-	EX/SH (*1)	-	EX	-	-	-	-	-	-	-	-
Alter table definition	rdbddlex	ALTER TABLE	SH	SH	-	EX	-	-	EX	EX	EX	EX	-	-	-	-

Function	Command		Locked resources													
			Logical						Storage					Physical	User authorization identifier	ROLE
			DB	SCH	SEQ	TBL	RTN	TRG	DSO	DSO (IX)	DSI	DSI (IX)	SCOP	DBS		
Swap tables	rdbddlex	SWAP TABLE	SH	SH	-	EX	-	-	EX	EX	EX	EX	-	-	-	-
Define storage structure	rdbddlex	CREATE DSO	SH	SH	-	EX	-	-	EX	-	-	-	-	-	-	-
		CREATE DSO (IX)	SH	SH	-	EX	-	-	EX	EX	EX	-	-	-	-	-
		CREATE DSI	SH	SH	-	EX	-	-	EX	-	EX	-	EX	SH	EX	-
		CREATE DSI (IX)	SH	SH	-	EX	-	-	EX	EX	SH	EX	-	SH	-	-
		CREATE SCOPE	SH		-	-	-	-	-	-	EX	-	EX	-	-	-
Delete storage structure	rdbddlex	DROP DSO (without CASCADE)	SH	SH	-	EX	-	-	EX	-	-	-	-	-	-	-
		DROP DSO (with CASCADE)	SH	SH	-	EX	-	-	EX	EX	EX	EX	EX	SH	EX	-
		DROP DSO (IX) (without CASCADE)	SH	SH	-	EX	-	-	EX	EX	-	-	-	-	-	-
		DROP DSO (IX) (with CASCADE)	SH	SH	-	EX	-	-	EX	EX	-	EX	-	SH	-	-
		DROP DSI (without CASCADE)	SH	SH	-	EX	-	-	EX	-	EX	-	EX	SH	EX	-
		DROP DSI (with CASCADE)	SH	SH	-	EX	-	-	EX	-	EX	EX	EX	SH	EX	-
		DROP DSI (IX)	SH	SH	-	EX	-	-	EX	EX	SH	EX	EX	SH	EX	-
		DROP SCOPE	SH	-	-	-	-	-	-	-	-	-	EX	-	EX	-
Alter DSI	rdbddlex	ALTER DSI	SH	SH	-	EX	-	-	EX	-	EX	-	-	SH	-	-
Apply scope	rdbddlex	APPLY SCOPE	SH	-	-	-	-	-	-	-	-	-	EX	-	EX	-
Release scope	rdbddlex	RELEASE SCOPE	SH	-	-	-	-	-	-	-	-	-	EX	-	EX	-
Define space	rdbddlex	CREATE DBS	SH	-	-	-	-	-	-	-	-	-	-	EX	-	-
Delete space	rdbddlex	DROP DBS	SH	-	-	-	-	-	-	-	-	-	-	EX	-	-

Function	Command		Locked resources													
			Logical						Storage					Phys ical	User authorization identifier	ROLE
			DB	SCH	SEQ	TBL	RTN	TRG	DSO	DSO (IX)	DSI	DSI (IX)	SCOP	DBS		
Define privilege informa-tion	rdbddlex	GRANT (SCH)	SH	MOD	-	-	-	-	-	-	-	-	-	-	SH	MOD
		GRANT (TBL)	SH	SH	-	MOD-	-	-	-	-	-	-	-	-	SH	MOD
		GRANT (DBS)	SH	-	-	-	-	-	-	-	-	-	-	MOD	SH	MOD
		GRANT (ROUTINE)	SH	SH	-	-	MOD	-	-	-	-	-	-	-	SH	MOD
		GRANT (TRG)	SH	SH	-	-	-	MOD	-	-	-	-	-	-	SH	MOD
		GRANT (SEQUENCE)	SH	SH	MOD	-	-	-	-	-	-	-	-	-	SH	MOD
		GRANT(ROLE)	-	-	-	-	-	-	-	-	-	-	-	-	SH	MOD
Delete privilege informa-tion	rdbddlex	REVOKE (SCH)	SH	MOD	-	-	-	-	-	-	-	-	-	-	SH	MOD
		REVOKE (TBL)	SH	SH	-	MOD	-	-	-	-	-	-	-	-	SH	MOD
		REVOKE (DBS)	SH	-	-	-	-	-	-	-	-	-	-	MOD	SH	MOD
		REVOKE (ROUTINE)	SH	SH	-	-	MOD	-	-	-	-	-	-	-	SH	MOD
		REVOKE (TRG)	SH	SH	-	-	-	MOD	-	-	-	-	-	-	SH	MOD
		REVOKE (SEQUENCE)	SH	SH	MOD	-	-	-	-	-	-	-	-	-	SH	MOD
		REVOKE(ROLE)	-	-	-	-	-	-	-	-	-	-	-	-	SH	MOD
Define user	rdbddlex	CREATE USER	-	-	-	-	-	-	-	-	-	-	-	-	EX	-
Alter user	rdbddlex	ALTER USER	-	-	-	-	-	-	-	-	-	-	-	-	MOD	-
Delete user	rdbddlex	DROP USER	-	-	-	-	-	-	-	-	-	-	-	-	EX	-
Define role	rdbddlex	CREATE ROLE	-	-	-	-	-	-	-	-	-	-	-	-	-	EX
Delete role	rdbddlex	DROP ROLE	-	-	-	-	-	-	-	-	-	-	-	-	-	EX
Specify parameter	rdbddlex	SET SYSTEM PARAMETER	-	-	-	-	-	-	-	-	-	-	-	-	SH	MOD

Function	Command		Locked resources													
			Logical						Storage					Physical		
			DB	SCH	SEQ	TBL	RTN	TRG	DSO	DSO (IX)	DSI	DSI (IX)	SCOP	DBS	User authorization identifier	ROLE
Specify optimiza-tion infor-mation	rdbddlex	SET STATISTICS (TBL)	SH	SH	-	SH	-	-	MOD	MOD	MOD	MOD	-	SH	-	-
		SET STATISTICS (DSO IX)	SH	SH	-	SH	-	-	-	MOD	-	SH	-	SH	-	-
		SET STATISTICS (DSI)	SH	SH	-	SH	-	-	-	-	MOD	-	-	SH	-	-
		SET STATISTICS (DSI IX)	SH	SH	-	SH	-	-	-	-	-	MOD	-	SH	-	-
Display optimiza-tion infor-mation	rdbddlex	PRINT STATISTICS (TBL)	SH	SH	-	SH	-	-	SH	SH	SH	SH	-	SH	-	-
		PRINT STATISTICS (DSO IX)	SH	SH	-	SH	-	-	-	SH	-	SH	-	SH	-	-
		PRINT STATISTICS (DSI)	SH	SH	-	SH	-	-	-	-	SH	-	-	SH	-	-
		PRINT STATISTICS (DSI IX)	SH	SH	-	SH	-	-	-	-	-	SH	-	SH	-	-
Define temporary table	rdbddlex	CREATE GLOBAL TEMPORARY TABLE	SH	SH	-	EX	-	-	-	-	-	-	-	SH	-	-
Define temporary table index	rdbddlex	CREATE INDEX	SH	SH	-	EX	-	-	-	-	-	-	-	SH	-	-
Initialize DSI	rdbfmt	DSI	SH	SH	-	SH	-	-	-	-	SH	SH	-	SH	-	-
		DSI(IX)	SH	SH	-	SH	-	-	-	-	SH	SH	-	SH	-	-
Load data	rdbsloader	DSI	SH	SH	-	SH	-	-	SH	SH	SH	SH	-	SH	-	-
		DSI(IX)	SH	SH	-	SH	-	-	SH	SH	SH	SH	-	SH	-	-
Unload data	rdbunl	DSI	SH	SH	-	SH	-	-	-	-	SH	-	-	SH	-	-
Reorganize database	rdbgcdsi	DSI	SH	-	-	-	-	-	-	-	SH	SH	-	-	-	-
		DSI(IX)	SH	-	-	-	-	-	-	-	SH	SH	-	-	-	-
Update database	rdbupt		SH	SH	-	SH	-	-	SH	SH	SH	SH	SH	-	SH	-
	rdbsaloader		SH	SH	-	SH			SH	SH	SH	SH		SH		
	rdbsuloader		SH	SH	-	SH	-	-	SH	SH	SH	SH	-	SH	-	-

Function	Command		Locked resources													
			Logical						Storage					Physical	User authorization identifier	ROLE
			DB	SCH	SEQ	TBL	RTN	TRG	DSO	DSO (IX)	DSI	DSI (IX)	SCOP	DBS		
Expand DSI area automatically	rdbalmdsi	DSI	SH	-	-	-	-	-	-	-	SH	-	-	-	-	-
		DSI (IX)	SH	-	-	-	-	-	-	-	-	SH	-	-	-	-
Exclude DSI	rdbexdsi	DSI	-	-	-	-	-	-	-	-	SH	-	-	-	-	-
Update optimiza-tion infor-mation	rdbups	TBL	SH	SH	-	SH	-	-	MOD	MOD	MOD	MOD	-	SH	-	-
		DSO	SH	SH	-	SH	-	-	MOD	-	SH	-	-	SH	-	-
		DSO (IX)	SH	SH	-	SH	-	-	-	MOD	-	SH	-	SH	-	-
		DSI	SH	SH	-	SH	-	-	-	-	MOD	-	-	SH	-	-
		DSI (IX)	SH	SH	-	SH	-	-	-	-	-	MOD	-	SH	-	-
Print manage-ment info-mation	rdbprt	Definition information	SH	SH	-	SH	-	-	-	-	-	-	-	-	-	-
Print space inform-ation	rdbspcinf		SH	-	-	-	-	-	-	-	SH	SH	-	SH	-	-
Output DSI usage status	rdbudsi	DSI	SH	-	-	-	-	-	-	-	SH	-	-	-	-	-
		DSI (IX)	SH	-	-	-	-	-	-	-	-	SH	-	-	-	-
Create save data	rdbdmp	DSI specification	-	-	-	-	-	-	-	-	SH	-	-	-	-	-
Restore database	rdbrcv	DSI specification	-	-	-	-	-	-	-	-	EX	-	-	-	-	-
		DBS specification	-	-	-	-	-	-	-	-	EX	-	-	SH	-	-
	rdbadjrcv		-	-	-	-	-	-	-	-	-	-	-	-	-	-
Manage archive	rdblog		-	-	-	-	-	-	-	-	-	-	-	-	-	-
Obtain operation informa-tion	rdbinf	DIC specification	SH	-	-	-	-	-	-	-	SH	SH	-	-	-	-
		DSI specification	-	-	-	-			-		SH	SH		-	-	-
		DBS specification	SH	-	-	-	-	-	-	-	SH	SH	-	SH	-	-
Display execution informa-tion	rdbps		-	-	-	-	-	-	-	-	-	-	-	-	-	-
Display resource lock infor-mation	rdblkinf		-	-	-	-	-	-	-	-	-	-	-	-	-	-
Display save media informa-tion	rdbdvinf		-	-	-	-	-	-	-	-	-	-	-	-	-	-

Function	Command		Locked resources													
			Logical						Storage					Physical	User authorization identifier	ROLE
			DB	SCH	SEQ	TBL	RTN	TRG	DSO	DSO (IX)	DSI	DSI (IX)	SCOP	DBS		
Specify usage rules	rdbrtr	DIC specification	-	-	-	-	-	-	-	-	SH	SH	-	-	-	-
		DSI specification	-	-	-	-	-	-	-	-	SH	SH	-	-	-	-
		DBS specification	-	-	-	-	-	-	-	-	SH	SH	-	SH	-	-
Set usage rules	rdbrls	DIC specification	-	-	-	-	-	-	-	-	SH	SH	-	-	-	-
		DSI specification	-	-	-	-	-	-	-	-	SH	SH	-	-	-	-
		DBS specification	-	-	-	-	-	-	-	-	SH	SH	-	SH	-	-
Set access inhibition	rdbinh	DSI specification	-	-	-	-	-	-	-	-	SH	SH	-	-	-	-
		DBS specification	-	-	-	-	-	-	-	-	SH	SH	-	SH	-	-
Cancel access inhibition	rdbprmt	DSI specification	-	-	-	-	-	-	-	-	SH	SH	-	-	-	-
		DBS specification	-	-	-	-	-	-	-	-	SH	SH	-	SH	-	-
Pre-processor	sqlpc/sqlpcob		-	-	-	-	-	-	-	-	-	-	-	-	-	-
Disconnect and connect database space	rdbexspc	rdetach	-	-	-	-	-	-	-	-	-	-	-	MOD	-	-
		rattach	-	-	-	-	-	-	-	-	-	-	-	MOD	-	-
		rrp	-	-	-	-	-	-	-	-	-	-	-	SH	-	-
Disconnect and connect RDB directory file	rdbexdir		-	-	-	-	-	-	-	-	-	-	-	-	-	-

Function		Command	Locked resources													
			Logical						Storage					Phys ical	User authorization identifier	ROLE
			DB	SCH	SEQ	TBL	RTN	TRG	DSO	DSO (IX)	DSI	DSI (IX)	SCOP	DBS		
Application program (with dynamic SQL statement)		-	SH	SH	SH	SH	SH	SH	SH	SH	SH	SH	SH	-	SH	SH
Application program (without dynamic SQL statement)		-	SH	SH	SH	SH	SH	SH	SH	SH	SH	SH	SH	-	SH	SH
Start RDB		rdbstart	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Stop RDB		rdbstop	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Check RDB operation		rdbchksanity	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Shared buffer pool	Open	rdbcrbf	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Close	rdbdrbf	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Register correspondence	rdbconbf	SH	-	-	-	-	-	-	-	SH	SH	-	-	-	-
	Cancel correspondence	rdbdisbf	SH	-	-	-	-	-	-	-	SH	SH	-	-	-	-
	Display	rdbprtbf	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Degenearte index		rdbalidx	SH	-	-	-	-	-	-	-	-	EX	-	-	-	-
Load definition information on memory		rdbpldic	SH	SH	-	SH	-	-	-	-	-	-	-	-	-	-
create RDB dictionary		rdbcrdic	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Expand size of RDB dictionary		rdbaldic	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Relocate RDB dictionary		rdbgcdic	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Display RDB dictionary area status		rdbprdic	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Restore RDB dictionary		rdbrcvdic	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Create RDB dictionary save data		rdbdmpdic	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Monitor performance		rdbsar	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Specify recovery point		rdbsetrp	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Collect and restore connection and display status		rdbterm	-	-	-	-	-	-	-	-	-	-	-	-	-	-
		rdbcninf	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Execute SQL statement		rdbexecsql	SH	SH	-	SH	SH	SH	SH	SH	SH	SH	SH	-	SH	-
Recovery under XA		rdbxarcv	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Display list of XIDs under XA		rdbprxid	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Display RDB message (*2)		rdbprtmsg	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Function	Command	Locked resources													
		Logical						Storage					Phys ical	User authorization identifier	ROLE
		DB	SCH	SEQ	TBL	RTN	TRG	DSO	DSO (IX)	DSI	DSI (IX)	SCOP	DBS		
Register, change, display and delete resource	rdbhsrsc	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Output hot standby declaration and preopen information	rdbhsuty	SH	-	-	-	-	-	-	-	SH	-	SH	-	-	-
Operate RDB directory file for user log group	rdbscldir	-	-	-	-	-	-	-	-	-	-	-	-	-	-

DB:	Database
SCH:	Schema
SEQ:	Sequence
TBL:	Base table
RTN:	Routine
TRG:	Trigger
DSO:	Table DSO
DSO (IX):	Index DSO
DSI:	Table DSI
DSI (IX):	Index DSI
SCOP:	Scope
DBS:	Database space
DIC:	RDB dictionary
SH:	Shared mode
MOD:	Moderate share mode
EX:	Nonshared mode
-:	Not locked

*1 A table specified in the ON clause is locked in EX. A table specified in a triggered SQL statement is locked in SH.

*2 Can be used only for UNIX.

[Table: Relationships among lock strengths]

Preceding	**Succeeding**		
	EX	**MOD**	**SH**
EX	N	N	N
MOD	N	N	P
SH	N	P	P

P:	Multiprocessing of the same resource permitted between two processes
N:	Multiprocessing of the same resource not permitted between two processes

Glossary

Access

The operations of reading data from a storage device and writing data to a storage device. In this manual, reading data from a database and writing data to a database is called access.

Application program

Generally, programs used by users for their work are defined as application programs. In this manual, programs that read data from, or write data to, databases in response to business processing performed by users are known as application programs.

Base table

A table defined as a base table in logical structure definition. The data body is stored in a database space.

BTREE structure

The storage structure for an index. Consists of an index part and a data part. The index part folds the values of information about the correspondence between groups of columns that are index keys and base table data, and manages the pages of the data part. The data part holds data consisting of values of information about the correspondence between groups of columns that are index keys and base table data.

Related terms:

Index part, storage structure, data part

Bucket

A structure unique to RANDOM structures. A RANDOM structure uses a hash function as follows. The function specifies the collection of pages that sore data from the values in the group of columns that form the data key. The collection of pages is called a bucket.

Related term:

RANDOM structure

B-tree structure

An index system using an effective retrieval technique for speedily finding table data satisfying the retrieval condition.

SymfoWARE/RDB uses this effective data retrieval technique to record data storage location in the root-trunk-leaf format and retrieve data satisfying the retrieval condition in high speed. An index using this technique is calld "B-tree structure." SymfoWARE/RDB adopts the B-tree structure for the index that is a table storage structure.

Related term:

Hash structure

Client

he operational unit that requests data processing in the client/server model. When a database is used according to the client/server model, the application program requesting access to the server database operates in the client.

Client/server model

A model used in data processing in which the operational unit (client) that requests data processing and the operational unit (server) that executes data processing are established separately.

Cluster key

A grouping of columns that become a key for determining the page that stores data. Data for which cluster key values are equal is stored in the same bucket or overflow part bucket. Cluster keys are specified in the CLUSTER option of a table DSO definition. If cluster keys are omitted, the primary key of the corresponding table becomes the cluster key.

Column

A constituent element of a table. A relational database represents data using two-dimensional tables consisting of rows and columns.

Column attribute

Column data types and column constraints

Column constraint

Column constraints include the NOT NULL constraint (NOT NULL) and unique constraints (UNIQUE, PRIMARY KEY).

Related terms:

Unique constraint, NOT NULL constraint

Column name (item name)

The name attached to a column as defined in the schema. The column name is used to specify a column that is the subject of an operation in an SQL statement that manipulates data.

Commit

Makes the data manipulation of a transaction being processed take effect. The data manipulation in the transaction is physically reflected in the database. An application program deliberately controls a commit by issuing a COMMIT statement.

Related term:

Rollback

Connection

Connection refers to the relationship that connects a client to a server. A connection is made by specifying a CONNECT statement in an application program. Conversely, a DISCONNECT statement cuts off the connection.

Cursor

A cursor is an indicator that indicates a row to be manipulated. A cursor is defined by a cursor declaration statement. The OPEN and CLOSE statements start and stop cursor operations, respectively. The FETCH statement moves the cursor.

Cursor SQL statement

A cursor SQL statement is a data manipulation SQL statements that uses a cursor to specify rows to be manipulated.

Data manipulation SQL

A data manipulation SQL is an SQL statement used to reference, add, delete, or update a database.

Data part

A constituent element of a storage structure. The data part contains storage data corresponding to the table data (SEQUENTIAL structure or OBJECT structure) or data made up of index keys and table cluster keys (BTREE structure).

Related term:
    Index part

Data structure instance (DSI)

Expresses the storage structure for a table (base table). In addition to the information expressed in a DSO, a DSI expresses a mapping to a database space. The relationship between a DSO and a DSI can be one-to-one or one-to-many. A one-to-many relationship occurs only when a split table operation is applied.

Related terms:
    Storage structure, DSO

Database generation

Storing the initial data in empty base tables immediately after database definition. SymfoWARE/RDB utilities are used.

Database name

Many database can be created as units of administration and design on one server system. To identify each database uniquely, each is assigned a unique name (database name) on the server system.

Database space

An area that stores base tables and indexes. Database space is created by physical structure definition in SymfoWARE/RDB.

Deadlock

A stopped status that occurs when several transactions share use of a database. A deadlock is to a loop that occurs when several transactions wait for the same resource. Each transaction waits for another transaction to release the resource, and all transactions in the loop end up in stopped status.

Related terms:
    Transaction, exclusive control

DEFAULT clause

An element of the definition of a column in a table. If the following condition applies, the value defined in the DEFAULT clause is inserted. The data to be inserted in a column is not specified when a row is inserted in a table using the INSERT statement. If a DEFAULT clause is not defined for a column, a NULL value is inserted.

DELETE statement

The DELETE statement is data manipulation SQL statement used to delete rows from tables.

DSO

Expresses the storage structure for a table (base table). DSOs include table DSOs and index DSOs.

Related terms:
    Storage structure, DSI

Dynamic SQL

Dynamic SQL is a function for generating and executing SQL statements when executing an application program. In general, this function is used by general-purpose package programs.

Embedded exception declaration

An embedded exception declaration specifies the processing to be performed if an exception condition is issued when and SQL statement is executed in an application program. The WHENEVER statement is used to specify an embedded exception declaration.

Embedded SQL

SQL statements can be used as embedded SQL statements. Use embedded SQL statements to manipulate data in application programs written in high-level languages such as C language or COBOL. For example, when COBOL is used, specify an SQL statement embedded between EXEC SQL and END-EXEC.

Esql

A generic name of the compiler function for embedded SQL C programs or embedded SQL COBOL programs. Embedded SQL C programs and embedded SQL COBOL programs are called as shown below:

Embedded SQL C program:
    Esql-c
Embedded SQL COBOL program:
    Esql-COBOL

Excel

Developed by Microsoft Corporation, Excel is spreadsheet software that runs on a workstation.

Exception condition

When an SQL statement is executed in an application program, data to be processed may not be able to be found or an error may occur. Such a condition is known as an exception condition or exception. if an exception condition occurs while an application program is running, the status code corresponding to be exception condition is set in SQLSTATE.

Exclusive control

Control used when multiple users use a database. While one user is updating the database, other users cannot reference the data being updated or cause data conflicts by updating the data being updated. This function is generally called locking.

External routine

A user-created program (written in a language such as C) registered to a server as a dynamic link library called from an SQL statement. External routines have an advantage because they use C and other languages to enable complex processing such as formatting of character string data to be easily accomplished. Such processing with SQL functions has limitations.

With SymfoWARE/RDB, C can be used for creating external routines.

Function routine

A function that defines a user-created C program as a function in an SQL statement and processes it.

Users can create functions that they want and that are not provided by SymfoWARE/RDB, and they can use them in SQL statements in the same way as for numeric, data string, and date-and-time functions.

Hash function

The function used to specify the collection of pages that store data from the value of the group of columns that form the data key. Unique to RANDOM structures.

Related term:
    RANDOM structure

Hash structure

An indexing method used to find quickly the data of a base table that matches a search condition. A hash structure is a storage structure that makes fast data manipulation possible as follows. A hash structure uses a has function to determine the storage location of data using the value of a data key. SymfoWARE/RDB uses this mechanism in RANDOM structures that are the storage structures for base tables.

Related terms:
B-tree structure, RANDOM structure

Host variable

A variable for passing data between an application program and a database in SQL statements that manipulate data.

Index

Key data for increasing the efficiency of retrieving table data. If efficient data retrieval is not possible in SymfoWARE/RDB using only the primary key specified in the table definition, a supplemental positional key data can be created. This key data is called an index. An index can be created for each column of a table or for several combined columns. An index is established in a storage structure definition.

Index definition

A definition that indicates the columns of a table for which the index is created. To increase data manipulation efficiency, an index is required for frequently searched columns.

Index part

A constituent element of a storage structure. The index part is the portion that stores the data of an index for retrieving data stored in the data part. The index part is a constituent element of a BTREE structure.

Related terms:
Data part

Indicator variable

In high-level languages such as C language and COBOL, the indicator variable is a variable specified to be paired with an SQL data variable. Indicator variables are used when SQL statements are used to fetch data from, and update a data base. When SQL statements are used to update a data base, the indicator variable indicates whether or not data stored in the SQL data variable contains a null value. When SQL statements are used to reference a data base, the indicator variable indicates whether or not the execution result of the SQL statement has a null value stored in the SQL data variable. The indicator variable also shows the number of characters in character-string data stored in the SQL data variable.

INSERT statement

The INSERT statement is an SQL data manipulation statement used to add rows to a table.

Log group

A log environment split unit is called a log group. Each log file consists of a log management file, temporary log file, and archive log file. There are two types of log groups: system log groups unique in the default RDB system and multiple user log groups to be added and defined.

Logical structure

One of the structures of a database along with the storage structure and physical structure. The data structure that includes the schema, table, and column configuration, and column data types is called the logical structure. Constraints such as unique constraints, privileges, procedure routines, and triggers are also elements of the logical structure.

Related terms:
    storage structure, physical structure

Logical structure definition

An element of SymfoWARE/RDB database definition (also called schema definition). Table and view table definitions apply to logical structure definition.

Lotus 1-2-3

Developed by Lotus Development Corporation, Lotus 1-2-3 is spreadsheet software.

M host

The information processing system built into the M series general-purpose computer is called "M host"

Multi-database space

A base table or index DSI (data structure instance) allocated to multiple database spaces. A multi-database space can correspond to a large capacity DSI that exceeds the absolute capacity of a disk volume.

Multi-RDB

Multi-RDB means to activate multiple SymfoWARE/RDB systems with different RDB dictionaries in a single system configuration. This configuration allows linkage to each SymfoWARE/RDB environment for data access.

NTFS (Windows NT(R) file system)

A file system for use in the Windows NT(R) operating system. This file system supports a file recovery function, mass storage media, long file names, and strict access privilege control.

Non-cursor SQL statement

An SQL statement used for data manipulation, the non-cursor SQL statement does not use a cursor to specify rows to be manipulated. Instead, the rows to be processed are specified in the search condition specified in the SQL statement.

NOT NULL constraint

A constraint on a column of a table. This constraint prohibits rows in the table for which the value in the column is NULL.

NULL

The value of the data in a specified column of a row in a table that is undefined.

Number of key values that differ

Optimization information that refers to the number of key values of storage data in a database that differ from one another. For example, if all the storage data key values in a database are different, the number of different key values matches the storage data count. Conversely, if all the storage data key values are the same, the number of different key values is one.

OBJECT structure

A storage structure of SymfoWARE/RDB database base tables. This structure is applied to base tables for handling image, voice, and other types of multimedia data.

Related terms:
    SEQUENTIAL structure, RANDOM structure

Open Systems Interconnection/Remote Database Access (OSI/RDA)

OSI is an international standard for interconnecting different types of computers. This standard is being developed by the International Organization for Standardization (ISO) and the International Telegraph and Telephone Consultative Committee (CCITT). RDA is an OSI application layer standard for interoperation of databases between different types of systems. RDA makes it possible to perform processes such as retrieval and update using the SQL database language in databases on different types of systems.

Operating environment file

A file for defining the operating environment used when an application program is executed. Operating environment files include client operating environment files and server operating environment files. A server operating environment file is used for tuning the system-provided files for the application program.

Optimization

Determining the most efficient processing procedures for the search conditions by investigating tables bound to SQL statement instructions.

Optimization information

Information that is the basis for optimization in SymfoWARE/RDB. Optimization information includes the amount of base table data (number of rows), the number of levels of indexes, and the number of different key values. SymfoWARE/RDB integrates and evaluates SQL statements and optimization information and determines the most efficient data manipulation processing procedures. Optimization information is collected using the rdbups command.

Overflow part

A constituent element of a storage structure unique to a RANDOM structure. When storage data can not be collected in prime part pages because of excess data, the overflow part provides reserved pages for storing the excess data. The overflows part consists of these reserved pages.

Related terms:

Prime part

Overflow pointer

When storage data is not collected in the prime part, the storage data is collected in the pages of the overflow part. (This condition does not apply to a page split.) When this condition occurs, an overflow pointer connects the relevant page of the prime part to the relevant page of the overflow part. The overflow pointer is unique to RANDOM structures.

Related terms:

Overflow part, prime part

Page

The smallest unit of I-O for a database. The size of a page is determined by the number of rows to store in the page.

Page split

A self-adjustment function unique to B-tree structures. When the amount of data stored in a given page exceeds a fixed value, this function relocates storage data between separate pages (including empty pages). This relocation evens the amount of data stored in pages to maintain a balance in processing efficiency.

Related terms:

B-tree structure

Parallel query

To raise the information processing efficiency in handling a large volume of data, a database is divided into several DSI units. The parallel processing of DSI units is called parallel query processing.

Physical structure

One of the database structures along with the logical structure and storage structure. The database space located on a magnetic disk volume is called the physical structure. A database space is an aggregate of fixed-size blocks.

Related terms:

Storage structure, logical structure

Physical structure definition

An element of a SymfoWARE/RDB database definition for creating database spaces.

Pointer variable

A host variable declared as a pointer in an embedded C program. An area dynamically obtained using the malloc function or obtained outside an embedded SQL declare section can be specified in an SQL statement.

Primary key

A column or group of columns that can uniquely specify a row of a table. One of the major elements for designing logical structure. The primary key is specified using PRIMARY KEY in the unique constraint of the table definition.

Prime part

A constituent element of a storage structure that is unique to RANDOM structures. Storage data corresponding to table data is obtained from a page (bucket) of this part first.

Related terms:

Overflow part

Private sort work area

One of the sort work areas of the SymfoWARE/RDB system. A private sort work area is prepared by the user. The directory is specified in the operating environment file specific to an application program.

Related terms:

Shared sort work area, sort work area

Procedure routine

The definition of a database processing procedure by SQL is called a procedure routine.

Procedure routine definition

Defining a procedure routine with a logical structure is called procedure routine definition.

Related term:

Procedure routine

RANDOM structure

A storage structure of SymfoWARE/RDB database base tables. A storage structure in which hash function indexing is used in the data storage method is called a RANDOM structure.

Related term:

OBJECT structure, SEQUENTIAL structure

Raw device

A disk area that can be accessed regardless of the UNIX file system is called a raw device. The raw device is a character-type special device created using the UNIX utility. It is ordinarily indicated by a node name created under /dev/rdsk/.

RDB configuration parameter

Information, such as where the RDB directory file is located, that defines the operating environment of a SymfoWARE/RDB system. Operating environment setup is performed in accordance with the description of the RDB configuration parameter during when the RDB system is activated.

Related terms:

RDB configuration parameter file

RDB configuration parameter file

A file containing RDB configuration parameters is called an RDB configuration parameter file. The RDB configuration parameters are used to define SymforWARE/RDB operating environments such as an RDB directory file allocation destination.

RDB dictionary

A file in which user database definition information is stored. An RDB dictionary is represented in a table like those the user defines. This table is called a system table.

Related terms:

System table

RDB library

When an application program that uses SQL statements is executed, shared objects provided by the SymfoWARE/RDB system are called to perform database processing. The shared objects also call shared objects provided by Windows NT(R) system. The load module of an application program that uses SQL statements must be dynamically linked to these shared objects. In this manual, these shared objects are called the RDB library.

Read-only cursor

In an SQL data manipulation statement, a cursor that cannot be used to update and delete is called a read-only cursor. The cursor declaration specifies whether a cursor is read-only. For example, if two or more tables are specified in the FROM clause of a query specification, the cursor is a read-only cursor.

Related terms:

Updatable cursor

Reference mode

A mode denoting strength of exclusion. Also called shared mode, the reference mode represents the strength of data locks in data manipulation. Data that is locked in reference mode can only be referenced by other transactions and cannot be updated. In general, the execution of other transactions waits until a commit is performed on the transaction that obtained the data lock.

Related terms:

procedure routine, function routine

Relational database

Database used in SymfoWARE/RDB that represents data using two-dimensional tables consisting of rows and columns. Database operations are performed using the SQL database language.

Remote database

When a database is distributed into multiple servers, the processing mechanism to access the database from one application program is called "remote database"

Remote database access - service (RDA-SV)

A software product provided by Fujitsu Limited that implements distributed database functions to be used by PC spreadsheet software and application programs on a server system

---

Role

A group of privileges required for one transaction. To specify at one time the privileges required for one transaction, define a role. For efficiency of privilege management, a role can be defined to grant the role privileges to all users who perform the relevant transaction.

Rollback

Nullifying the data manipulation of a transaction that is being processed. The two kinds of rollback can be performed. Rollback in data manipulating SQL statement units can be the occurrence of an exception condition in a data operation. Rollback in transaction units can be the deliberate execution of a ROLLBACK statement by an application program.

Related terms:

Commit

Routine

Procedure routines and function routines are generally called routines. Related terms:

procedure routine, function routine

Routine name

The name of a procedure routine or function routine is called a routine name.

Row

A row is one of the components of a table. In relational data bases, data is expressed in terms of two-dimensional tables containing rows and columns.

Row identifier

The rows of a database table are uniquely identified. A user can manipulate a row using the row identifier fetched by the single row SELECT statement or by using the FETCH statement.

Scalable log operation

Splits log environments into multiple log groups in the RDB system for definition.

Schema

A constituent element of a database. SymfoWARE/RDB performs data analysis using information analysis system AA/BRMODELLING to create tables or view tables.

Schema definition

Logical structure definition that includes defining the base tables and view tables that constitute a database, the column configuration of each table, and the attributes of each column. In addition, privileges, a procedure routine, and triggers can be defined. This is called "schema definition"

Scope

When a table is accessed by use of a SQL statement for data manipulation, the access range can be limited. The access range of is called the scope. The scope function limits the data manipulation range for each user by applying or canceling the scope for the user who accesses the table.

Security

SymfoWARE/RDB assures security for resources such as schemas, tables, procedure routines, and database spaces.

Sequence

A function that generates a value unique within an entire system. A sequence can be specified in an SQL statement to use the generated values primarily for creating primary key values in a table.

SEQUENTIAL structure

A storage structure of base tables in a SymfoWARE/RDB database. This storage structure is applied to base tables for adding rows (records) in the order of data generation, as in a historical journal.

Related terms:
    RANDOM structure

Server

The operational unit that executes data processing in the client/server model. When a database is used according to the client/server model, the database operates in the server.

Shared buffer pool

A buffer for accessing a database (also called a shared buffer). Because data can be shared by multiple application programs, a shared buffer pool can minimize the number of inputs and outputs of data application programs accessing in common.

Shared buffers

A buffer for accessing a database (also called a shared buffer pool). Because data can be shared by multiple application programs, a shared buffer can minimize the number of inputs and outputs of data application programs accessing in common.

Related term:
    Shared memory

Shared memory

A memory area that can be mutually referenced by more than one process. In a SymfoWARE/RDB system, shared buffers and the log collection area are placed in shared memory.

Related terms:
    Shared buffers

Shared sort work area

One of the sort work areas in the SymfoWARE/RDB system. The shared sort work area is provided by the user and is specified in the common application environment file of the directory system.

Related terms:
    Private sort work area, sort work area

Sort work area

A work area the SymfoWARE/RDB system uses on magnetic disk. The SymfoWARE/RDB system saves data in a work table when it needs to save an intermediate result while manipulating data. If sorting is required, the SymfoWARE/RDB system uses sort work. A work table or sort work of up to a specified fixed amount uses virtual memory. When a work table or sort work exceeds the fixed amount, it uses an area on magnetic disk. This area is called a sort work area. A shared sort work area is shared by the entire SymfoWARE/RDB system; own sort work areas are used privately by each application program or command.

Related terms:
    Shared sort work area, private sort work area, working sort area, work table

---

Split condition

When a split table operation is applied, the rule for dividing data into split units is called the split condition. The split condition specifies a list of column names representing split keys and a list of dummy values in which "?" is specified. A split condition is specified in table DSO definition.

Related terms:

Split key, split key value

Split key

he key used to locate data by dividing it into split units when applying split table operation. This key corresponds to a specific column (or multiple columns) of a table.

Related terms:

Split value, split condition

Split value

The value for a "?" specified in a split condition in a table DSO definition that is specified using a constant when the table DSI is defined. This value applies to split table operation. Data stored in the defined DSI is a row in which evaluation of the condition is true when the "?" in the split condition is replaced by the split value.

Related term:

Split key, split condition

Split table operation

SymfoWARE/RDB allows splitting a single logical base table into physically separate tables. Splitting tables speeds up the access to a large database and ensures practicality from an operations standpoint. For example, a sales table for all stores can be split by store. An application program can apply split table operation without giving it special consideration. Moreover, database tuning tasks and database save operations can be performed independently and concurrently in the split units.

SQL

SQL is a standard database language for performing database definition and data manipulation. SymfoWARE/RDB basically conforms to the protocols of the international standards IS 9075, JIS X3005, and ANSI X3.135.

SQL embedded host program

In an application program that manipulates a data base, the parts of the program that perform database processing are written using SQL statements. The parts of the program that perform other kinds of processing are written using a programming language such as C language or COBOL. Application program in which SQL statements are embedded are called SQL embedded host programs. The following terms are used for application programs that contain embedded SQL statements:

a. SQL embedded C program
b. SQL embedded COBOL program

SQL embedded host program is the generic term for a) and b).

Status variable

The status variable, SQLSTATE, reports the processing result of an SQL statement to an application program. When an SQL statement is executed, the status code for the execution result is stored in the status variable.

Storage data

Data that is stored in a database space. The rows and columns of tables represent the logical aspect of data, and stored data represents the physical aspect of data.

Storage structure

A database structure along with logical structure and physical structure. Storage structure physically locates data logically expressed as rows and columns of tables in a database as storage data. A storage structure is expressed using DSO and DSI. The storage structures for base tables are RANDOM structures, SEQUENTIAL structures, and OBJECT structures; BTREE structures are the storage structures for indexes.

Related terms:
    Physical structure, logical structure

Storage structure definition

The definition of mapping between tables and database spaces. The two kinds of storage structure definition are data structure organization (DSO) definition and data structure instance (DSI) definition.

Structure host variable

A host variable declared as the structure type is called a structure host variable. When a structure host variable is declared, each member is handled to correspond to individual columns in a database so that multiple-column data can be manipulated in row units. Multiple rows can also be inserted at one time. A structure host variable can be used in combination with a pointer variable to simplify the application program and improve maintainability.

System table

A table that manages definition information for, for example, databases and schemas defined by a SymfoWARE/RDB user. The system table is also called the RDB dictionary.

Table

In a relational database, data is represented using two-dimensional tables consisting of rows and columns. The two kinds of tables are base tables and view tables.

Table constraint

Constraints on tables are unique constraints (UNIQUE or PRIMARY KEY)

Related terms:
    Unique constraint

Table declaration

A table declaration declares the schema in which a table is located. If a table declaration is specified, the schema name need not be specified for a table in data manipulation statements. Using a table declaration simplifies the specification of table names. Using a table declaration also helps to make an application program independent of a data base.

Table name

A name attached to table. Table names are set in schema definitions. Table names are used to specify the tables to be made the subjects of operations in SQL statements that manipulate data.

Temporary table

A table created specifically for a user of an application program. Multiple users can use temporary tables with the same table name. To temporarily save data being processed by an application program, a temporary table can be used independently of other application programs.

Transaction

The unit that guarantees consistency of sequential data operations. A database can be accessed and updated serially or updated by arranging a number of SQL statements. If a problem such as an unexpected system crash occurs during sequential data operations, database recovery status can be based on units of transactions.

Trigger definition

Trigger definition defines table data manipulation (insert) in conjunction with other table data manipulations (insert, delete, update).

Unique constraint

A constraint on a table or column. This constraint requires that a table cannot have multiple rows having the same value in a column or combination of columns.

Updatable cursor

In an SQL data manipulation statement, a cursor that can be used to update and delete is called an updatable cursor. The cursor declaration specifies whether a cursor is updatable. For example, if a table specified in the FROM clause of a query specification satisfies just one condition, the cursor is an updatable cursor.

Related terms:

Read-only cursor

Update mode

A mode denoting strength of exclusion. Also called nonshared mode, the update mode represents the strength of data locks in data manipulation. Data that is locked in update mode cannot be manipulated by other transactions. In general, the execution of other transactions waits until a commit is performed on the transaction that obtained the data lock.

Related term:

Reference mode

UPDATE statement

The UPDATE statement is a SQL data manipulation statement used to update data in table rows.

Upgrade

Index update processing. In conjunction with data update, insertion, and deletion processing in a table, this process updates indexes attached to that table to reflect the latest status. This processing is called upgrading indexes.

View definition

The definition of a view in a logical structure definition. The view definition defines the portion of the base table that forms the view, the name of the view, and the name of each column.

WHERE clause

In an SQL data manipulation statement, the WHERE clause is a search condition that specifies rows to be manipulated.

Work table

A temporary table the SymfoWARE/RDB system uses for manipulating data. Intermediate results often must be saved when the SymfoWARE/RDB system executes operations on data. A table that saves intermediate results is called a work table. A work table of up to a specified fixed amount uses virtual memory. When a work table exceeds the fixed amount, it uses a sort work area on magnetic disk. The amount of virtual memory to use can be specified in the application environment file.

Related terms:

Sort work area

Working sort area

A temporary area used for sorting by the SymfoWARE/RDB system. If the SymfoWARE/RDB system has to sort data during data manipulation, loading, or unloading, it sorts this data in a work area for sorting. This work area for sorting is called the working sort area. Up to a fixed amount of virtual storage is used for a working sort area. However, when this amount is exceeded, a sort work area on magnetic disk is used. The amount of virtual storage to be used can be specified in the operating environment file.

Related terms:

Sort work area